National University of Computer and Emerging Sciences

Data Science (7-B)
Assignment 3

SustainAI - Automated Recycling Classification

Submitted to: Saif Ul Islam

Submitted By:

Shizza Asher (20L-1184)
Haleema Jamil (20L-1210)
Amin Ahmad (20L-2164)

Department of Computer Science
FAST-NU, Lahore, Pakistan

1. Provide Data Set Dimension

Total number of original images: ~ 5100
Dimensions of the original dataset were very diverse. Therefore, scaling all images to a standard dimension is significant.

Total number of images after augmentation: ~ 6100
Dimension of processed images: 224 x 224 x 3

2. Training Set/Test Set Details

Original Training set:
Leaf_waste: 1000 samples
Metal: 1000 samples
Paper: 835 samples
Plastic: 692 samples
Wood_waste: 568 samples

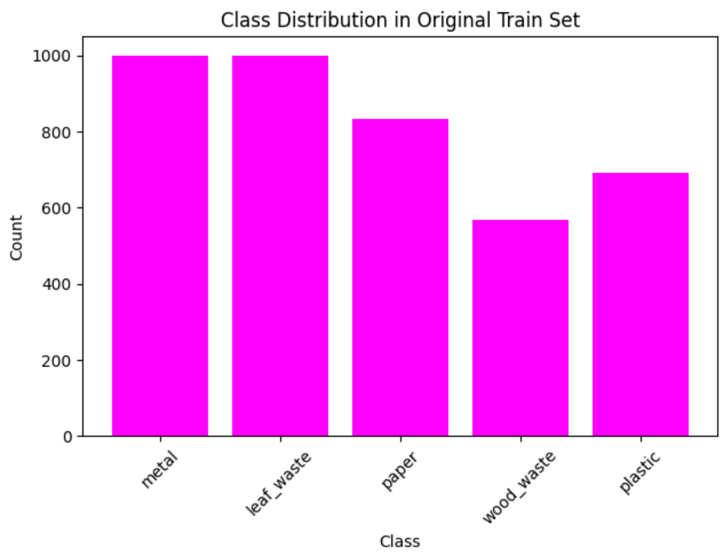
Preprocessed Training set:
Leaf_waste: 1000 samples
Metal: 1000 samples
Paper: 1003 samples
Plastic: 1040 samples
Wood_waste: 994 samples

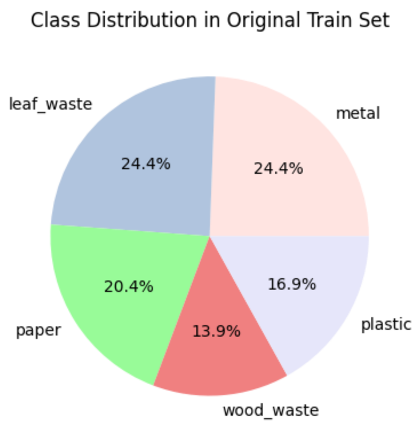
Validation set:
Leaf_waste: 394 samples
Metal: 169 samples
Paper: 212 samples
Plastic: 83 samples
Wood_waste: 59 samples

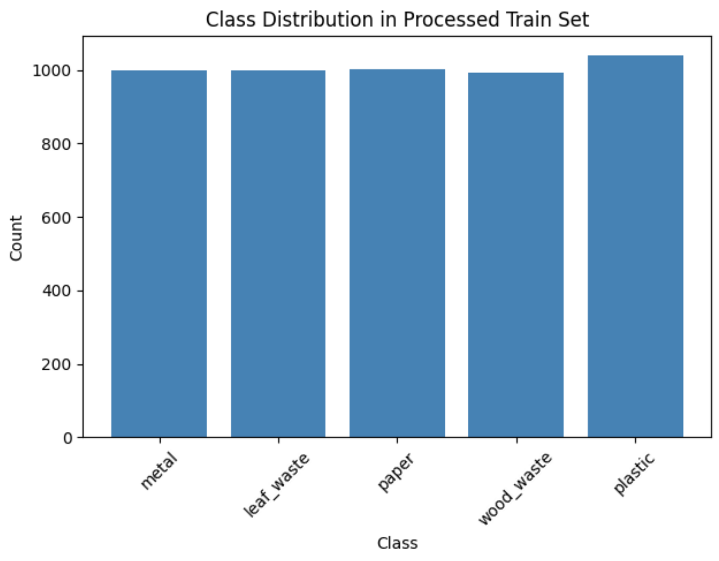
Testing set:
Leaf_waste: 25 samples
Metal: 25 samples
Paper: 25 samples
Plastic: 25 samples
Wood_waste: 25 samples

