Project 1 Report: Color Calibration Tool

By Vedanshi Shah

Due Sunday, March 23, 2025 at 11:59 PM

COSI 149B: Practical Machine Learning with Big Data

Prof: Pengyu Hong

1. Introduction

Color perception and reproduction are critical aspects of imaging systems, especially in fields like computer vision, photography, and industrial automation. The goal of this project was to develop an automated color calibration tool that corrects observed color values to match ground truth references using a combination of statistical models, machine learning techniques, and deep learning approaches.

This report details the pipeline architecture, methodologies, evaluation metrics, and results of the implemented color calibration system.

Additionally, this report presents a detailed workflow for processing images, detecting objects using YOLOv8, and extracting RGB color values for analysis. The project involves dataset preparation, object detection, pattern recognition, and color extraction, providing a structured approach to image-based machine learning tasks.

2. Problem Statement

2.1 Background

Different imaging devices perceive colors differently due to variations in **lighting conditions, sensor sensitivity, and environmental factors**. This necessitates a **color calibration process** to align observed colors with their true reference values.

2.2 Objective

- Develop a **color calibration model** that can correct the observed RGB values to match true values.
- Compare multiple models including:
    - **Simple Difference Model**
    - **Weighted Difference Model**
    - **Neural Networks**

    - **Random Forest Regressor**
- Evaluate model performance using **Mean Squared Error (MSE), R² Score, and other statistical metrics**.

3. Dataset and Preprocessing

3.1 Dataset Description

The dataset used for training and evaluation contained the following color attributes:

- **Observed RGB Values:** Captured from a camera or sensor.
- **True RGB Values:** The ground truth reference colors.
- **Channel-Specific Adjustments:** Red, Green, and Blue channel correction factors.

3.2 Preprocessing Steps

- **Handling Missing Data:** Applied methods such as dropna(), mean imputation, and interpolation.
- **Feature Engineering:** Constructed color difference features between observed and reference values.
- **Normalization:** Applied standardization to ensure **better convergence in neural network models**.

4. Methodology

4.1 Model Selection

We implemented and tested four different models for color calibration:

4.1.1 Simple Difference Model

- Computes the difference between true and observed values for each RGB channel.
- Applies these differences as correction factors to new observations.
- **Formula:**

$$R_{corrected} = R_{obs} + (R_{true} - R_{obs})$$

4.1.2 Weighted Difference Model

- Learn trainable weights for each color channel correction factor.
- Trained using **gradient descent** to minimize MSE.
- **Optimization via backpropagation**.

4.1.3 Neural Network Model

- A **fully connected feedforward network** with:
    - **Input Layer:** Observed RGB + channel-wise correction factors.
    - **Hidden Layers:** ReLU activation, dropout regularization.
    - **Output Layer:** Predicted true RGB values.
- **Loss Function:** Mean Squared Error (MSE).
- **Optimizer:** Adam.

4.1.4 Random Forest Model

- A **tree-based ensemble regressor** that maps observed colors to true values.
- Trained with **100 estimators** to reduce variance and improve stability.

5. Model Evaluation

5.1 Metrics Used

The following performance metrics were computed:

- **Mean Squared Error (MSE)**

  Measures the average squared difference between predicted and actual RGB values.

$$MSE = \frac{1}{N}\sum(Y_{true} - Y_{pred})^2$$

- **Root Mean Squared Error (RMSE)**

  Square root of MSE, providing error values in the same scale as the original data.
- **R^2 Score (Coefficient of Determination)**

  Measures how well predictions match the actual values, where **1.0** is a perfect fit.

6. Implementation

6.1 Pipeline Overview

The implementation of the **Color Calibration Tool** followed a structured and iterative approach, where multiple tools and techniques were explored before selecting the most effective model. We prioritized a **data-centric approach** over a **model-centric approach**, emphasizing improvements to the quality and diversity of the dataset rather than merely fine-tuning models.

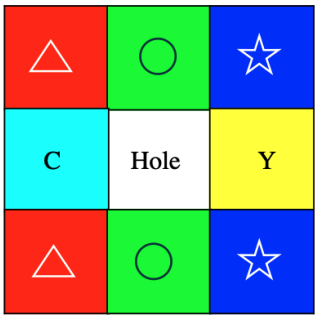


Figure: Original color calibration tool Iteration

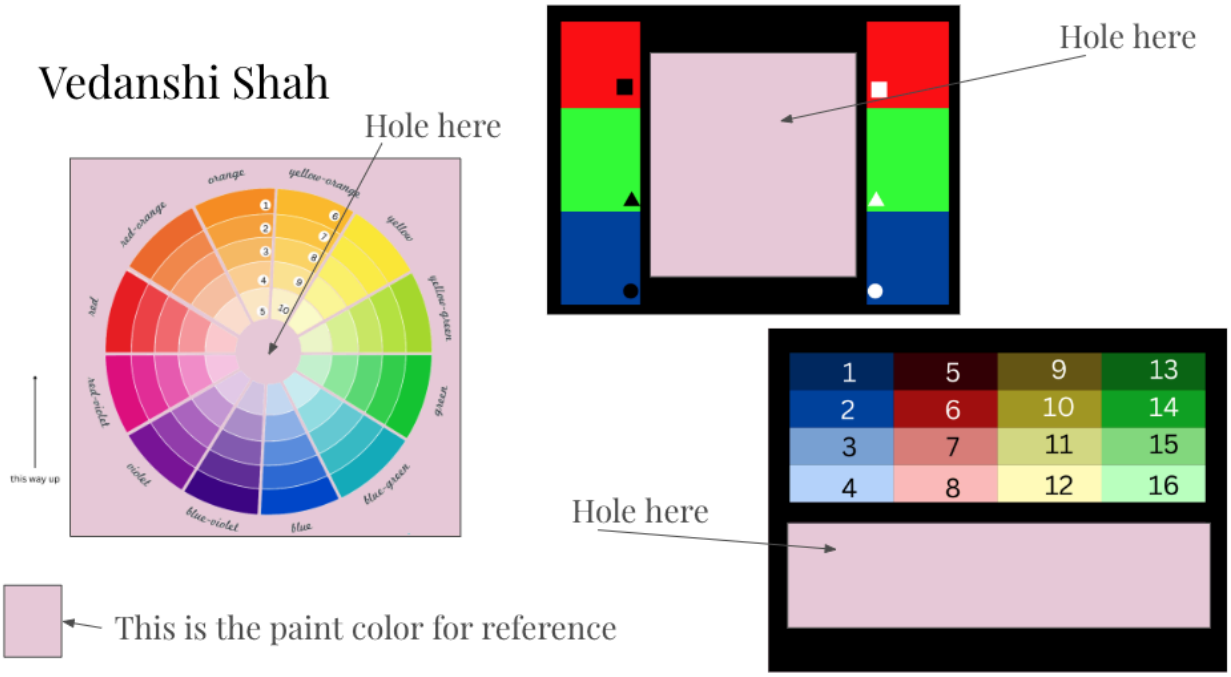
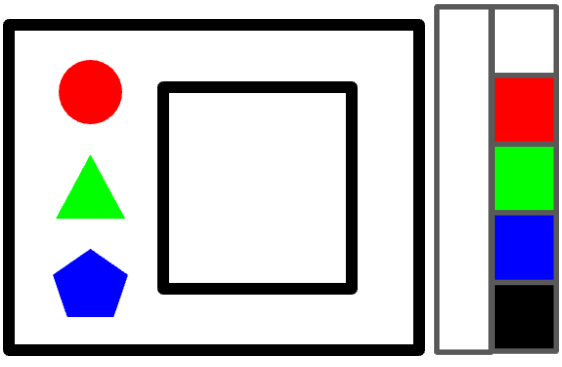


