SmartClass A.I.ssistant Project Part 1 Report

Methodology for Data Collection, Cleaning, Labeling & Preliminary Analysis

COMP 472

Team OB_18

Alma Almidany (40197854) - Data Specialist

Carmen Derderian (40244084) - Training Specialist

Hawraa Al-Adilee (40246450) - Evaluation Specialist

Repository: https://github.com/7awraaa/COMP472_AI

Professor: Dr. René Witte

"*We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"*

A.A. - C.D - H.A.A.

TABLE OF CONTENTS

DATASET

Overview of Existing Dataset

- Total number of images: 2000 images. (TO BE MODIFIED AS WE ADD OUR OWN PICTURES)
- Number per class: 500 images per class, including 400 evaluation images and 100 test images.
- Special characteristics: The data set consists primarily of color images with several black and white images. Most images are front facing shots of faces from diverse backgrounds and age ranges. The diversity is important in variability in facial features across demographics.

Justification for dataset choices

The chosen dataset was taken from the Kaggle website. The folder from which they were sourced contains a training and a testing set, each containing subfolders labeled with numbers. Each numbered subfolder contains photos of a specific emotion. We made sure to select pictures for the required three emotions as well as the additional emotion: happy. To ensure randomness and unbiased datasets, a program was used to select the images (the script can be found on GitHub).

A characteristic which distinguishes this dataset is the extensive categorization of emotions. To ensure the quality and diversity of our dataset, it is advantageous that each subfolder contains an important number of images. The training dataset contains at least 3000 images per subfolder. This large quantity allows a more effective randomization. Each of the emotions folders contains faces of various ages and backgrounds. This diversity ensures that the program will be able to recognize facial expressions regardless of details such as age, background, accessories, skin tone, and facial features. Some accessories, such as sunglasses, would hide the majority of the face,

and therefore the program would not be able to detect the facial expression accurately. To verify that most of the pictures conform to the instructions and are valid for facial detection, we reviewed the generated folders.

The datasets are directly relevant to the project as they provide essential data required for the evaluation and testing models. The categorization of emotions facilitates the creation of precise and accurate models, speeding up the data-cleaning process compared to using a dataset where random pictures of mixed expressions are stored.

Despite its advantages, the chosen dataset may present challenges in the future as the variety of pictures comes with a variety of specifications (brightness, clarity, filters, angles, etc.). We ensured equal representation of each emotion as we randomly selected 500 pictures for each. As the process is entirely randomized, variations in image quality and lighting are unavoidable. These variations can potentially impact model performance.

Dataset Details

Dataset Component	**Source**	**Licensing Type**	**Description**
Training Set	Kaggle AffectNet Dataset - train/0 - train/3 - train/6	Unknown (as per the website)	Subfolders labeled with numbers, each representing a different emotion
Testing Set	Kaggle AffectNet Dataset - train/0 - train/3 - train/6	Unknown (as per the website)	Subfolder labeled with numbers, each representing a different emotion

DATA CLEANING

Techniques applied for standardizing the dataset

Resizing:

- Goal: To standardize the dimensions of all the images in both the training and testing sets
- Method: We resized each image to (256X256) pixels using a python code (shown in the github repository)
- Implementation: We used Python Pillow library to resize the images and save them in (JPG) format in order to ensure efficiency, JPEG format provides sufficient image quality for machine learning and ensures uniformity and compatibility with existing datasets.[1]

Light Augmentation:

- Goal: increase clarity of dataset
- Method: manually alter some of the images in the dataset by slightly rotating the images and adjusting the brightness.
- Implementation: From Python Pillow library we used ImageEnhance to enhance brightness
- Rotation: We manually rotate tilted images between -15 to 15 degrees to achieve uniformity.
- Brightness: We adjusted the brightness of some of the photos for clarity.

Challenges Encountered and Addressed

Various Image Sizes and Resolutions:

- Challenge: The original dataset contained images of different sizes and resolutions.
- Solution: This was addressed by resizing all images to a uniform dimension of 256x256 pixels using the Pillow library. This ensures that all images are consistent in size.

Losing image quality when resizing the images[1]:

- Challenge: losing quality could potentially affect the quality of machine learning.
- Solution: This was solved by using Image.Resampling.LANCZOS from the Python Pillow library to maintain quality.

