Inverting Film Negatives Using Multilayer 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

2.1 Gradient Boosted Decision Trees

The baseline model selected for this project is a gradient boosted decision tree. A decision tree[Breiman et al. (1984)] is a machine learning algorithm that repeatedly "splits" the training data, branching off in a tree-like fashion, based on input features and values that give the most homogenous groupings of outcome labels.

Gradient boosted decision trees in turn is an ensemble model based on the basic decision tree. Specifically, it constructs multiple trees sequentially, each trying to predict the residuals, or prediction errors, produced by the previous tree. As shown in figure 1 below, this process introduces non-linearity into the model, and promotes a more accurate "fit" to the underlying function that we are trying to learn.

This project employs a more efficient variant of regular GB based on LightGBM[Ke et al. (2017)] and histogram-based GB, which, in broad strokes, buckets continuous feature values into discrete bins instead of traversing through all values to find the split points. During training, GB uses these bins to construct feature histograms from which the decision trees would find their split points.

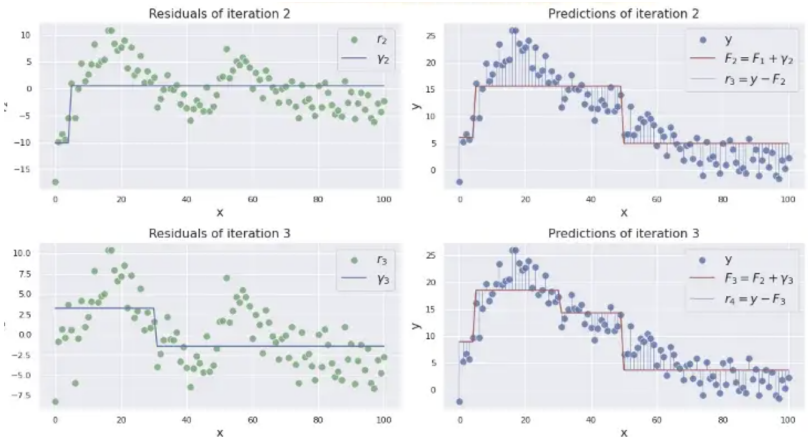

Figure 1: The gradient boosting process[T.Masuri, 2022]

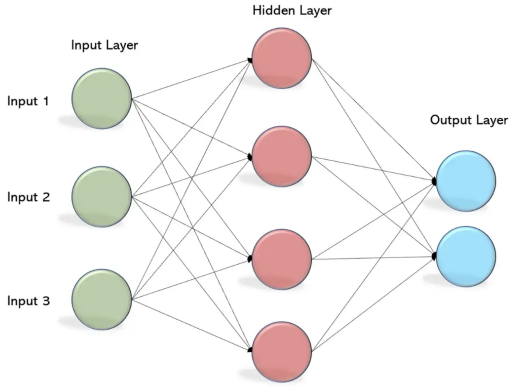

Figure 2: MLP, or multilayer perceptron

2.2 Multilayer perceptrons

This project builds its model using a multilayer 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 2)

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 $x_1, \ldots, x_d$,

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

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 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.

2.3 Sparse Autoencoder

A sparse autoencoder is a variant of MLP. Specifically, an autoencoder "encodes" data by transforming the input matrices into a representation of itself. A sparse autoencoder enforces sparsity by applying, among other methods, L1 regularization, which is expressed as the following equation:

$$\arg\min_{\theta} L(\theta) + \lambda \sum_{j=1}^{d} |\theta_j|$$

L1 regularization enforces sparsity by penalizing the sum of the absolute values of the parameters, pushing less significant and smaller weights to 0. A hyperparameter $\lambda \geq 0$ controls the magnitude of the penalty, or the strength of regularization.

2.4 Brief Intro To Print Film

To assist readers in understanding exactly what is the learning objective of my model, here I provide some (oversimplified) background on how print film and color paper work.

When a piece of print film is exposed, a negative color image is generated. This negative is then projected onto a piece of color paper by an enlarger, which negates the colors in the image again, resulting in an image with positive colors that humans can appreciate. Two non-linear mapping functions are involved in this process: 1, print film maps colors projected onto it by the lens into negative colors, and 2, the color paper maps those negative colors once again into positive colors. Shown below in Figure 3 are the approximate mapping functions for each color channel for Fujifilm's current color paper products.

What I'm interested in learning in this project is the second mapping function. Specifically, the mapping between a camera-scanned image of the print film, and the final color-positive image. This would enable photographers to "print" their negatives directly and digitally, without requiring access to any special equipment (e.g., enlargers and scanners) and materials (e.g., color paper and chemistries requried to develop it).