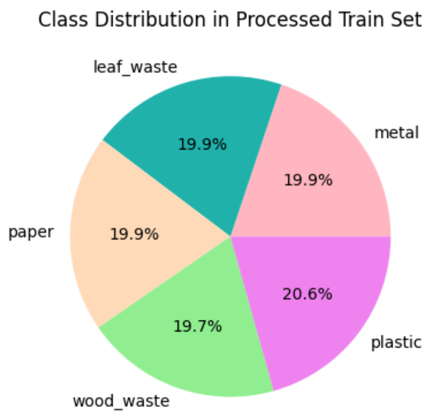
3. Class Distribution

Firstly, we count and record the number of images in each waste class in our dataset, organizing the information in a Pandas DataFrame for analysis and further processing. Then, using that, we create a figure with two subplots, one with a bar chart and the other with a pie chart to visualize the class distribution of each waste class in the original and processed train set for comparison. The visualizations are as follows:



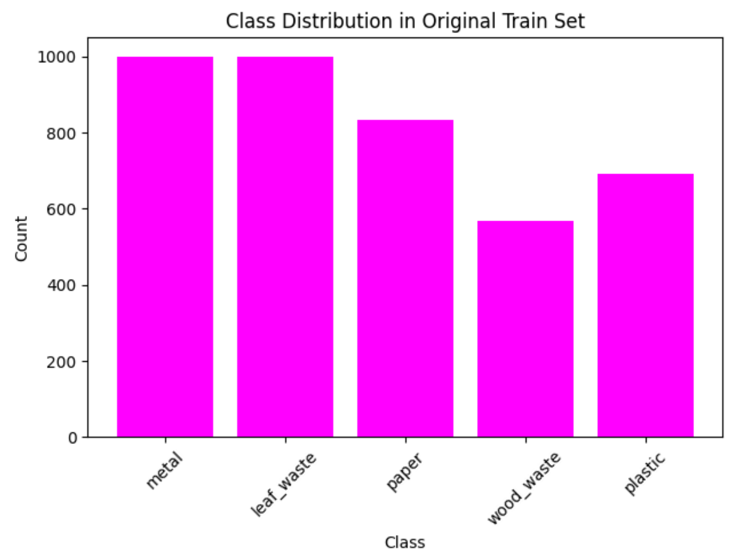


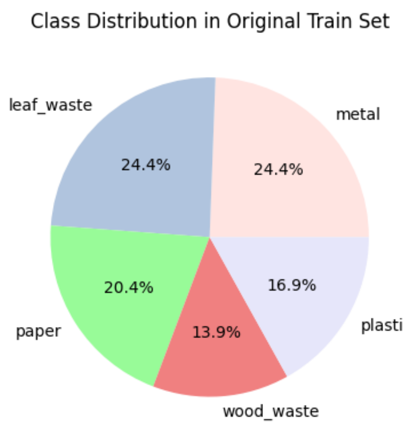


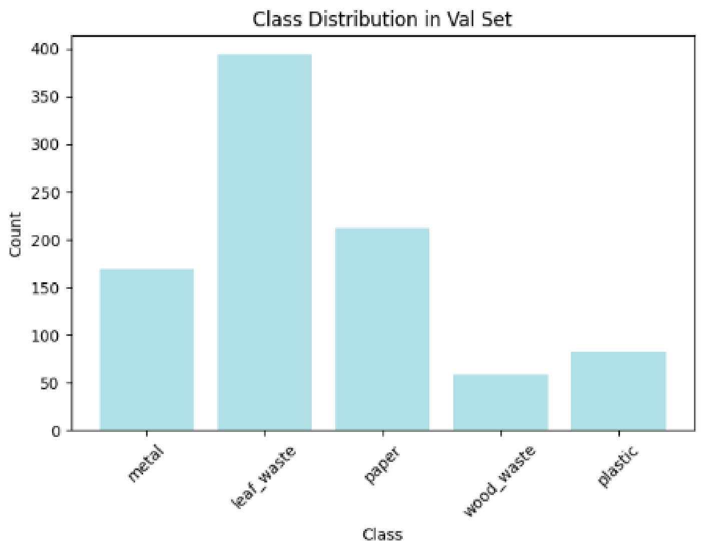