Figure: Color Calibration tool Iteration 1 development



(a) (b)

Figure: color calibration tools Iteration 1 based on the group consensus

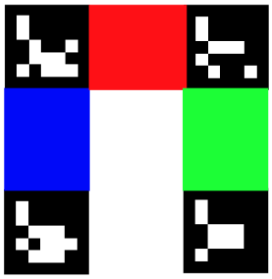


Figure: We added fiducials to detect the landmarks better

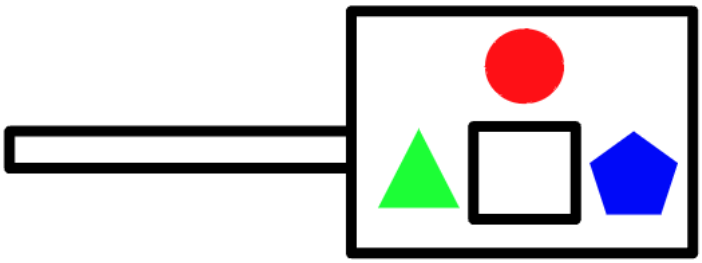


Figure: color calibration tool Iteration 2 to create a smaller version to be able to deter from any shadows and reduce the drastic lighting effects

Step 1: Data Collection and Preprocessing

Collaborative Data Collection

- All participants in the class contributed to data collection by capturing **images under diverse lighting conditions and angles** using different devices.
- This collaborative effort ensured that the dataset was rich, diverse, and representative of real-world scenarios where color perception varies due to external factors such as:
    - Variations in natural and artificial light.
    - Camera sensor inconsistencies.
    - Angular changes that affected color refraction.

Sample Name	Sample Number	File Name	True Value			Observed Value			Red Value			Green Values			Blue Values		
			True R	True G	True B	Observed R	Observed G	Observed B	Red R	Red G	Red B	Green R	Green G	Green B	Blue R	Blue G	Blue B
Lilac Lane	1002-4B	big_1002-4B_1.jpg	227	209	223	236	207	210	208	169	220	216	209	163	251	155	123
		big_1002-4B_2.jpg	227	209	223	150	140	150	103	85	146	94	95	82	74	56	52
		big_1002-4B_3.jpg	227	209	223	141	144	155	75	78	148	99	114	100	85	73	72
		big_1002-4B_4.jpg	227	209	223	195	186	195	139	122	203	177	184	151	183	140	123
		big_1002-4B_5.jpg	227	209	223	190	181	188	96	75	135	124	123	104	153	113	100
		1002-4B_1.jpg	227	209	223	158	148	148	97	79	132	109	107	88	153	138	127
		1002-4B_2.jpg	227	209	223	145	129	134	90	67	129	94	90	77	128	98	90
		1002-4B_3.jpg	227	209	223	197	158	159	120	84	136	165	148	122	151	94	79
		1002-4B_4.jpg	227	209	223				99	87	151	127	129	111	135	119	115
		1002-4B_5.jpg	227	209	223	169	159	160	141	123	196	189	198	182	192	160	146
Dreamy Memory	1004-4B	big_1004-4B_1.jpg	234	200	217	234	197	206	218	182	222	217	206	168	248	147	118
		big_1004-4B_2.jpg	234	200	217	176	159	181	153	135	208	142	142	122	111	83	77
		big_1004-4B_3.jpg	234	200	217	189	171	194	136	115	196	169	172	148	169	129	118
		big_1004-4B_4.jpg	234	200	217	124	120	147	77	67	136	93	102	91	77	65	66
		big_1004-4B_5.jpg	234	200	217	173	157	179	82	62	117	111	109	95	146	108	97
		1004-4B_1.jpg	234	200	217	158	152	174	101	96	185	132	140	117	158	125	116
		1004-4B_2.jpg	234	200	217	192	185	199	171	168	218	169	179	156	187	182	176
		1004-4B_3.jpg	234	200	217	192	177	195	141	122	211	173	182	150	181	139	128
		1004-4B_4.jpg	234	200	217	145	133	142	179	175	218	157	162	139	168	152	141
		1004-4B_5.jpg	234	200	217				127	106	161	125	121	107	112	84	78

Figure: Collected Data that we trained it on

Data Augmentation for Robustness

To further enhance the diversity of the dataset, we applied **data augmentation** techniques such as:

- **Brightness and Contrast Adjustments:** To simulate different lighting conditions.
- **Rotation and Flipping:** To account for varying camera angles.
- **Noise Injection:** To mimic real-world image inconsistencies.

This augmented dataset ensured that the models were exposed to a wide range of possible input conditions, leading to **better generalization**.