Example Image

Original image:

Size: 244x244

After Cleaning:

Size: 256x256

The image was slightly rotated to standardize the training images, and the brightness was slightly adjusted to simulate different lighting conditions. The resolution was set to 256x256 to make the dataset uniform. Applying this to all the images in the dataset allows for more accurate machine learning.

DATA LABELING

Methods and Tools for Labeling

Labeling images is necessary in preparing the dataset for the machine learning process.

Labeling Steps:

First, The cleaned images were organized into separate directories based on their class. We had the following directories:

- 'happy_faces': ' COMP472_AI/data/cleaned/happy_faces',
- 'neutral_faces': ' COMP472_AI/data/cleaned/neutral_faces'
- 'focused_faces': ' COMP472_AI/data/cleaned/focused_faces'
- 'angry_faces': ' COMP472_AI/data/cleaned/angry_faces'

Where COMP472_AI is the github repository. The data was acquired from a pre-existing dataset, so it was sorted into folders based on the classes. Some classifications were inaccurate and some images showed more than one emotion or belonged to more than one class, which required manually verifying the labels on the images for accuracy and consistency. If the photos are ambiguous or show more than one expression, the image is removed from the dataset.

To structure the labeled dataset, we created corresponding directories for each class where the labeled images would be stored:

- 'happy_faces': ' COMP472_AI/data/labeled/happy_faces',
- 'neutral_faces': ' COMP472_AI/data/labeled/neutral_faces',
- 'focused_faces': ' COMP472_AI/data/labeled/focused_faces',
- 'angry_faces': ' COMP472_AI/data/labeled/angry_faces'

We developed a python script for the labeling process that iterates through the input directory and stores the images in the output directory and creates a JSON file to store each image's label.

For certain expressions, such as the engaged/focused class, no predefined set was available. To address this, we had to manually map a subset of the neutral class to the focused/engaged class to ensure proper labeling.

Quality Assurance

To ensure accuracy of our labels, each team member randomly verified 15-20 images from the data labeled by the other members. In the case of disagreement, the image was replaced with another one. [2]

Accurate labels are necessary to train an accurate and robust machine learning program. The use of the proper methods and tools can help lay a proper foundation for the process and allow the model to make precise predictions.

DATASET VISUALIZATION

Figure 1. *Class Distribution* [4]. This figure shows the number of images in each class (angry faces, neutral faces, focused faces, and happy faces) within the dataset, highlighting any overrepresented or underrepresented classes**.**

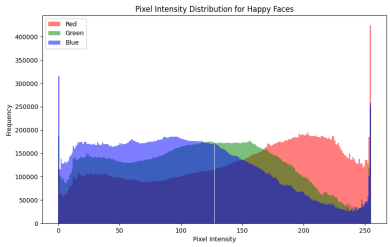

Figure 2. *Pixel Intensity Distribution for Happy Faces* [5][6][11][12][13][14][15]. This figure shows the aggregated pixel intensity distribution for 503 images of happy faces, illustrating the distribution of red, green, and blue pixel intensities.

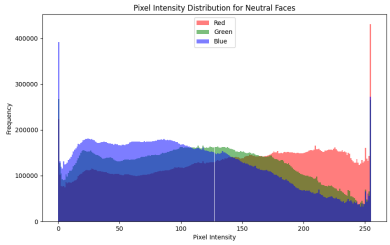

Figure 3. *Pixel Intensity Distribution for Neutral Faces* [5][6][11][12][13][14][15]. This figure shows the aggregated pixel intensity distribution for 503 images of neutral faces, illustrating the distribution of red, green, and blue pixel intensities.

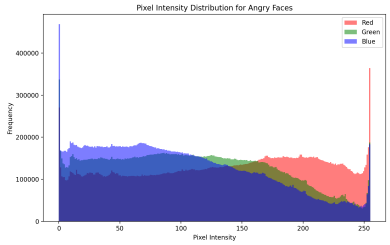

Figure 4. *Pixel Intensity Distribution for Angry Faces* [5][6][11][12][13][14][15]. This figure shows the aggregated pixel intensity distribution for 503 images of angry faces, illustrating the distribution of red, green, and blue pixel intensities.

