chrisheimbuch /
traffic_signs_classification_report

Code Issues Pull requests Actions Projects Wiki Security In

traffic_signs_classification_report / source / traffic_eda.ipynb

chrisheimbuch structure update d2658d1 · now

1070 lines (1070 loc) · 674 KB

Preview Code Blame Raw

Traffic Signs EDA - Research Report Part 1 of 2

Dataset:

https://www.kaggle.com/datasets/ahemateja19bec1025/traffic-sign-dataset-classification/data?select=labels.csv

Chris Heimbuch: https://github.com/chrisheimbuch

Table of Contents

Overview

In this notebook, I will be working on an EDA on an image dataset consisting of Traffic Signs. There are 58 different types of signs, with all different varying amounts of pictures of each sign. I will go on to explore the CSV file attached to understand what each type of sign is, inspect if there are any null values, and make any changes necessary if the data does not seem uniform. Next I will view the distribution of widths and heights of all the images, the RGB channel intensity, class distribution, aspect ratios, and total images we have to work with. This will be a fun EDA which will tie into my next notebook, where we will dive into machine learning and deep learning!

In [1]:

```python
#Standard imports.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Libraries for image directory location manaagement and image manipulation.
import os
import cv2
from collections import defaultdict
from PIL import Image

#Ignore all warnings.
import warnings
warnings.filterwarnings("ignore")
```

Section 1: Data Cleaning and Inspection

In [2]:

```python
#Read in label class file for see classes of signs.
path = r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_4\P4_Project\
df = pd.read_csv(path)
```

In [51]:

```python
#Visually inspect the classes and see what type of images I am working with.
df.head(15)
```

Out[51]:

	ClassId	Name
0	0	Speed Limit (5Km/H)
1	1	Speed Limit (15Km/H)
2	2	Speed Limit (30Km/H)
3	3	Speed Limit (40Km/H)
4	4	Speed Limit (50Km/H)
5	5	Speed Limit (60Km/H)

6	6	Speed Limit (70Km/H)
7	7	Speed Limit (80Km/H)
8	8	Dont Go Straight Or Left
9	9	Dont Go Straight Or Right
10	10	Dont Go Straight
11	11	Dont Go Left
12	12	Dont Go Left Or Right
13	13	Dont Go Right
14	14	Dont Overtake From Left

There are columns in which are lower case and title case. I will make all words in the "Name" column title case so it looks nice and everything is uniform.

In [4]:

```
#Making all names title case to deal with random sporadic name conventions.
df['Name'] = df['Name'].str.title()
```

In [5]:

```
#Sanity Inspection
df.head()
```

Out[5]:

	ClassId	Name
0	0	Speed Limit (5Km/H)
1	1	Speed Limit (15Km/H)
2	2	Speed Limit (30Km/H)
3	3	Speed Limit (40Km/H)
4	4	Speed Limit (50Km/H)

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58 entries, 0 to 57
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   ClassId  58 non-null     int64 
 1   Name     58 non-null     object
dtypes: int64(1), object(1)
memory usage: 1.0+ KB
```

In [7]:

```
# Inspecting duplicate names in the df and printing out all instances of duplic
duplicate_names = df[df['Name'].duplicated(keep=False)]['Name'].unique()

# Display the duplicate names
```

```
print(duplicate_names)
```

```
['Speed Limit (40Km/H)' 'Speed Limit (50Km/H)' 'Bicycles Crossing']
```

In [8]:

```
#Setting "dataset" variable to the path where my data images are stored
dataset = r"C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_4\P4_Proje
```

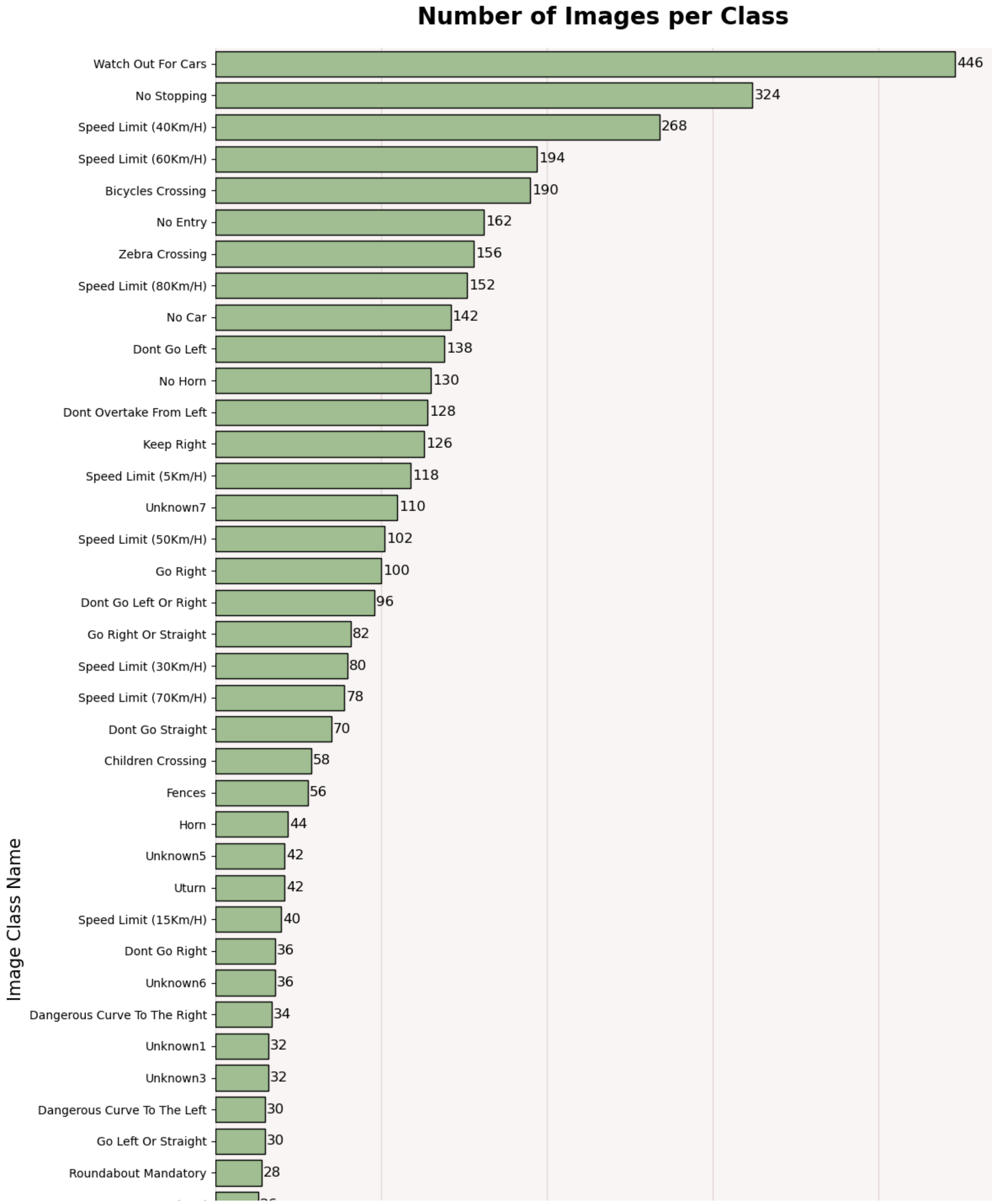
Section 2: Data Visualization

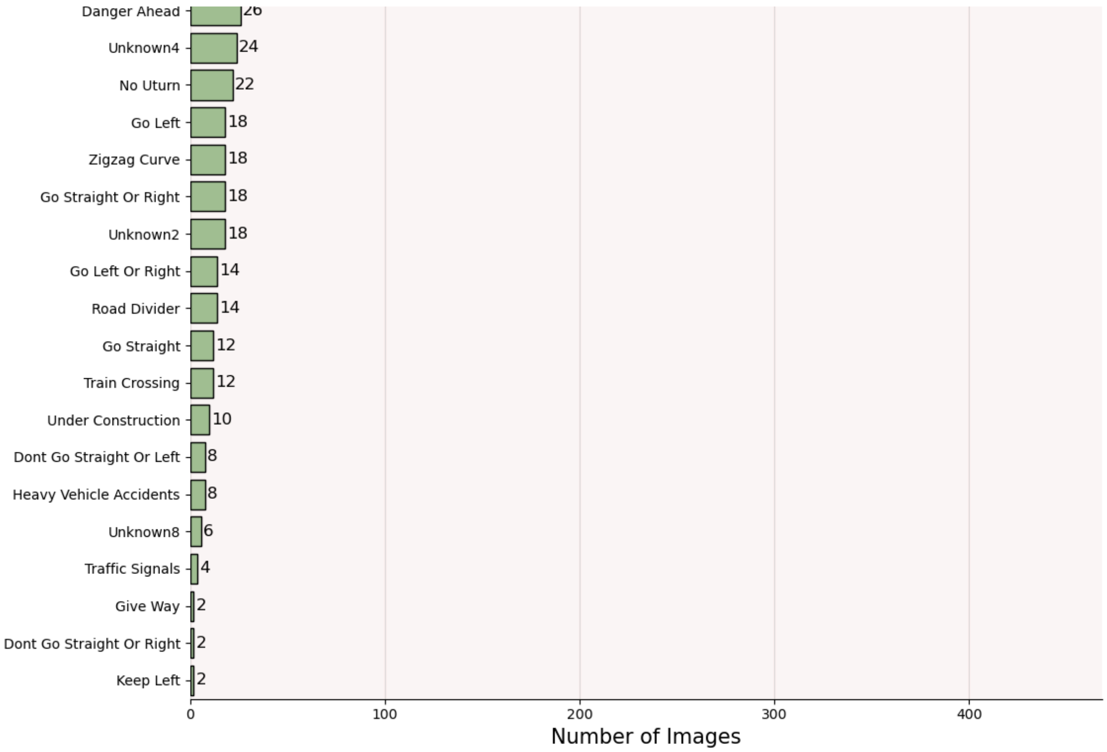
1. What are the class distribution among the images?

In [10]:

```
#Set data to an empty list to extract the data from the directory and build a d
data = []

# Loop through each folder (class ID) inside the dataset directory
for folder in os.listdir(dataset):
    folder_path = os.path.join(dataset, folder)

    # Check if it is a directory
    if os.path.isdir(folder_path):
        # Filter only image files based on their extensions
        num_images = len([f for f in os.listdir(folder_path) if os.path.isfile(

        # Check if the folder name is an integer, else skip
        try:
            class_id = int(folder)
            class_name = df['Name'][class_id]  # Get the class name from the Da
        except (ValueError, KeyError):
            print(f"Skipping folder {folder} - invalid class ID or not found in
            continue

        # Append the class name and image count to the data list
        data.append([class_name, num_images])

#Create a DataFrame
display_data = pd.DataFrame(data, columns=['Class_Name', 'num_images'])

#Group by Class_Name to sum up the number of images for duplicate classes
grouped_data = display_data.groupby('Class_Name', as_index=False).agg({'num_ima

#Sort the data by 'num_images' in ascending order for the plot
grouped_data_sorted = grouped_data.sort_values('num_images', ascending=True)

#Set up the plot to visualize our data.
fig, ax = plt.subplots(figsize=(12, len(grouped_data_sorted) * 0.4))
ax.barh(grouped_data_sorted['Class_Name'], grouped_data_sorted['num_images'], c

#Add title, x labels, and adjust y limiter to get rid of unneccesary whitespace
plt.title('Number of Images per Class', fontsize=20, weight='bold', pad=20)
plt.xlabel('Number of Images', fontsize=15)
plt.ylabel('Image Class Name', fontsize=15)
```

```
ax.set_ylim(-0.5, len(grouped_data_sorted) - 0.5)
ax.grid(axis='x', alpha=0.4)
ax.set_facecolor('#FCF6F5')

ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Annotate each bar for their values.
for index, value in enumerate(grouped_data_sorted['num_images']):
    plt.text(value + 1, index, f'{value}', ha='left', va='center', fontsize=12)