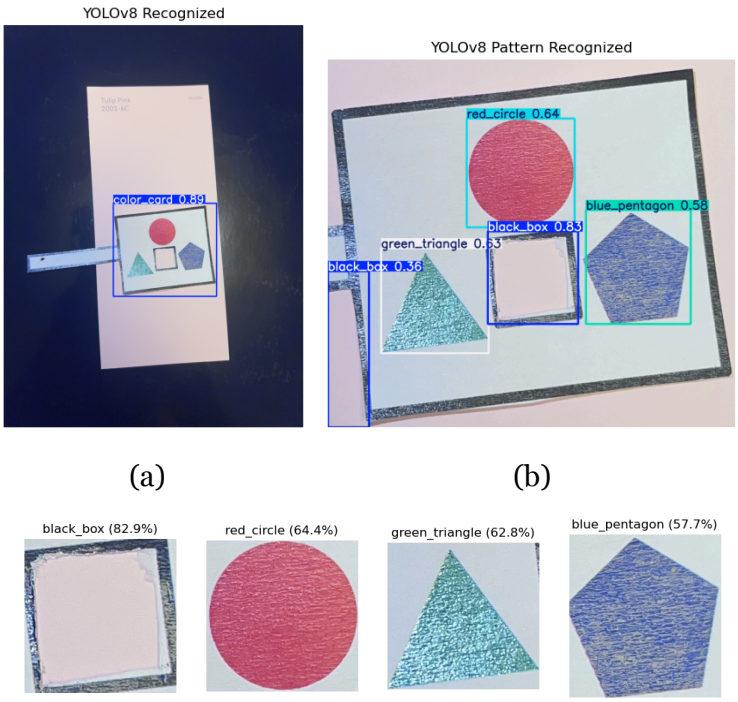


Figure: used YOLOV8 model to (a) detect the card (b) detect the landmarks (c) extract the information from the landmarks

Step 2: Model Training and Calibration

Feature Engineering and Model Selection

Domain Knowledge for Feature Engineering

Before exploring deep learning models, we applied domain knowledge to engineer relevant features. These included:

- **Color Landmarks:** Identifying reference points in the image with known color values.
- **Pixel Intensity Ratios:** Accounting for variations in light reflection across color channels.

- **Object Detection Signals:** Focusing on specific objects in the image whose color needed to be identified and corrected.

Transition to Deep Learning for Feature Extraction

While domain knowledge-driven feature engineering was effective, we recognized its limitations in identifying complex feature relationships. **Deep learning models**, particularly neural networks, eliminated the need for manual feature engineering by learning these relationships automatically. The neural network models could map non-linear relationships between the observed and true colors, significantly enhancing prediction accuracy.

Exploring Multiple Tools and Models

We experimented with multiple models and frameworks to identify the best-performing approach. The models explored included:

1. **Simple Difference Model** – A basic baseline that adjusted colors based on simple arithmetic differences.
2. **Weighted Difference Model** – An improvement that introduced channel-wise correction weights.
3. **Neural Networks (Fully Connected Models)** – Capable of learning non-linear relationships between observed and true RGB values.
4. **Random Forest Regressor** – A robust ensemble method that handled non-linear mappings efficiently.

YOLOV8 Model: Used for **object detection and landmark identification** to focus color calibration on specific regions of interest in the image.

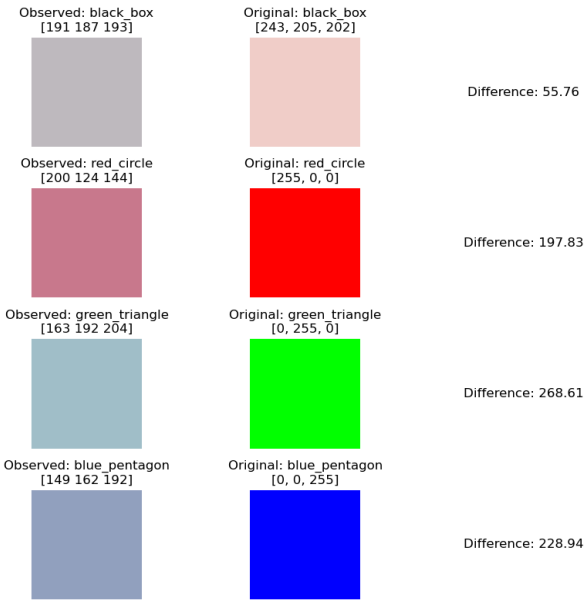


Figure: Color and the difference from observed to the real values after adjustment

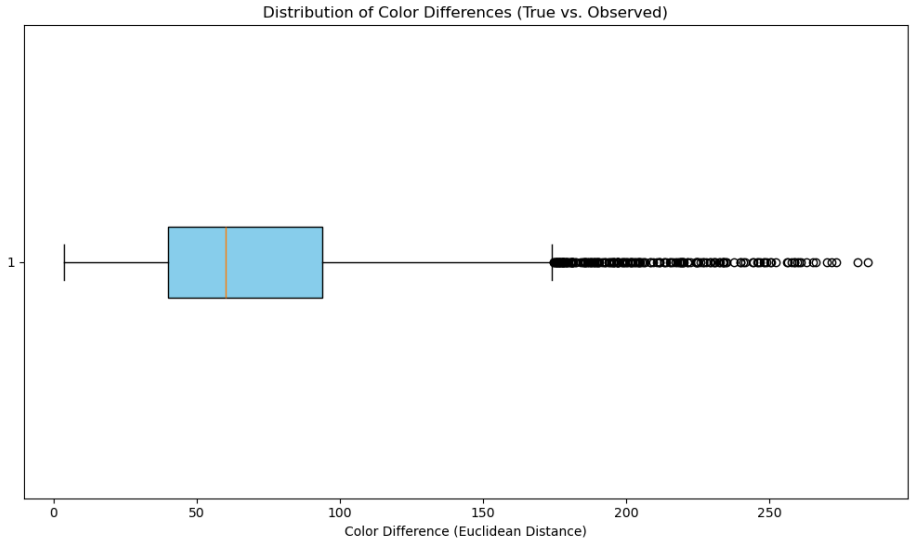


Figure: Distribution of the color differences between the true and observed values