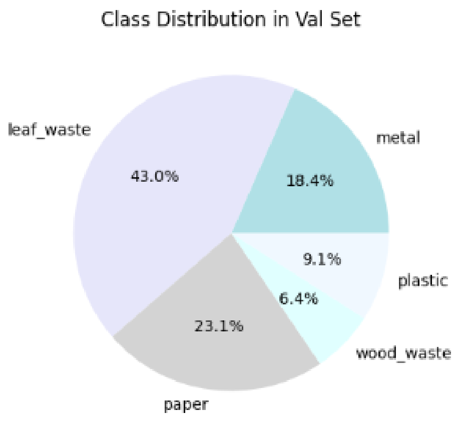
4. Class distribution in Train/Test Set Separately

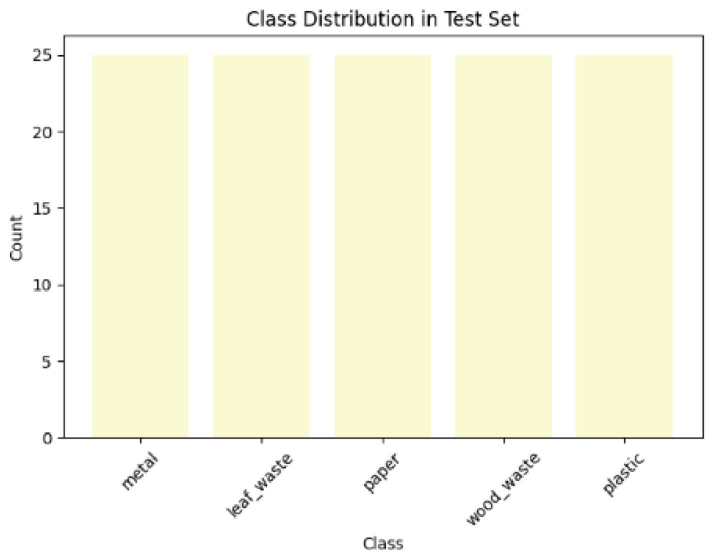
Now, we repeat point 3 for train, validation, and test sets. We create a figure with two subplots, one with a bar chart and the other with a pie chart to visualize the class distribution of each waste class in the original and processed train set, validation set, and test set. For each, we use a different color theme to highlight the differences. The results are as follows:

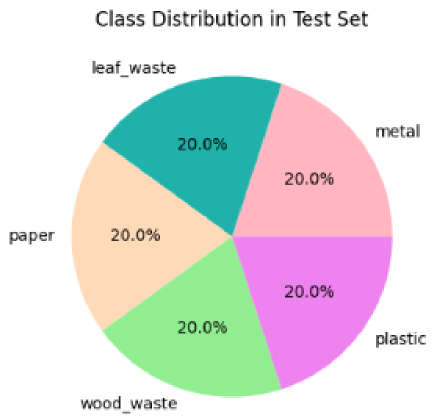









Class Distribution in Test Set
Count
25
20
15
10
5
0
metal
leaf_waste
paper
wood_waste
plastic
Class