# Display the plot
plt.tight_layout()
plt.show()
```



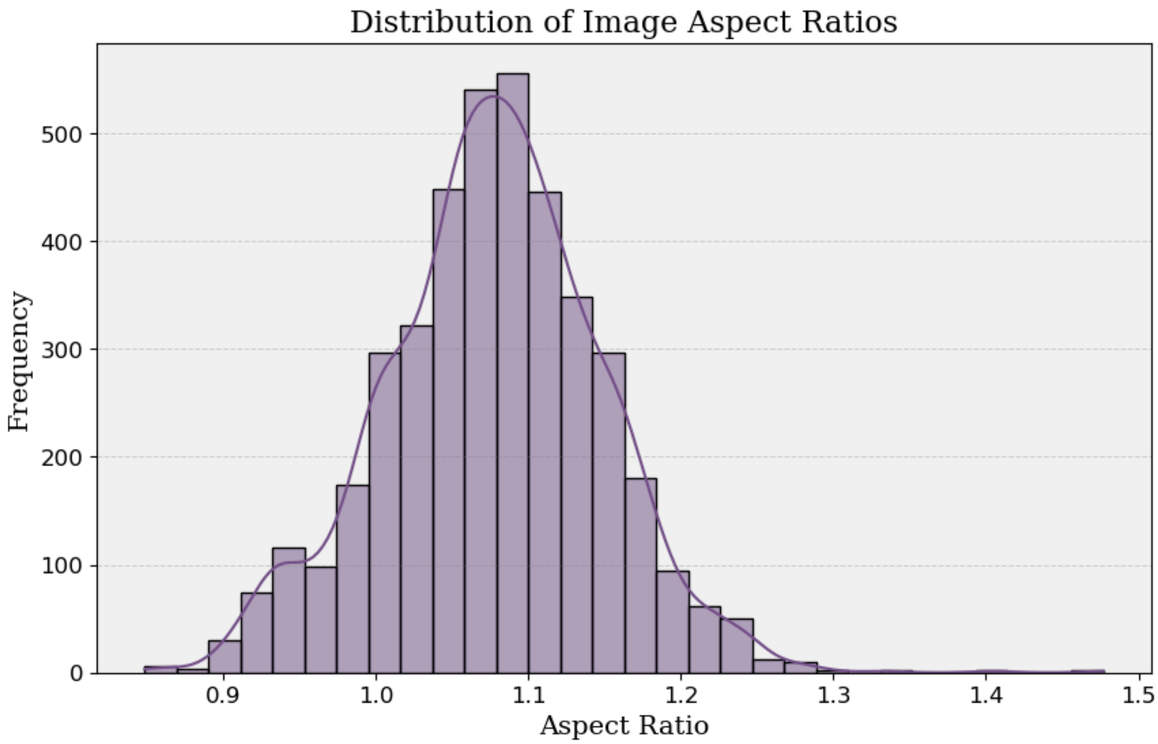


2. What is the distribution of image aspect ratios?

In [28]:

```
#Empty list to store aspect ratios
aspect_ratios = []

# Loop through each folder (class ID) inside the dataset directory
for folder in os.listdir(dataset):
    folder_path = os.path.join(dataset, folder)

    # Check if it's a directory that exist, and loop through each image file in
    if os.path.isdir(folder_path):
        # Loop through each image file in the folder
        for filename in os.listdir(folder_path):
            image_path = os.path.join(folder_path, filename)

            # Check if it's a file (image)
            if os.path.isfile(image_path):
                try:
                    # Open the image and get its dimensions
                    with Image.open(image_path) as img:
                        width, height = img.size
                        # Calculate the aspect ratio (width / height)
                        aspect_ratio = width / height
                        aspect_ratios.append(aspect_ratio)
                except Exception as e:
                    print(f"Error processing image {image_path}: {e}")

# Convert aspect ratios list to a DataFrame for visualization
aspect_data = pd.DataFrame(aspect_ratios, columns=['Aspect_Ratio'])

#Set up the plot
plt.figure(figsize=(10, 6))

#Custom font styling
font1 = {'family':'serif','color':'black','size':16}
```

```
font1 = {'family':'serif','color':'black','size':16}
font2 = {'family':'serif','color':'black','size':14}

#Set up the histogram plot with data.
sns.histplot(aspect_data['Aspect_Ratio'], bins=30, kde=True, color='#76528BFF')

#Set title and axis names.
plt.title("Distribution of Image Aspect Ratios", fontdict=font1)
plt.xlabel("Aspect Ratio", fontdict=font2)
plt.ylabel("Frequency", fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

#Grid customization
ax = plt.gca()
ax.set_facecolor('#f0f0f0')
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='gr