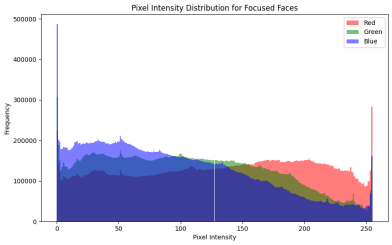

Figure 5. *Pixel Intensity Distribution for Focused Faces* [5][6][11][12][13][14][15]. This figure shows the aggregated pixel intensity distribution for 503 images of focused faces, illustrating the distribution of red, green, and blue pixel intensities.

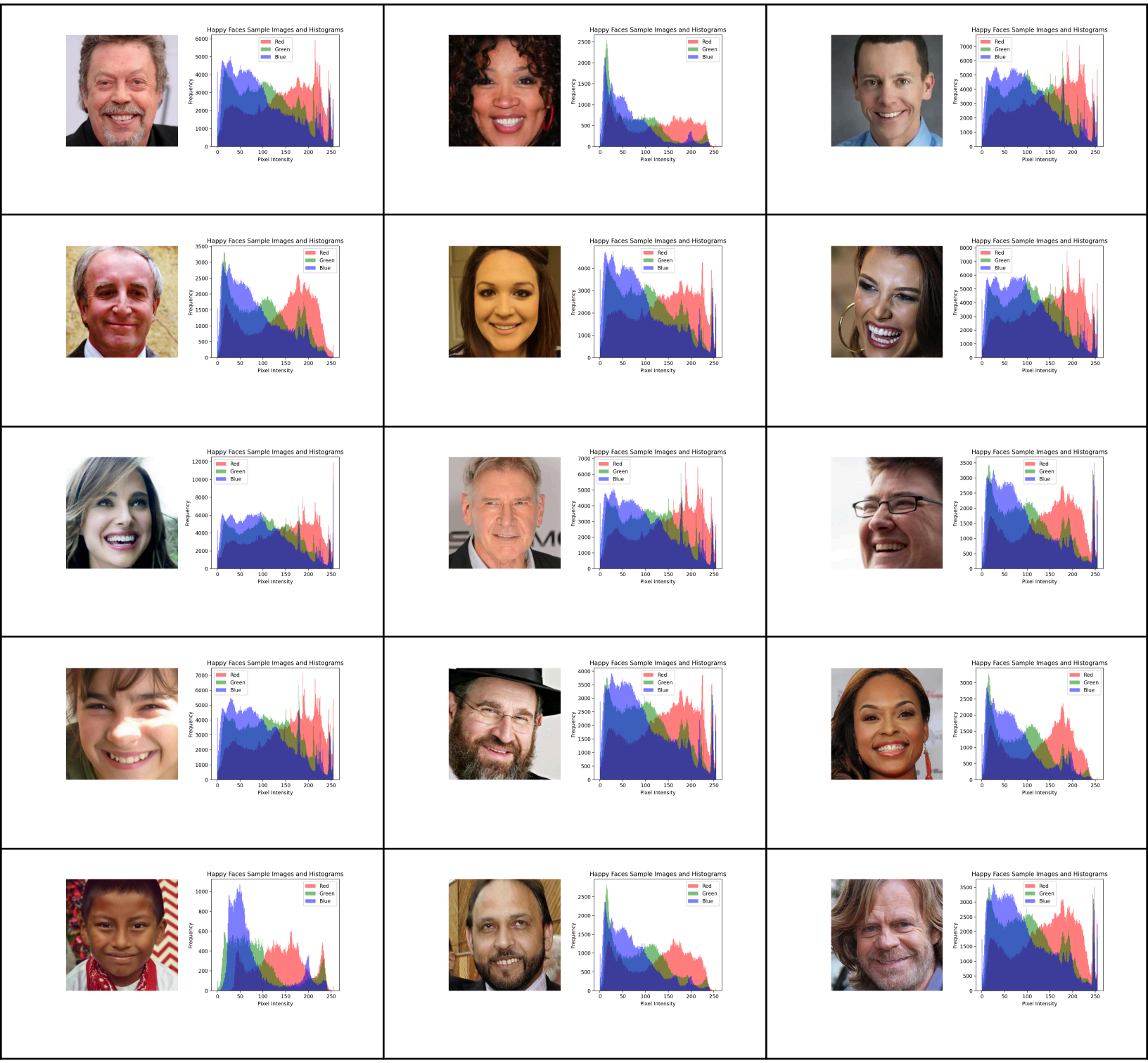

Figure 6. *Happy Faces Sample Images and Histograms* [5][6][7][11][12][13][14][15]. This figure shows a 5x3 grid of 15 randomly selected images of happy faces, each displayed alongside its corresponding pixel intensity histogram, illustrating the distribution of red, green, and blue pixel intensities.