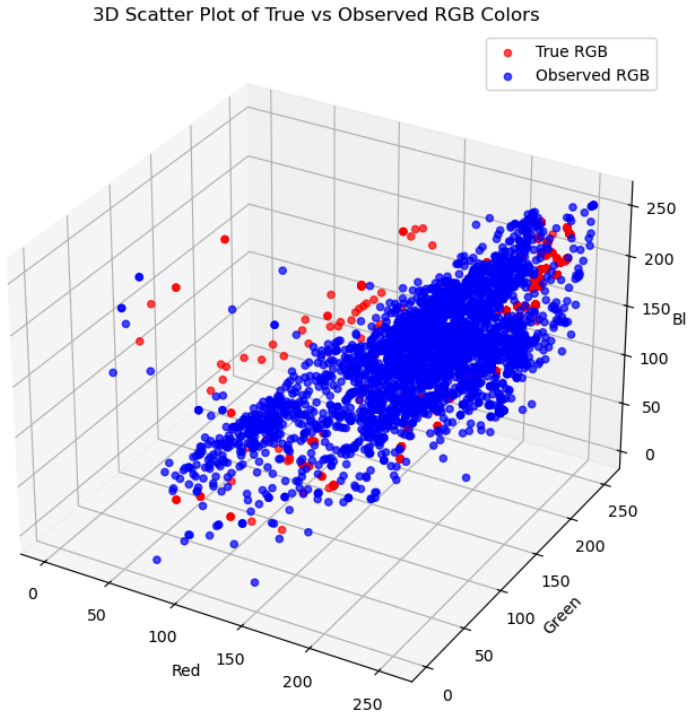


Figure: Sample plot of the true and observed values on the training set between the blue, red and green values in a 3D plot

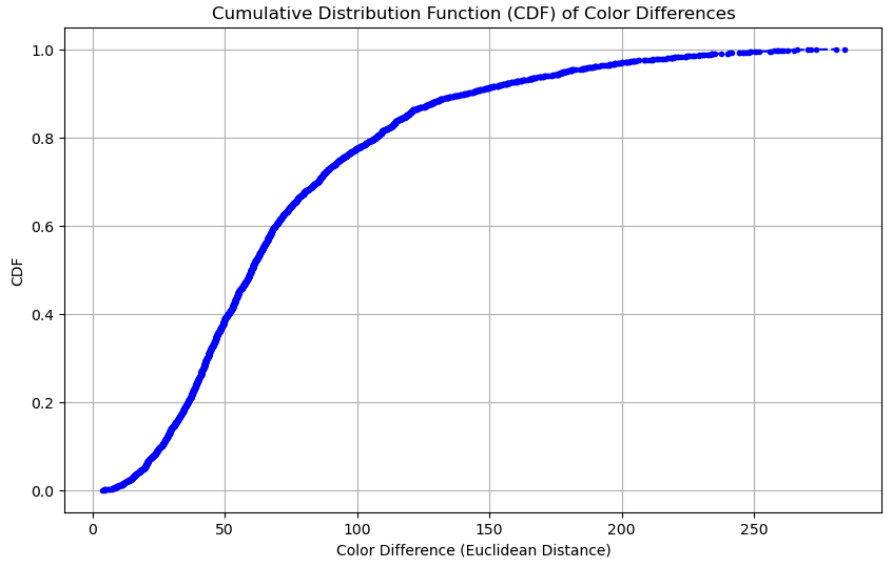


Figure: Cumulative distribution function of the color differences

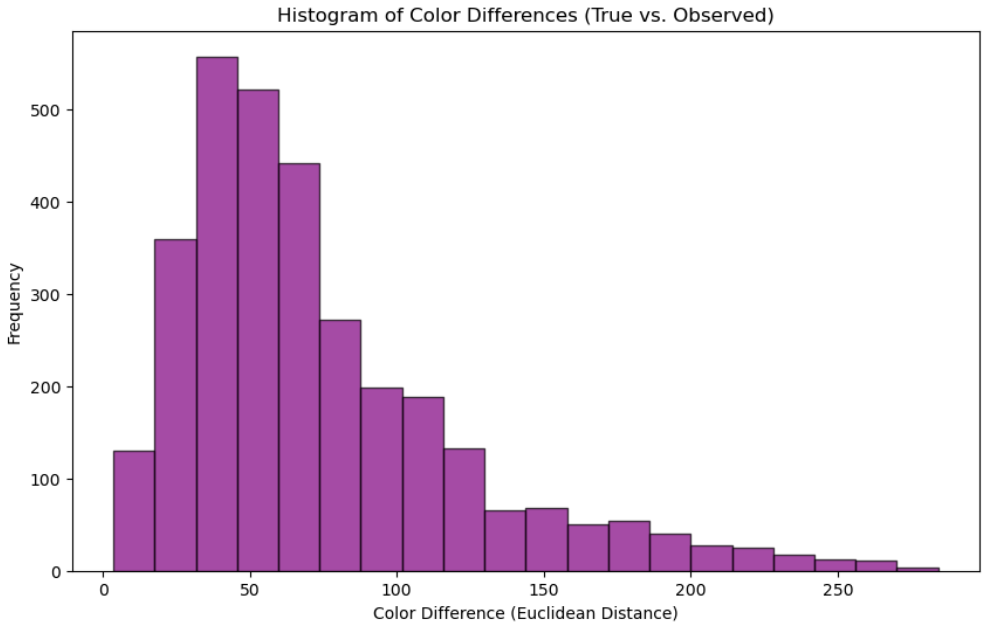


Figure: The color difference, to understand color difference distribution

1. Simple Difference Model

- This model calculated the correction needed by taking the difference between the true and observed RGB values.
- For new observations, it applied the calculated difference as a correction factor to adjust the predicted RGB values.
- Although this model provided a baseline, its effectiveness was limited as it did not account for nonlinear variations in color perception.

2. Weighted Difference Model

- An enhancement of the Simple Difference Model, this method introduced **weights** for each color channel (R, G, B).
- The model learned optimal weights to minimize the mean squared error (MSE) between predicted and true values.
- Weights were adjusted iteratively through gradient-based optimization, ensuring that corrections were more accurate than the simple difference method.

3. Neural Network Model

- A **feedforward neural network** was constructed to predict corrected RGB values.
- **Input Layer:** Received the observed RGB values and any additional correction factors.
- **Hidden Layers:** Applied non-linear transformations to the inputs using activation functions such as ReLU (Rectified Linear Unit).
- **Output Layer:** Predicted the corrected RGB values.
- The neural network was trained using backpropagation and an adaptive optimizer (Adam) to minimize the loss function (Mean Squared Error).
- The model learned complex relationships between observed and true colors, improving the accuracy of predictions.