plt.show()
```

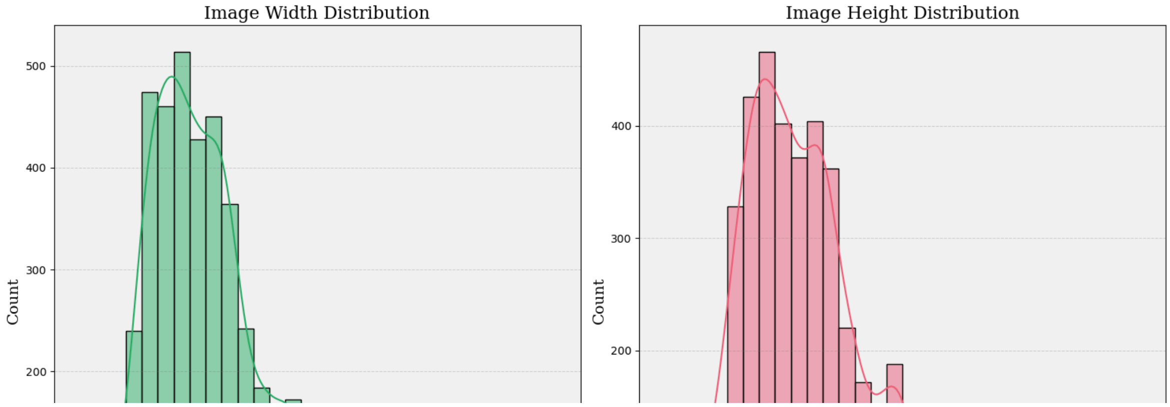


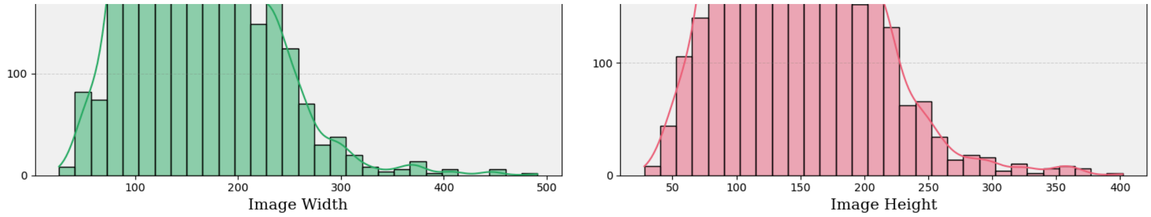
3. What is the average distribution of widths and heights of all the images?

In [45]:

```
# Lists to store width and height of images
width_list = []
height_list = []

# Loop through each folder in the dataset
for folder in os.listdir(dataset):
    folder_path = os.path.join(dataset, folder)

    # Check if it's a directory
    if os.path.isdir(folder_path):
        # Iterate through each image file in the folder
        # Loop through each image file in the folder
        for filename in os.listdir(folder_path):
            image_path = os.path.join(folder_path, filename)

            # Open image and get its dimensions
            try:
                with Image.open(image_path) as img:
                    width, height = img.size
                    width_list.append(width)
                    height_list.append(height)
            except Exception as e:
                print(f"Error opening {image_path}: {e}")

# Calculate average width and height
average_width = sum(width_list) / len(width_list)
average_height = sum(height_list) / len(height_list)

print(f'Average width: {average_width} and height: {average_height}')

# Plot the width and height distributions
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

# Plot for image widths
sns.histplot(width_list, ax=ax[0], bins=30, kde=True, color='#2BAE66FF')
ax[0].set_title('Image Width Distribution', fontdict=font1)
ax[0].set_xlabel("Image Width", fontdict=font2)
ax[0].set_ylabel("Count", fontdict=font2)

#Grid customization for axis 0
ax[0].set_facecolor('#f0f0f0')
ax[0].grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='

# Plot for image heights
sns.histplot(height_list, ax=ax[1], bins=30, kde=True, color='#EF6079FF')
ax[1].set_title('Image Height Distribution', fontdict=font1)
ax[1].set_xlabel("Image Height", fontdict=font2)
ax[1].set_ylabel("Count", fontdict=font2)


#Grid customization for axis 1
ax[1].set_facecolor('#f0f0f0')
ax[1].grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='

plt.tight_layout()
plt.show()
```

```
Average width: 152.13717026378896 and height: 140.85659472422063
```





4. What are the average Red channel intensities among the images?

In [14]:

```
# Initialize a list to store RGB channel intensities
rgb_data = []

# Loop through each folder and image
for folder in os.listdir(dataset):
    folder_path = os.path.join(dataset, folder)

    if os.path.isdir(folder_path):
        for filename in os.listdir(folder_path):
            image_path = os.path.join(folder_path, filename)

            # Open the image
            with Image.open(image_path) as img:
                # Convert the image to RGB
                img = img.convert('RGB')