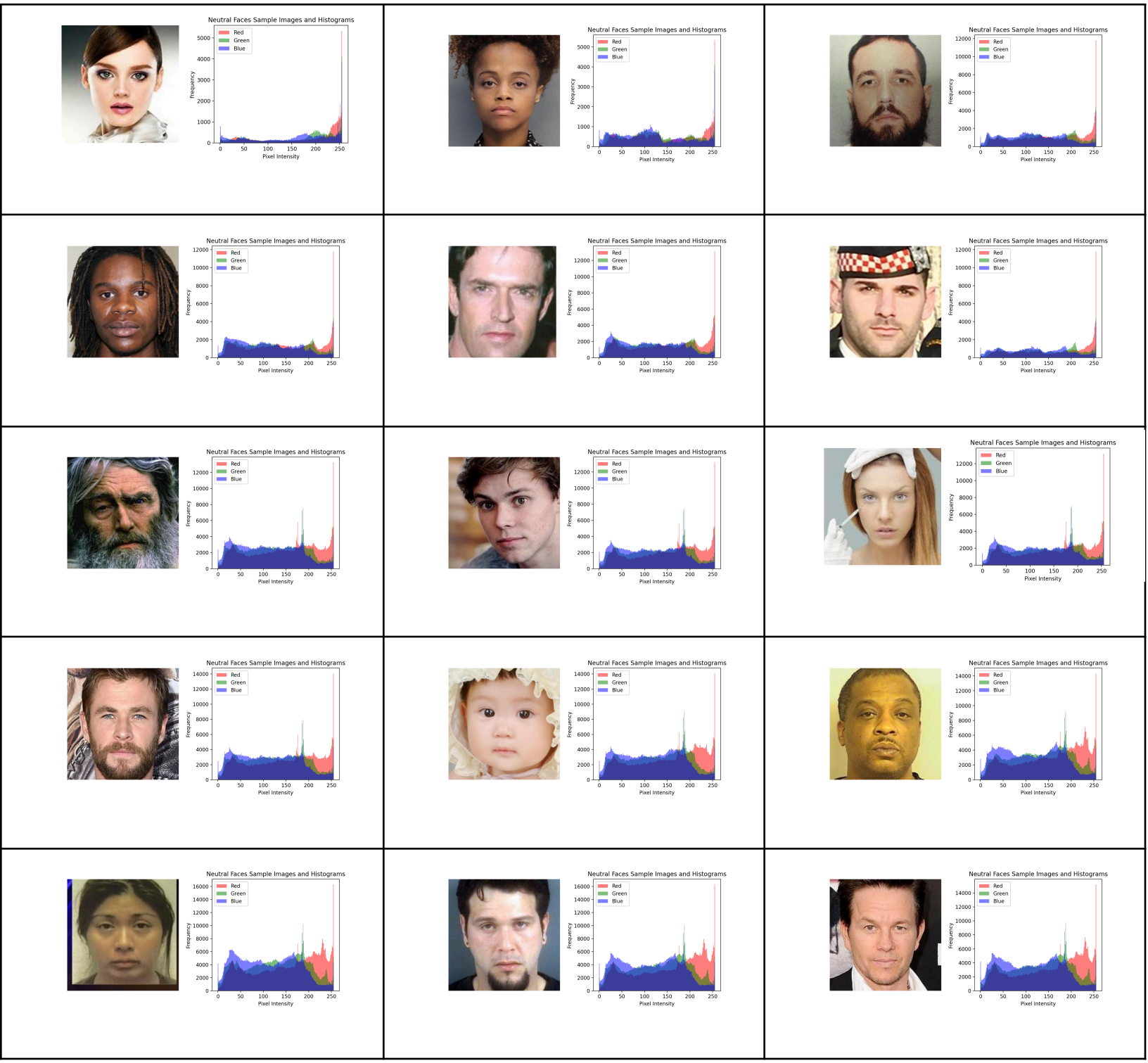

Figure 7. *Neutral Faces Sample Images and Histograms* [5][6][7][11][12][13][14][15]. This figure shows a 5x3 grid of 15 randomly selected images of neutral faces, each displayed alongside its corresponding pixel intensity histogram, illustrating the distribution of red, green, and blue pixel intensities.

