COLOR DETECTION SYSTEM

UML 510 – MACHINE LEARNING

Project Report

BE Third Year (CSE)

Submitted by:

Deeksha Aggarwal (102397012)

Kamal (102397015)

Sub-group: 3CS9

Submitted to: Dr. Anjula Mehto (Assistant Professor)

Dept. of Computer Science and Engineering

Thapar Institute of Engineering and Technology, Patiala

November 2024

ACKNOWLEDGEMENT

Being students utmost important for us to know how to implement our theoretical knowledge in our daily lives. Even for veteran engineers it sometimes becomes difficult to choose the correct resourced while developing the basic website. In such a case it is important that they have a preliminary knowledge about prerequisites which we require.

We take this momentous opportunity to express our heartfelt gratitude, ineptness and regards to highly esteemed guide, Dr. Anjula Mehto for providing us an opportunity to showcase our project. We with full pleasure converge our heartiest thanks to our project guide, Dr. Anjula Mehto for her valuable advice and cooperation without which this project would not have seen the light of day.

CONTENT

PROJECT OVERVIEW

Introduction

Color detection is a process where specific colors in an image are identified and analyzed based on their RGB (Red, Green, Blue) or other color space values. It involves extracting pixel data from an image and processing it to classify or predict the color using machine learning techniques like K-Nearest Neighbors (KNN) or color space transformations (e.g., converting RGB to HSV). This technique is widely used in applications such as object tracking, image segmentation, and even in user interfaces where selecting, identifying, or classifying color is essential.

In this project, we will be accomplishing the task using computer vision, image processing and KNN algorithm on supervised data having RGB values along with labels of the colors (i.e. supervised machine learning).

KNN is a non-parametric, instance-based and *supervised* machine learning algorithm used for *classification* tasks. The KNN algorithm stores all available data and classifies new data points based on a similarity measure (Euclidean distance).

For color detection, when a new RGB value is input (e.g., the average RGB of a selected image area), KNN finds the closest RGB values in the training data and assigns the label of the majority color class among the k nearest neighbors.

Key functionalities and libraries used

- **cv2(OpenCV)**: Used for image processing and allows pixel-level access to an image in BGR format (Blue-Green-Red).
- **PIL (Pillow):** Converts downloaded images into formats compatible with processing libraries like OpenCV.
- **NumPy**: Handles numerical operations and manipulates image data as arrays.
- **Pandas**: For managing and preprocessing the color dataset (colors_new.csv).
- **Requests**: Handles downloading files from URLs. Here, it fetches the image from a specified URL.
- **sklearn(scikit-learn):** Provides tools for machine learning tasks:
    - train_test_split: Splits data into training and testing subsets.
    - MinMaxScaler: Scales RGB features to the [0, 1] range.
    - KNeighborsClassifier: Implements the KNN algorithm.
- **io.BytesIO:** Handles binary streams of data (used here for converting downloaded image content into an in-memory file).
- **matplotlib.pyplot:** Displays the image to the user for interactive color selection.
- **ipywidgets.interact:** Allows user interaction for specifying coordinates (x, y) in the image.

PROBLEM STATEMENT

In the modern digital age, the ability to identify colors accurately plays a crucial role in fields like graphic design, image processing, and accessibility solutions. However, interpreting color data from an image can be complex due to varying lighting conditions, color spaces, and human perception. This project seeks to address the challenge of automatically detecting and identifying color names from specific pixels in digital images.

Objective

Develop a machine learning-based system that uses K-Nearest Neighbors (KNN) to detect and classify colors from the RGB values of a pixel within an image. The system should allow users to upload or provide a URL of an image, visually select any pixel, and receive the predicted color name.

Scope of the project

- **Dataset Preparation**:
  - Build a labeled dataset of RGB values and corresponding color names.
- **Machine Learning Model**:
  - Implement and train a KNN classifier to learn the relationship between RGB values and color names.
- **Image Processing**:
  - Enable the system to load and preprocess images from local storage or via URL.
- **Interactive Visualization**:
  - Provide a user interface for displaying the image and allowing pixel selection for color detection.
- **Color Prediction**:
  - Use the trained KNN model to predict the color name based on the RGB values of the selected pixel.

DATASET OVERVIEW

Dataset Reference

https://www.kaggle.com/datasets/adityabhndari/color-detection-data-set

Structure of Dataset

Number of Entries: 865 rows

Number of Features: 4 columns

Column Name	Data Type	Description
Color	Object	Name of the color corresponding to the RGB values.
R	Integer	Red component of the color (0-255).
G	Integer	Green component of the color (0-255).

B	Integer	Blue component of the color (0-255).