                # Convert image data to a NumPy array (easier to manipulate)
                img_array = np.array(img)

                # Calculate mean intensity for each channel (R, G, B)
                red_mean = np.mean(img_array[:, :, 0])  # Red channel
                green_mean = np.mean(img_array[:, :, 1])  # Green channel
                blue_mean = np.mean(img_array[:, :, 2])  # Blue channel

                # Store the results in the list
                rgb_data.append([filename, red_mean, green_mean, blue_mean])

# Create a DataFrame to display the RGB intensities
rgb_df = pd.DataFrame(rgb_data, columns=['Image_Name', 'Red_Intensity', 'Green_
```

In [15]:

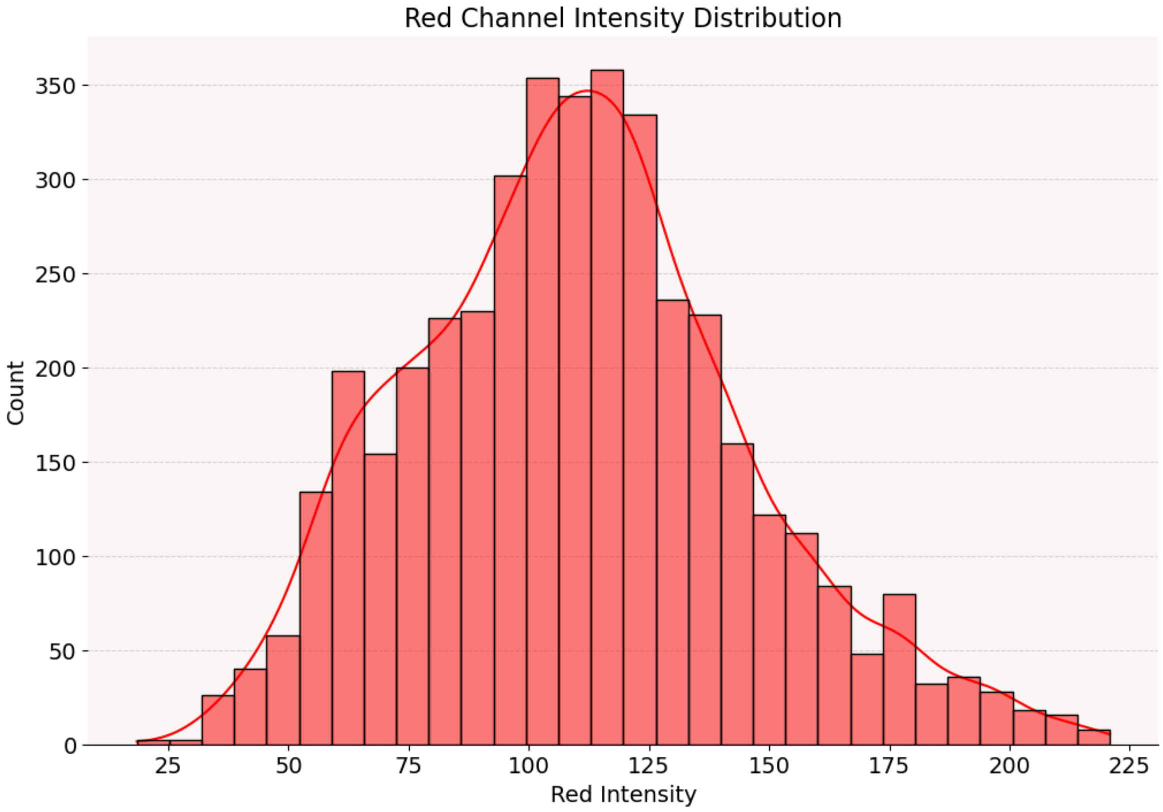
```
#Inspect the df of all the image intensities of RGB values.
rgb_df.head()
```

Out[15]:

	Image_Name	Red_Intensity	Green_Intensity	Blue_Intensity
0	000_0001.png	125.130830	102.479653	92.594683
1	000_0002.png	124.596709	96.950552	88.883765
2	000_0003.png	126.347656	103.322170	97.209529
3	000_0004.png	120.299174	92.751322	85.797934
4	000_0005.png	116.834804	89.274101	82.158333

In [16]:

```
# Set the size of the plot
fig, ax = plt.subplots(figsize=(12, 8))