Figure 8. *Focused Faces Sample Images and Histograms* [5][6][7][11][12][13][14][15]. This figure shows a 5x3 grid of 15 randomly selected images of focused faces, each displayed alongside its corresponding pixel intensity histogram, illustrating the distribution of red, green, and blue pixel intensities.

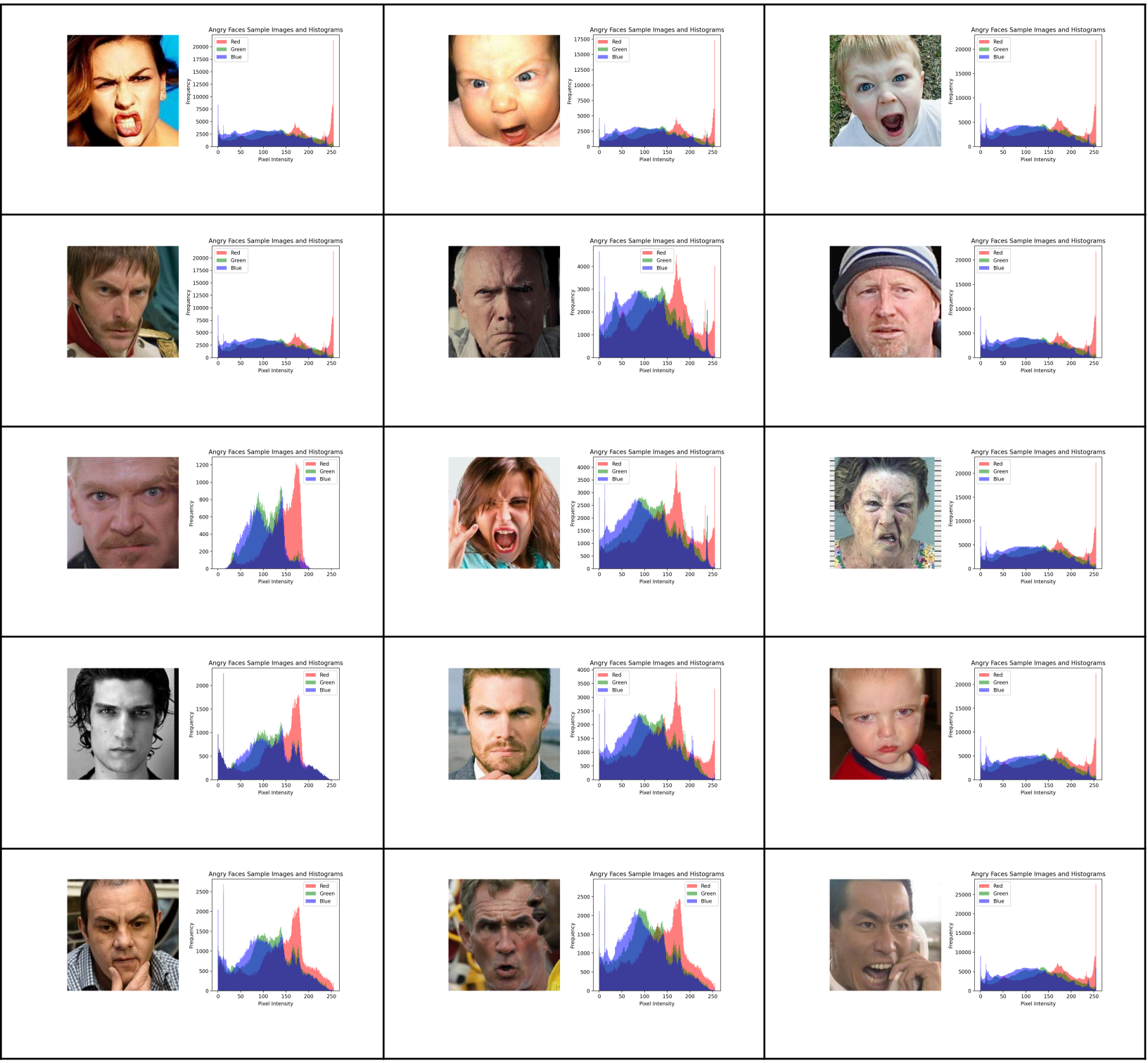

Figure 9. *Angry Faces Sample Images and Histograms* [5][6][7][11][12][13][14][15]. This figure shows a 5x3 grid of 15 randomly selected images of angry faces, each displayed alongside its corresponding pixel intensity histogram, illustrating the distribution of red, green, and blue pixel intensities.