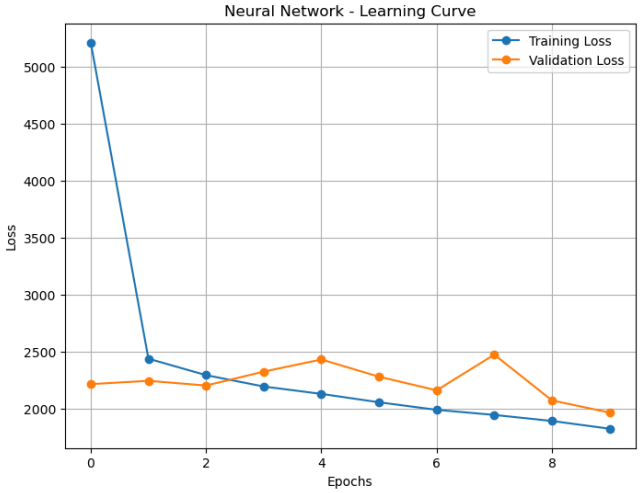


Figure: Learning curve with training and validation loss

4. Random Forest Model

- A **Random Forest Regressor** was used to map observed colors to their true values.
- The model utilized an ensemble of decision trees to learn the best corrections.
- Each decision tree learned different aspects of the dataset, and the final prediction was made by averaging the predictions of all trees.

- Random Forests were particularly useful due to their robustness to noise and ability to capture nonlinear relationships in the data.

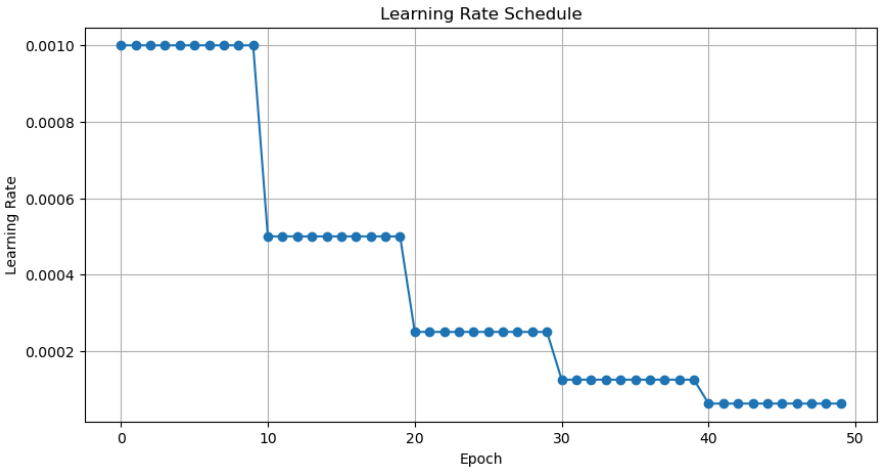
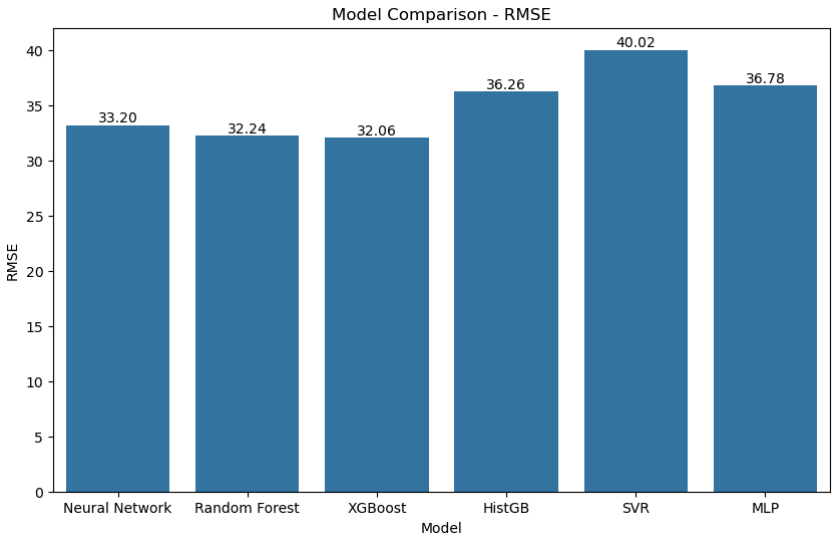


Figure: Used a learning rate scheduler

Step 3: Model Evaluation

After training the models, they were evaluated using a holdout test set. The models' performances were compared using several statistical metrics:

- **Mean Squared Error (MSE):** Measures the average of the squared differences between the predicted and actual RGB values.
- **Root Mean Squared Error (RMSE):** Provides a more interpretable error by taking the square root of MSE.
- **R^2 Score (Coefficient of Determination):** Indicates how well the model's predictions match the actual values, where a value closer to 1.0 signifies a better fit.