Sample

	A	B	C	D
1	Color	R	G	B
2	Air Force Blue (Raf)	93	138	168
3	Air Force Blue (Usaf)	0	48	143
4	Air Superiority Blue	114	160	193
5	Alabama Crimson	163	38	56
6	Alice Blue	240	248	255
7	Alizarin Crimson	227	38	54
8	Alloy Orange	196	98	16
9	Almond	239	222	205
10	Amaranth	229	43	80

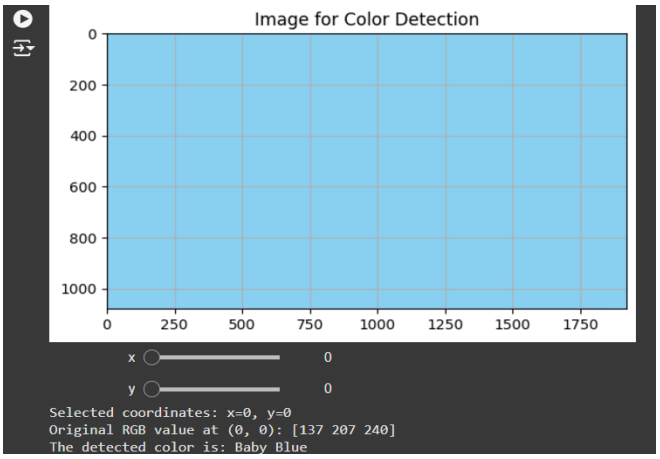
PROJECT WORKFLOW

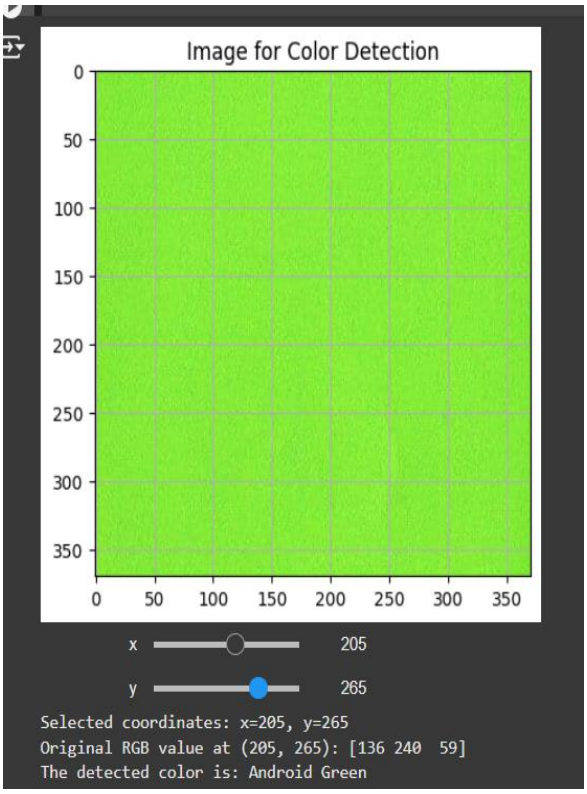
- Data Preparation:
  - Read the Dataset colors_new.csv with RGB values and corresponding color names using pandas.read_csv().
  - Create a Data Frame using Pandas.DataFrame().
  - Separate the dataset into features (R,G,B) and target (color).
- Data Preprocessing:
  - Perform Normalization on features (R,G,B) to map these in 0 to 1.
- Data Splitting:
  - Split the dataset into training, testing, and validation sets using train_test_split. Here, Training Set: 70% of the data and Testing Set: 30% of the data.
- Model Training:
  - Train a K-Nearest Neighbors (KNN) classifier:
  - Set a value for k (number of neighbors). Here, 5.
  - Fit the classifier on the scaled RGB training data.
- Model Testing:
  - Run the model to predict target values for Test set.
- Image Loading and Preprocessing
  - Accept an image input (URL or local file)
  - Download and open the image using the requests, io.BytesIO and Pillow libraries.
  - Convert the image to the OpenCV BGR format and return as an array (if error, return NONE).
  - Display the image for visualization and user interaction using matplotlib (in RGB), with an interactive grid.
- User Interaction for Pixel Selection:
  - Allow users to select specific (x, y) coordinates on the image using ipywidgets.interact.
  - Extract the RGB values of the selected pixel.
- Color Prediction:
  - Convert the extracted RGB to BGR.