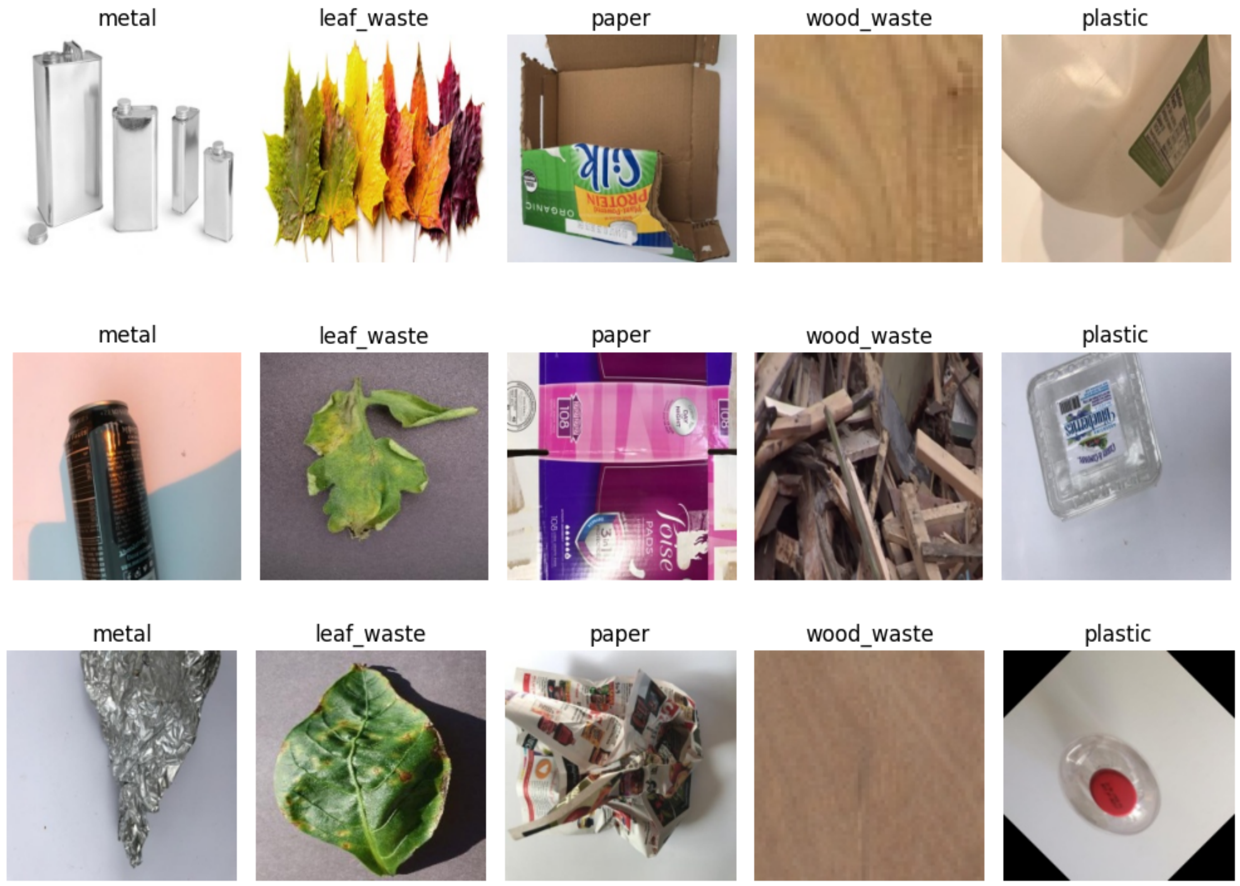

Class Distribution in Test Set
leaf_waste
metal
paper
wood_waste
plastic
20.0%
20.0%
20.0%
20.0%
20.0%

5. Sample Images

We randomly select 3 images from each of the waste classes. We create a figure with three subplots and display the images using subplots. The results are shown below:

6. Image Thumbnails

For each waste class, we select three random images from that class and display them as small thumbnail images in a grid. The thumbnails are displayed in columns, where each column represents a different class, and each column displays a random image from that class. This is to provide a visual sample of images from different classes in the dataset, making it easier to understand and explore the data.