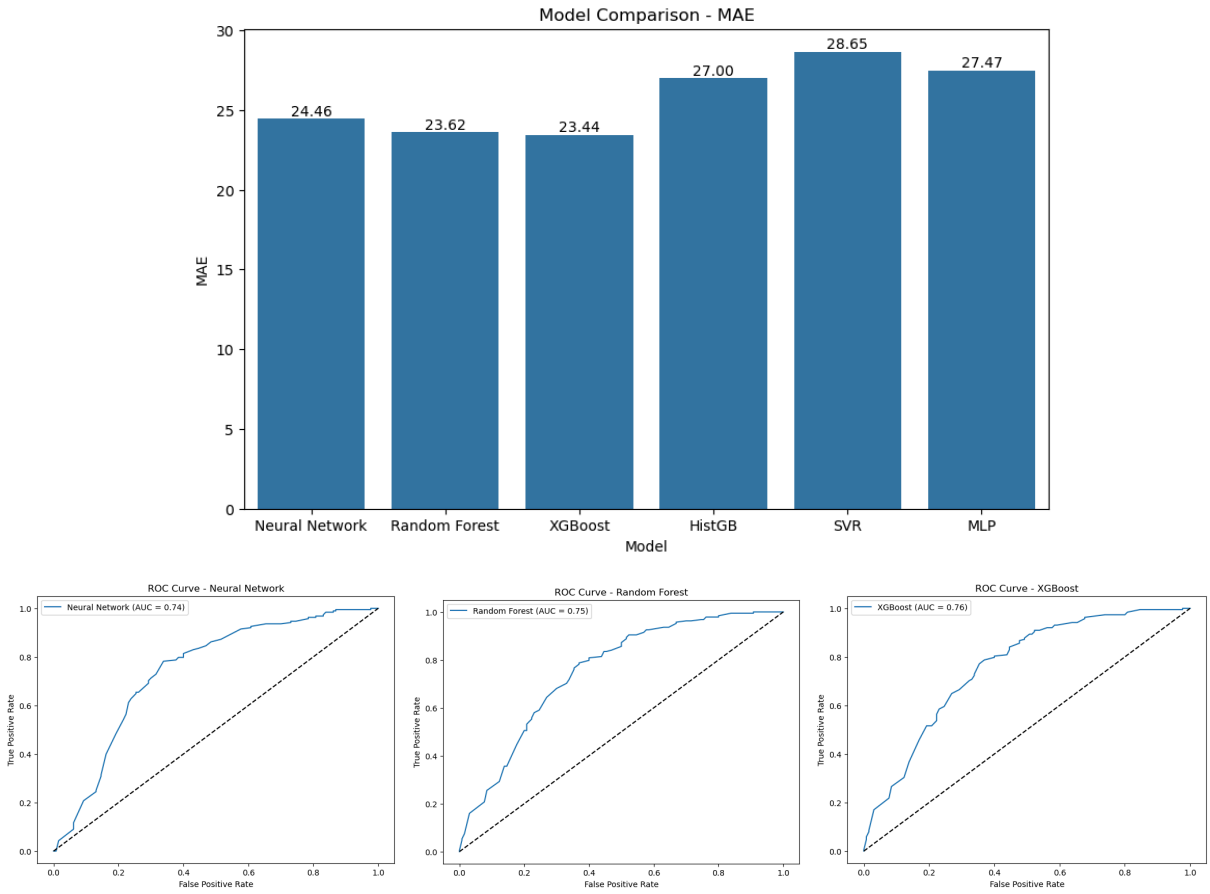


Figure: Models and score comparisons for RMSE, MAE, ROC curve

Data Partitioning Strategy

A **data-centric approach** was adopted, focusing on improving the dataset and input signals to maximize model performance. The dataset was divided into:

- **Training Set (70%)** – For model learning and optimization.
- **Validation Set (15%)** – For hyperparameter tuning and model selection.
- **Test Set (15%)** – For final evaluation to assess the model's generalization on unseen data.

By ensuring that the validation and test sets reflected diverse lighting conditions and angles, we maintained consistency in assessing the models' effectiveness across different scenarios. We also divided based on the values such that the colors remain together and the test and validation sets get different previously unseen colors to be able to more accurately understand the performance.

YOLOv8 for Landmark and Object Detection

The inclusion of a **YOLOv8 model** added a critical dimension to the color calibration process by enabling:

- **Landmark Detection:** Identifying key color regions or objects in the image whose colors needed correction.
- **Object-Specific Calibration:** Allowing focused calibration on target objects rather than applying global corrections to the entire image.
- **Signal Tracking:** Tracking necessary signals and objects to ensure corrections were targeted and effective.

Step 4: Best Model Selection and Validation

4.1 Avoiding Simple Averages

- We avoided the use of **simple averaging techniques** in color correction, recognizing that pixel intensities refract light differently under varying conditions.
- Instead, we focused on learning dynamic correction factors that adapted based on input conditions.

4.2 Performance Evaluation

The performance of all models was compared using:

- **Mean Squared Error (MSE):** To quantify the average squared difference between predicted and actual RGB values.
- **Root Mean Squared Error (RMSE):** To express the error in the same scale as the original data.
- **R^2 Score:** To measure how well the model predictions aligned with the ground truth.

4.3 Best model selection

- Once all models were evaluated, the model with the **highest R^2 score and lowest MSE** was selected as the best-performing model.
- The best model was then validated using a separate dataset to ensure its robustness and generalization to unseen data.
- The thought process behind the selection and evaluation of models was made transparent by displaying confidence scores and correction values for each approach.

4.4 Cross-Validation and Model Iterations

- To ensure robustness and prevent overfitting, we used **K-Fold Cross-Validation**. This allowed us to split the dataset into multiple folds, iteratively training and validating models to assess their consistency across different subsets.
- Several iterations of model training and tuning were conducted, where:
    - Hyperparameters were optimized for neural networks.
    - Tree depth and estimator count were fine-tuned for the Random Forest model.