REFERENCES

1. OpenAI, "ChatGPT: Chat Generative Pre-trained Transformer," OpenAI, San Francisco, CA, 2024. [Online]. Available: https://chat.openai.com/ . [Accessed: May 25, 2024].
    - Prompt: “Why would I use JPEG image format”
    - Prompt: “How not to lose quality when resizing and augmenting images in a dataset”

2. OpenAI, "ChatGPT: Chat Generative Pre-trained Transformer," OpenAI, San Francisco, CA, 2024. [Online]. Available: https://chat.openai.com/ . [Accessed: May 27, 2024].
    - Prompt: “How to ensure accuracy of labeled images”

3. OpenAI, "ChatGPT: Chat Generative Pre-trained Transformer," OpenAI, San Francisco, CA, 2024. [Online]. Available: https://chat.openai.com/ . [Accessed: May 28, 2024].
    - Prompt: “How to store the remaining pictures in data/evaluation?”
    - Response:

      “remaining_images = [f for f in os.listdir(input_dir) if f not in selected_images] for image in remaining_images: shutil.move(os.path.join(input_dir, image), os.path.join(evaluation_dir, image))”

4. GeeksforGeeks, “Matplotlib tutorial,” GeeksforGeeks, https://www.geeksforgeeks.org/matplotlib-tutorial/ [Accessed May 27, 2024].

5. GeeksforGeeks, “OpenCV python program to analyze an image using histogram,” GeeksforGeeks, https://www.geeksforgeeks.org/opencv-python-program-analyze-image-using-histogram/ [Accessed May 27, 2024].

6. OpenAI, "ChatGPT: Chat Generative Pre-trained Transformer," OpenAI, San Francisco, CA, 2024. [Online]. Available: https://chat.openai.com/ [Accessed: May 27, 2024].

   OpenAI provided assistance in understanding the concepts for accurate application in the project.

7. “python - how to get average pixel intensities along a line of an image and plot them on a graph?,” *Stack Overflow*. https://stackoverflow.com/questions/59297798/python-how-to-get-average-pixel-intensities-along-a-line-of-an-image-and-plot [Accessed: May 27, 2024].

8. GeeksforGeeks, "How to use os with python list comprehension," GeeksforGeeks, https://www.geeksforgeeks.org/how-to-use-os-with-python-list-comprehension/[Accessed: May 27, 2024].

9. Pynative, "Python Copy Files and Directories," Pynative, https://pynative.com/python-copy-files-and-directories/#:~:text=Suppose%20you%20want%20to%20copy,using%20the%20copy()%20function [Accessed: May 27, 2024].

10. Educative, "How to Enhance Image in Python Pillow," Educative, https://www.educative.io/answers/how-to-enhance-image-in-python-pillow [Accessed: May 27, 2024].

11. Analytics Vidhya, "Advanced OpenCV: BGR Pixel Intensity Plots," Analytics Vidhya, https://www.analyticsvidhya.com/blog/2022/05/advanced-opencv-bgr-pixel-intensity-plots/ [Accessed: May 27, 2024].

12. GeeksforGeeks, "Python | Flatten a 2d numpy array into 1d array," GeeksforGeeks, https://www.geeksforgeeks.org/python-flatten-a-2d-numpy-array-into-1d-array/[Accessed : May 27, 2024].

13. Educative, "Splitting RGB Channels in Python," Educative, https://www.educative.io/answers/splitting-rgb-channels-in-python [Accessed: May 27, 2024].

14. se7en se7en.dev, "Understanding Image Histograms with OpenCV," se7entyse7en.dev, https://se7entyse7en.dev/posts/understanding-image-histograms-with-opencv/ [Accessed: May 28, 2024].

15. YouTube, "Understanding Image Histograms with OpenCV," YouTube, https://youtu.be/kSqxn6zGE0c?si=3rEk9othGLyyzKV3 [Accessed: May 28, 2024].