metal
leaf_waste
paper
wood_waste
plastic
metal
leaf_waste
paper
wood_waste
plastic
metal
leaf_waste
paper
wood_waste
plastic

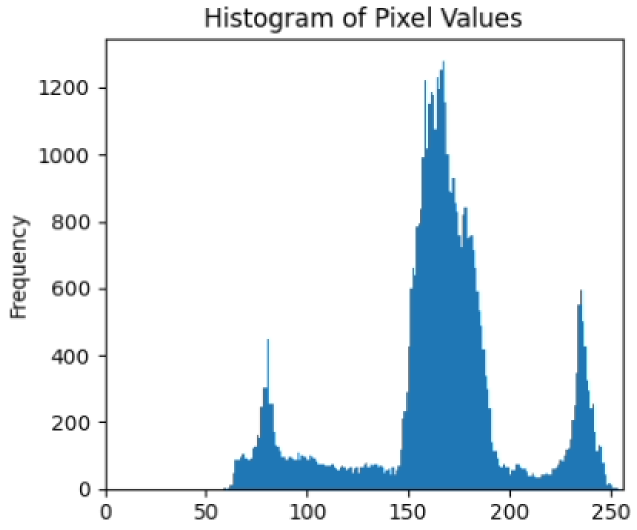
7. Histogram of Pixel Values

Again, we randomly select three images from the entire dataset and for each image, the following is done:

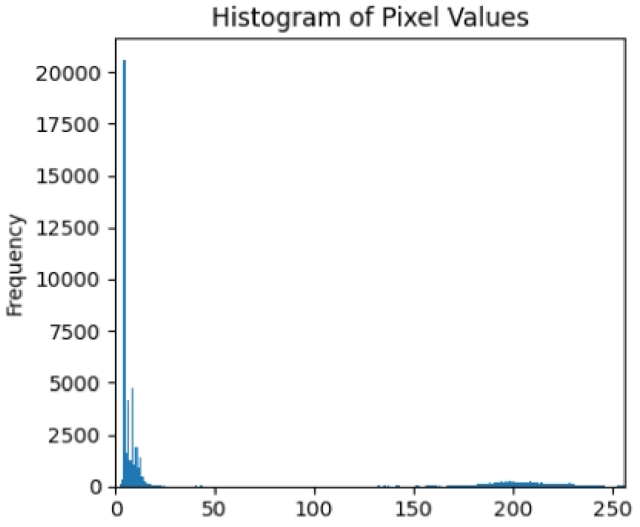
1. The image is read using the OpenCV library.
2. A figure is created for plotting with two subplots.
3. In one subplot, the original image is displayed.
4. In the second subplot, a histogram of the image's pixel intensity values is plotted.
5. Finally, both the image and its histogram are shown side by side in the same figure.

This is to visualize the original images and their corresponding pixel intensity histograms, which can be useful for image analysis and processing tasks.

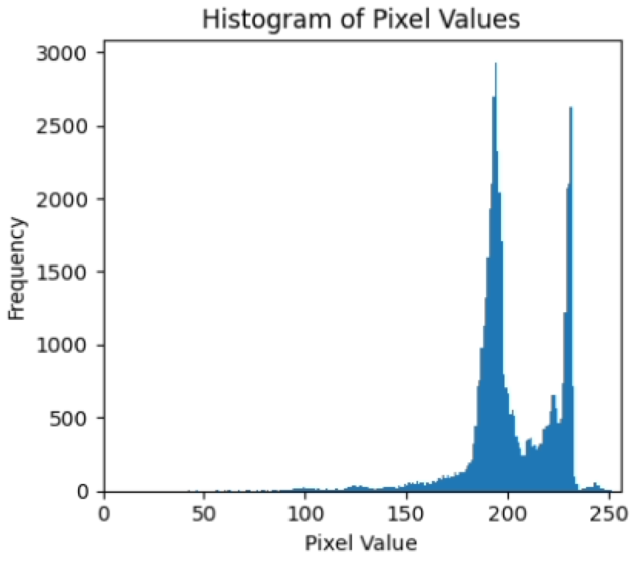
Original Image


Histogram of Pixel Values
Frequency
0
200
400
600
800
1000
1200
0
50
100
150
200
250