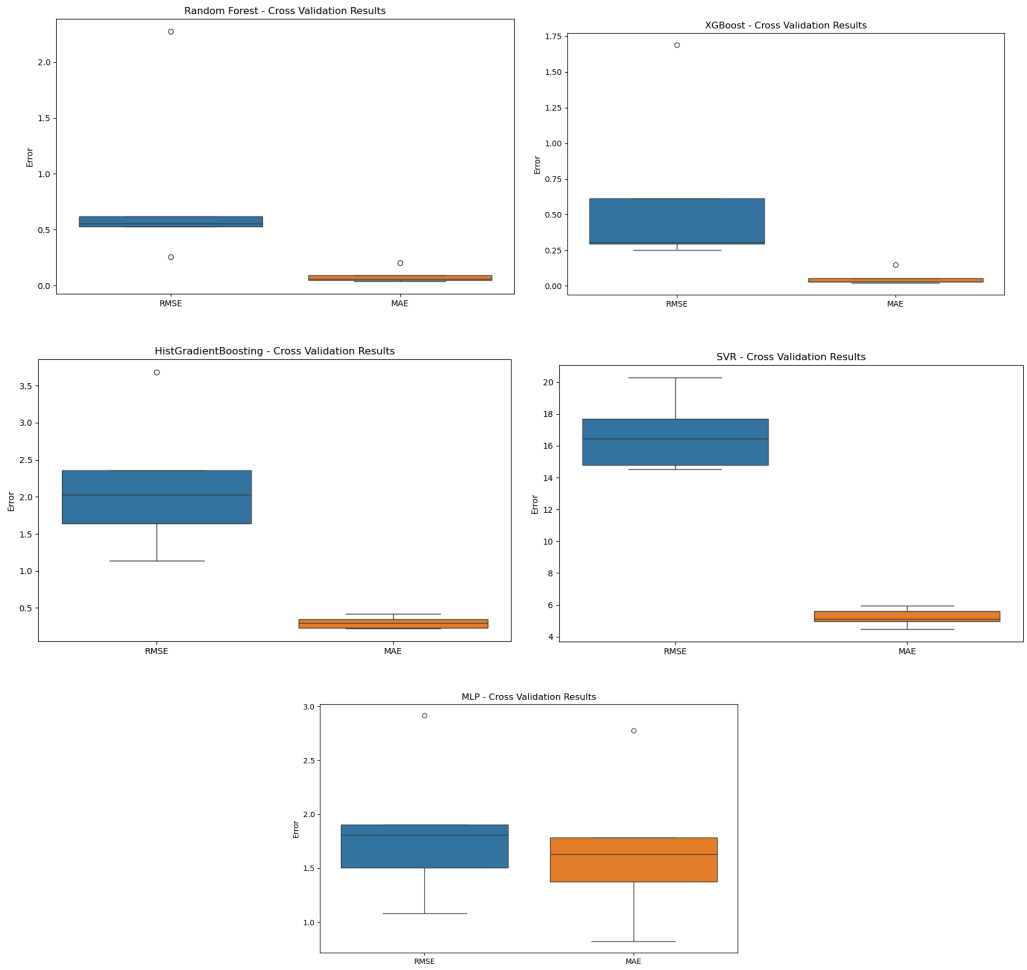


Figure: Cross-Validation comparison between models

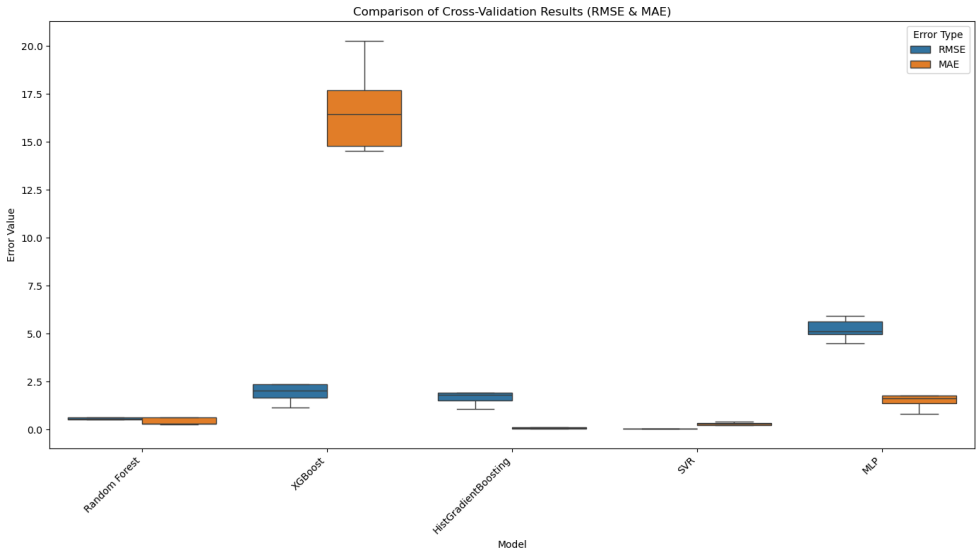


Figure: Overall comparison between models for cross-validation

4.5 Thought Process Behind Model Selection and Refinement

4.5.1 Signal Tracking and Color Detection

- We tracked relevant signals such as **color landmarks** and specific objects whose colors needed identification.
- This signal-based approach allowed for better calibration by focusing on relevant regions rather than applying global corrections.

4.5.2 Neural Networks for Feature Learning

- Neural networks emerged as the best choice due to their ability to:
    - **Automatically Learn Features:** Identifying non-linear relationships that manual feature engineering could not capture.
    - **Handle Variations:** Adapting to different lighting conditions and perspectives more effectively.

Step 5: User Interface and Visualization

- The final model was integrated into a **web-based interface** where users could upload an image, and the system would predict the corrected colors.
- The interface displayed:
    - The **best model prediction** with the corrected color.
    - A **confidence score** representing the accuracy of the prediction.
    - A **thought process panel** that detailed the corrections applied and compared the outputs of all models.
- Users had the option to toggle the thought process visualization to understand how each model contributed to the final prediction.

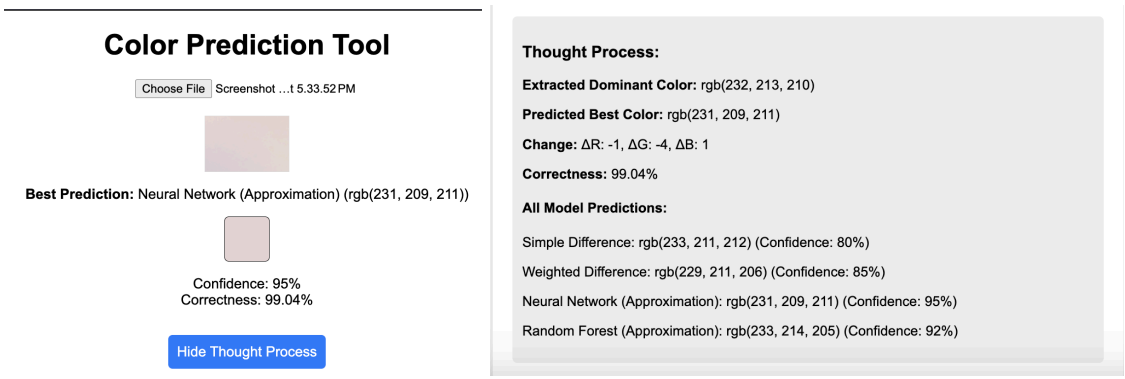


Figure: Color Prediction Tool Website [hosted here]

7. Results and Analysis

Model	MSE	RMSE	R^2 Score
Simple Difference Model	119199	345.25	-100.86
Weighted Difference Model	1697	34.34	-0.34
Neural Network Model	20.05	4.47	0.98
Random Forest Model	18.06	4.25	0.98

Key Observations

- The **Simple Difference Model performed poorly**, indicating that basic arithmetic corrections are not enough.
- The **Weighted Difference Model improved the results**, but still had a low R^2 score.
- The **Neural Network and Random Forest models achieved the best results**, with an R^2 of **~0.98**.
- **Random Forest slightly outperformed the neural network** in MSE but was comparable overall.