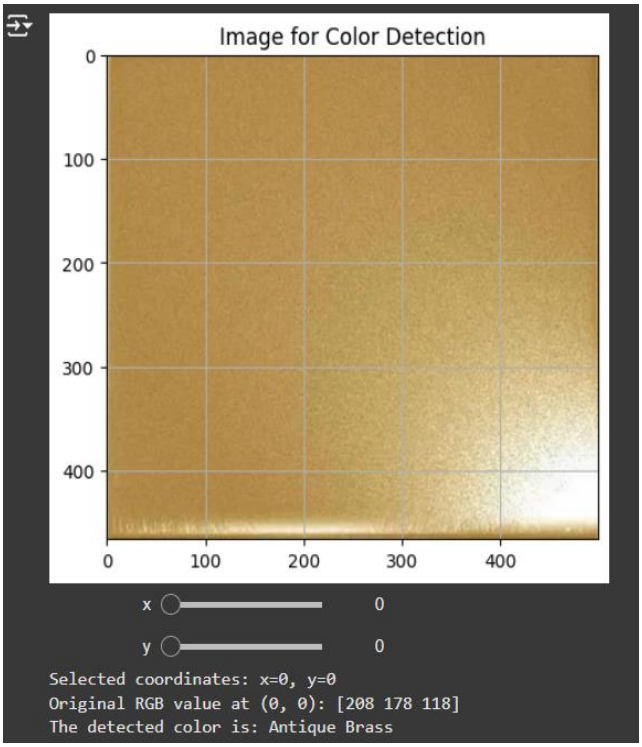
- Normalize the extracted RGB values.
- Use the trained KNN model to predict the color name corresponding to the selected pixel.

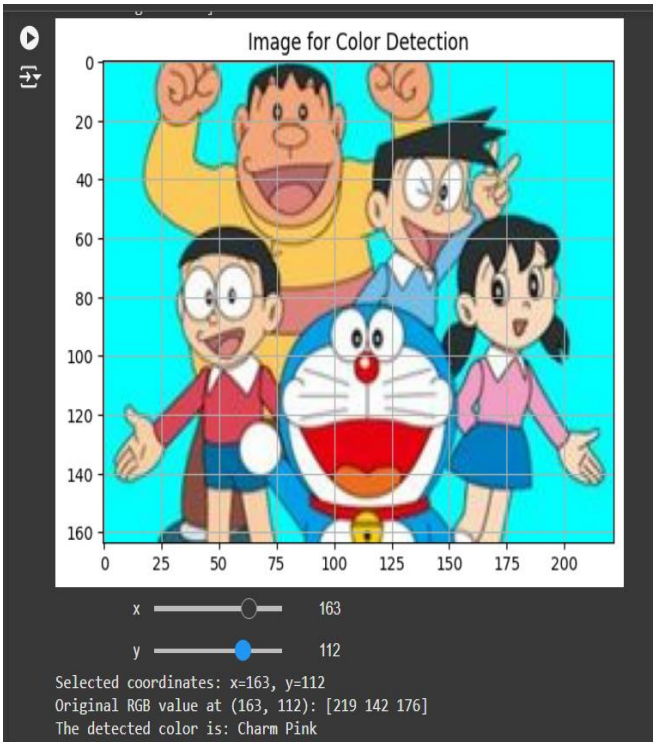
- Output Results:
  - Display the predicted color name and the RGB values of the selected pixel to the user.
- Output Visualization:
  - Show the image with an interactive grid for easier pixel selection.

RESULTS










```
if __name__ == "__main__":
    main()
```

```
Error: URL does not point to an image.
Error loading image. Please check the URL.
```

FUTURE SCOPE

At this point, the project is giving correct outputs for varying shades of color.

This project has significant potential for further development and practical applications. Future work can focus on expanding the dataset to include a wider range of colors and scenarios, such as variations in lighting, texture, and saturation, to improve the model's robustness and accuracy. Additionally, implementing more advanced machine learning models like Random Forests could enhance prediction accuracy for complex datasets. Exploring deep learning techniques, such as Convolutional Neural Networks (CNNs), would enable the system to analyze entire image regions rather than individual pixel values, making it more effective for real-world applications.

Integration with mobile or web-based platforms can make the system more user-friendly and accessible for tasks like color matching in design, interior decoration, or image editing. Another area of improvement includes optimizing the KNN model parameters, such as the number of neighbors (`k`) and the choice of distance metrics, through rigorous cross-validation. Furthermore, addressing challenges like dataset imbalance and color similarity using techniques like oversampling or clustering can make the model more robust. Expanding the system to detect color names in multiple languages or formats would also increase its versatility and usability across different domains.

CONCUSION

The project aimed to build a K-Nearest Neighbors (KNN) classification model to predict color names based on RGB values. The model was trained using a labeled dataset of RGB values corresponding to various color names and tested on unseen data. Additionally, the project incorporated an interactive feature to detect color names from images using pixel values.

Thus, the project, as shown above, performs satisfactorily in real time by predicting correct color labels and interacts with user in a convenient way as intended.