Original Image


Histogram of Pixel Values
Frequency
0
2500
5000
7500
10000
12500
15000
17500
20000
0
50
100
150
200
250

Original Image

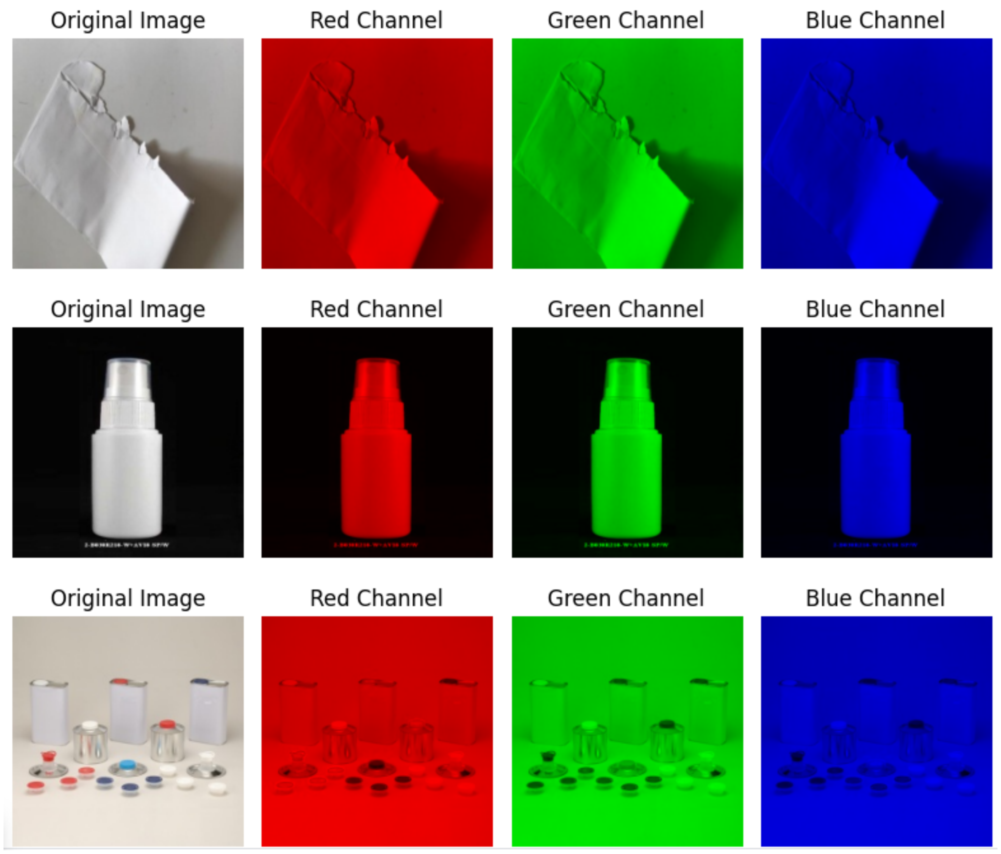

Histogram of Pixel Values
Frequency
0
500
1000
1500
2000
2500
3000
0
50
100
150
200
250
Pixel Value

8. Color Channel Separation

Similarly, for three randomly picked images, the following is performed:

1. The image is read using OpenCV.
2. A figure is created for plotting with four subplots.
3. In the first subplot (1st out of 4), the original image is displayed.
4. In the second subplot (2nd out of 4), the color channels of the image, typically red, green, and blue channels, are separated and visualized.
5. Both the original image and the separated color channels are then shown side by side in the same figure.

This allows the visualization of both the original images and their color channels, helping in understanding the color composition of the images.