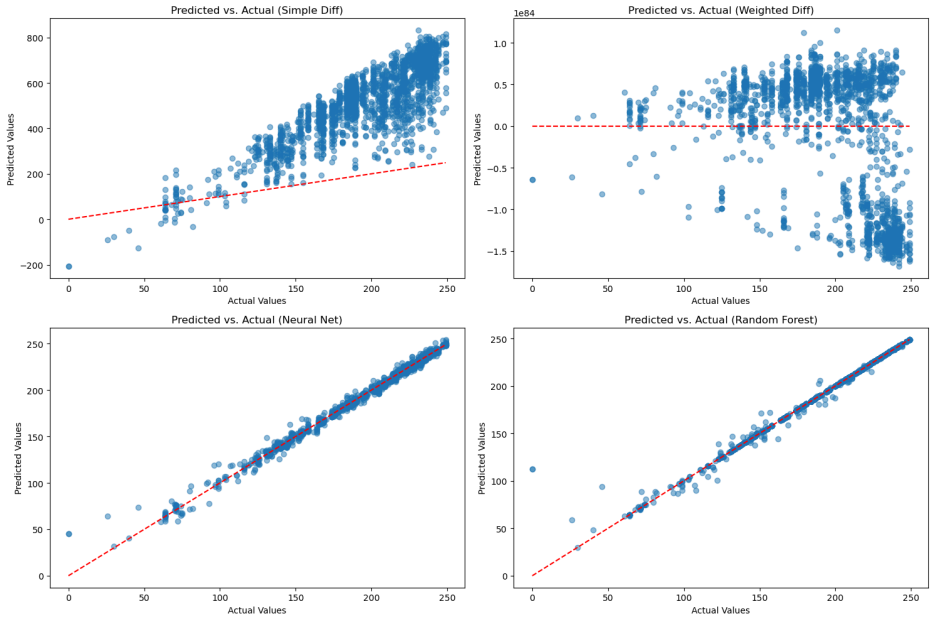


Figure: This shows predicted vs actual values for the different models, where we can see that Neural Network and Random Forest works better

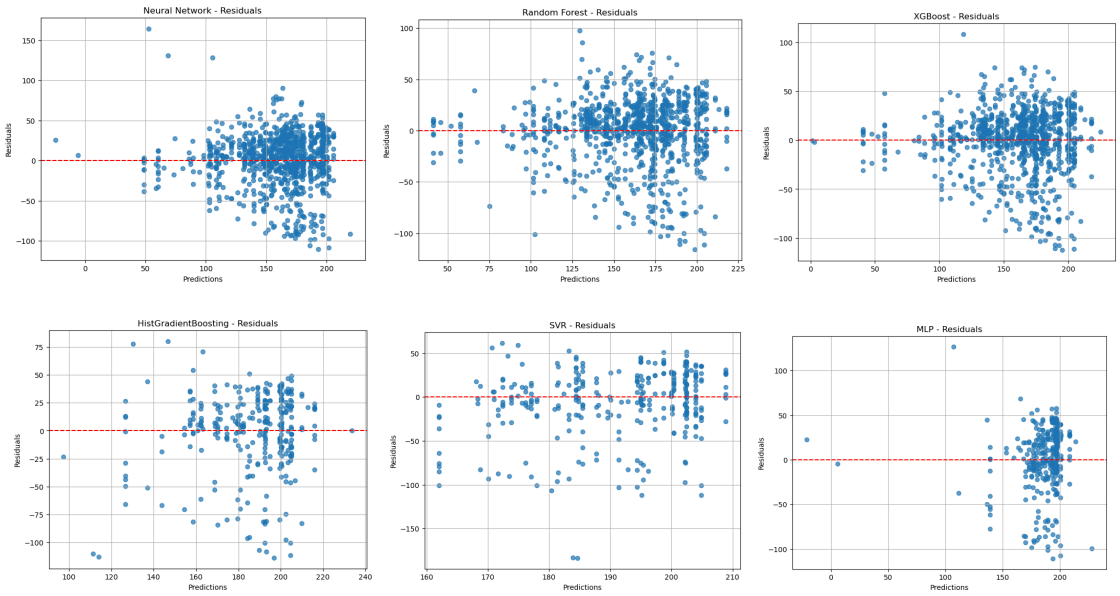


Figure: Model comparison residuals

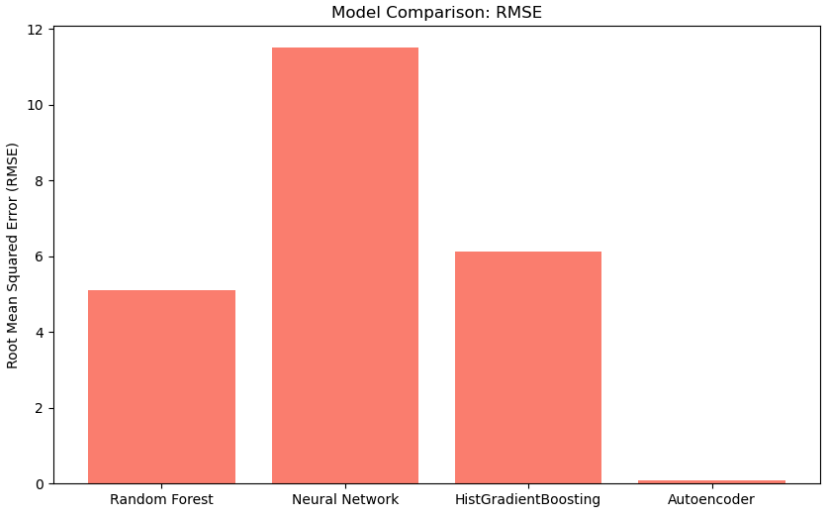


Figure: Comparison of RMSE between different models

8. Conclusion

The implementation journey involved extensive experimentation, data augmentation, and model validation, ultimately culminating in a powerful and reliable color calibration tool. Our data-centric approach ensured robust performance under varying conditions, with YOLO enhancing object detection and landmark identification. Through this project, we successfully implemented and evaluated multiple models for color calibration, with the Neural Network and Random Forest emerging as the most accurate, significantly reducing color prediction errors. Future improvements could include:

- **Hyperparameter tuning** for the Neural Network.
- **Data augmentation** to improve robustness.
- **Integration with real-time color correction software**.

9. References

- Scikit-learn Documentation: https://scikit-learn.org
- TensorFlow Documentation: https://www.tensorflow.org
- YoloV8 Model: https://yolov8.com/
- Valspar color dataset: https://www.valspar.com/en/colors/browse-colors
- BEHRs color dataset: https://www.behr.com/consumer/colorstudio
- OpenCV documentation: https://opencv.org/
- Tools used: Visual Studio Code, GitHub, Google Colab, ChatGPT

- I would like to thank my classmates to help with the data collection process, the insightful discussions and support with the code. Additionally, I would like to thank Prof. Pengyu Hong, for the support and guidance throughout the project and for the incredible opportunity to work on something so interesting.

10. The Code

The code is viewable on GitHub: https://github.com/VedanshiShah7/practical-ml-project-1
The best way to view it is through the webpage linked in the README. Another way is through the jupyter notebook main.ipynb. Furthermore, the code is available through individual files and models. Please view the Readme for further information.