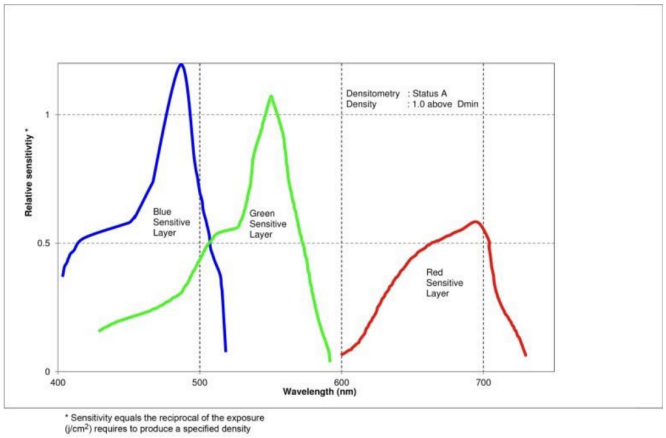

Figure 3: Mapping functions (aka response curve) of FUJIFILM's color paper[fuj]

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. One concern with this approach, however, is that since the film and the paper are scanned separately, aligning and matching the two would be very difficult. In other words, it would be very difficult to accurately match X with its Y on a pixel level. Furthermore, dust is common during scanning, and they introduce noise into the dataset. Therefore, for this project I will be using manual inversions of film scans to approximate the color paper prints. This enables me to create pixel-to-pixel mappings between input and outcome matrices, and keeps the noise distribution (i.e., dust and other distortions) consistent between the two. Although this approach indeed introduces the uncertainty of my inversion techniques, I have verified to the best of my ability (by comparing my inversions and prints) that manual inversions can be adquate approximations of paper prints.

In the project repository [link here] readers 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 desktop. So I've downsized all images to 1/16 their original size, and renamed them for easier indexing. This dataset is used for quick verification of new models, and for the ablation study in Section 6 that trains the final model on a smaller scale dataset. A larger version named "Dataset_new", scaled to 1/4 original size, can be found in this [Google Drive folder]. 1/4 was chosen because it is the largest dataset that can still fit into my computer's memory and VRAM. The final model is trained using this larger dataset. Sample images from the dataset are shown in Figures 4 and 5.

The dataset consists of 51 training images, labeled 0-14 (multiple images are in each tiff file), 8 validation images labeled v0-v7, and 6 test images labeled t0-t5, consitituting a train-valid-test split ratio of roughly 80-10-10. There are also three additional images labeled a0-a2 for use in the ablation study.

4. Training And Validation Of Models

4.1 Baseline Models

Since the model is supposed approximate a non-linear function, one baseline model that can be used to fufill this task is the gradient-boosting (GB) ensemble model. The intuition behind selecting GB is that, as discussed in the preliminary section, each iteration in the boosting process predicts the residuals from the previous iteration, which resembles the process of approximating the shape of a non-linear function. Since my learning objective is also to approximate an underlying mapping function, GD seems an appropriate off-the-shelf baseline model to begin.

To build the GB model, I use the HistGradientBoostingRegressor in sklearn, the details of which were dicussed in the Preliminary section, to speed up training on my large and low-dimension dataset. Since the GB regressor does not accept RGB tuples as its outcome, I split the data into color channels of red, green, and blue, and fit the model to each channel separately.

Results:

	Train Accuracy	**Validation Accuracy**	**Test Accuracy**
Red	0.7263	0.4916	0.6621
Green	0.7842	0.7592	0.8370
Blue	0.8253	0.4279	0.5013

4.2 Simple MLP Model

I experimented with two structures for a simple MLP model: model A takes and returns an RGB tuple, and model B learns all three color channels in parallel. Model A has shape 3*32*64*3, with ReLU activation and no regularization. Model B has three parallel networks each with shape 1*32*32*1, also with ReLU activation and no regularization. Model A achieved training and validation accuracy of 0.7782 and 0.6350 respectively, and Model B performed very similarly to the GB model.

4.3 Sparse Autoencoder

To create the sparse autoencoder model, I apply L1 regularization on model A to enforce sparsity for the trained weights. This results in higher accuracy across the board compared to simple MLP models:

Results:

Figure 4: An inverted sample from a dataset

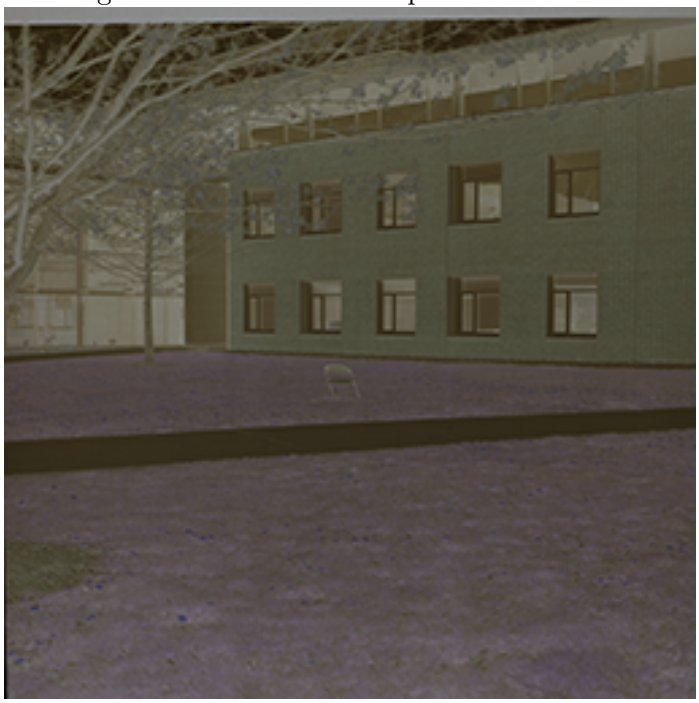

Figure 5: The original camera scan of the same image

	Train Accuracy	Validation Accuracy	Test Accuracy
Sparse Autoencoder	0.8847	0.7895	0.6821

I select the sparse autoencoder as my final model. It was a tough choice between the GB model and the sparse autoencoder model: while the sparse autoencoder achieves higher accuracy on paper, its mispredictions often produce visual artifacts, while the GB model generates a more "natural" image with no artifacts, but is less accurate. However, the sparse autoencoder was eventually selected because compared to the GB model, it generated technically more accurate test results that, although contain artifacts, are more correct in terms of white balance. A comparison between the two models' results is shown in Figure 6 through 9.

5. Results

Final Model Results:

	Train Accuracy	Validation Accuracy	Test Accuracy
Sparse Autoencoder	0.8847	0.7895	0.6821

As discussed in the previous section, the main issue with this final model is the existence of visual artifacts. After some testing and observing that visual artifacts are virtually non-existent in models that predict each color channel individually (see Figure 7 and 11), I've come to speculate that this has to do with the structure of the network itself: treating each RGB pixel as a sigular unit means that mispredictions are always clearly visible when they happen. Whereas a misprediction in one of the color channels could be less catastrophic. Distribution shift is also introduced into the testing dataset via means of images with different white balance or color temperatrues. This means that the color white in test images could be different from the white in training images. This is done by including different kinds of print film, and images taken under different lighting conditions in the test set.

6. Ablation Study

I performed the ablation study by reducing amount of data.

To reduce the amount of data, I scaled the dataset to 1/16th its original size, and retrained the model. The performance of this model is shown in the table below:

	Train Accuracy	Validation Accuracy	Test Accuracy
Regular-size data	0.8847	0.7895	0.6821
Reduced-size data	0.8182	0.7187	0.6093

As we can see the performance of the model is lower when trained with a smaller dataset. This is fair because the model is attempting to learn the mapping between pixel values, and

Figure 6: A sample test image

Figure 7: Ditribution shift between overcast fall afternoon, and bright winter morning

Figure 8: Result generated by the GB model

Figure 9: Result generated by the sparse autoencoder

Figure 10: Enlargement showing visual artifacts in the image above

Figure 11: Manual inversion of the same negative

Figure 12: Result from simple MLP model B

for 16-bit images in the dataset, there would be 2^{48} total colors to be learned (which could also be a good thing, since it would be very difficult to build an overparameterized network for this task). Therefore, limiting the size of the training set should limit the model's ability to learn accurately. However, we do observe that the performance of the model is not severelly degraded, which could imply that our model is indeed learning to generalize an underlying function, and not just memorizing the mapping between pixel values it has seen.

7. Discussion and Conclusion

Although accuracy metrics are promising, I do not think that the current model is suitable for deployment in this application which requires critial color accuracy. Beyond the issue of visual artifacts, one other potential issue I have noticed in this project is the problem of white balance. Specifically, the color white is presented as different colors under different lighting conditions. In the darkroom, correcting white balance requires trial-and-errors, often passing the image through color-correcting filters. However, since all inverted images in my dataset have been corrected for white balance, this could introduce a subtle distribution shift that confuses the model. To move toward a more accurate and realistic model in the future, I might attempt to, counterintuitively, build a model that less resembles the color-corrected image but more honestly reflect the state of the film negative before color correction.

References

Fujicolor crystal archive paper type ca. URL https://www.fujifilm.com/us/en/business/photofinishing/paper-lab-products/type-ca.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. Routledge, 1984.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.