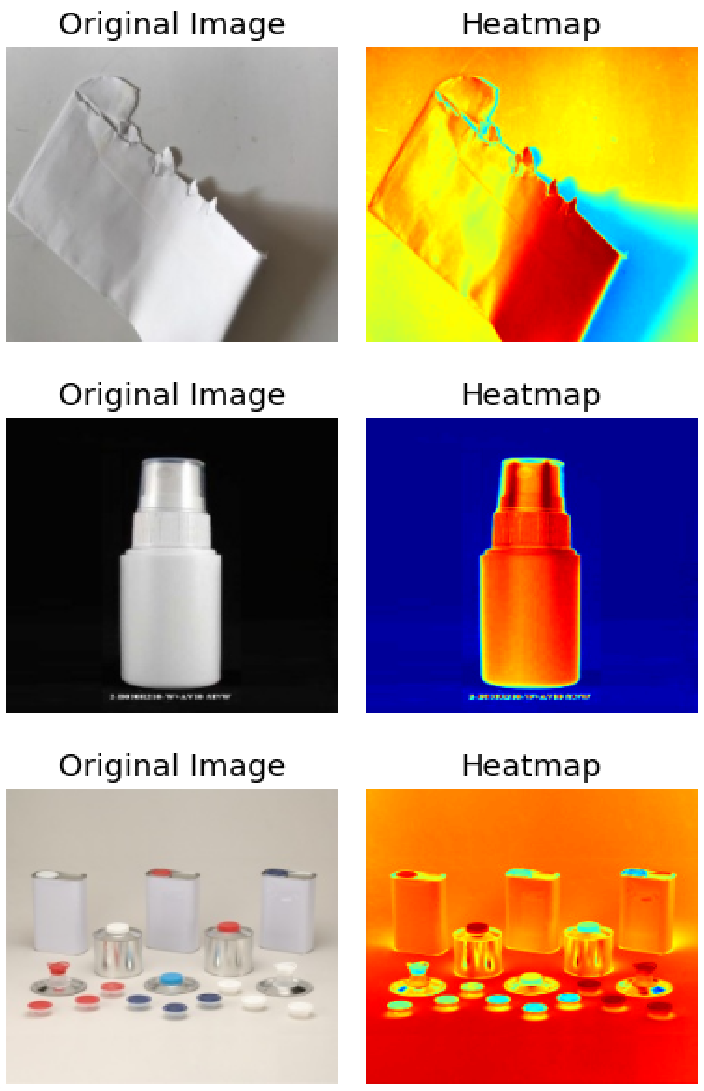


9. Heatmaps

Similarly, for three randomly picked images, the following is performed:

1. The image is read using OpenCV.
2. A figure is created for plotting with two subplots.
3. In the first subplot (1st out of 2), the original image is displayed.
4. In the second subplot (2nd out of 2), a heatmap is generated and visualized based on the image.
5. Both the original image and the generated heatmap are then shown side by side in the same figure.

This allows the visualization of the original images and the corresponding heatmaps, facilitating the analysis of image characteristics.



10. Data Augmentation Examples

We define the ***augment*** function to apply a series of image transformations to an input image. These transformations include rotating the image by 45 degrees, flipping it horizontally, and flipping it vertically. The ***rotate_image*** function takes an input image and rotates it by a specified angle. It calculates a rotation matrix and applies it to the image using OpenCV's *warpAffine* function. The ***horizontal_flip*** function horizontally flips an image using OpenCV's *flip* function. The ***vertical_flip*** function vertically flips an image using OpenCV's *flip* function.

Then, we use the following steps to create the augmentations:

1. An image is read in RGB format from a specified file path.
2. The *augment* function is called on the input image, resulting in three augmented images: a rotated image, a horizontally flipped image, and a vertically flipped image.
3. A 2x2 grid of subplots is created to display the original image and its augmentations.
4. The original image is displayed in the first subplot with an appropriate title.
5. The augmented images are displayed in the remaining subplots, each with a corresponding title indicating the type of augmentation applied.
6. The spacing is adjusted for a clean layout, and the entire plot is displayed to visualize the original and augmented images.

This code demonstrates image augmentation techniques that can be useful for expanding a dataset for machine learning and computer vision tasks.