# Plot the histogram for the Red channel
sns.histplot(rgb_df['Red_Intensity'], bins=30, color='red', kde=True, zorder=3)

#Set Title, x & y axis names, face color
plt.title('Red Channel Intensity Distribution', fontsize=16)
ax.set_xlabel("Red Intensity",fontsize=14)
ax.set_ylabel("Count",fontsize=14)
ax.tick_params(labelsize=14)
ax.set_facecolor('#FCF6F5')

#Remove lines of grid perimeter to make more appealing.
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Set background of grid to custom color.
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.3

#Display plot
plt.show()
```

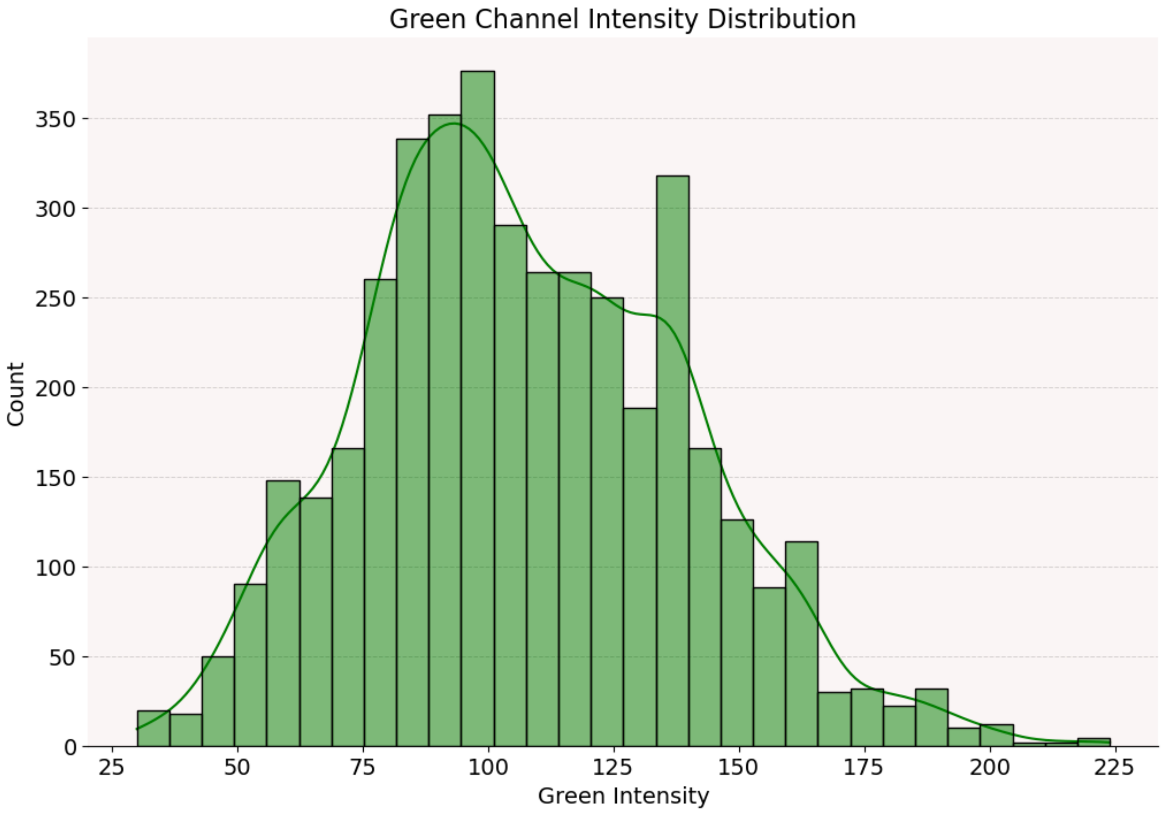


5. What are the average Green channel intensities among the images?

In [17]:

```
# Set the size of the plot
fig, ax = plt.subplots(figsize=(12, 8))
```

```
# Plot the histogram for the Green channel
sns.histplot(rgb_df['Green_Intensity'], bins=30, color='green', kde=True, zorde

#Set Title, x & y axis names, face color
plt.title('Green Channel Intensity Distribution', fontsize=16)
ax.set_xlabel("Green Intensity",fontsize=14)
ax.set_ylabel("Count",fontsize=14)
ax.tick_params(labelsize=14)
ax.set_facecolor('#FCF6F5')

#Remove lines of grid perimeter to make more appealing.
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Set background of grid to custom color.
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.3

#Display plot
plt.show()
```

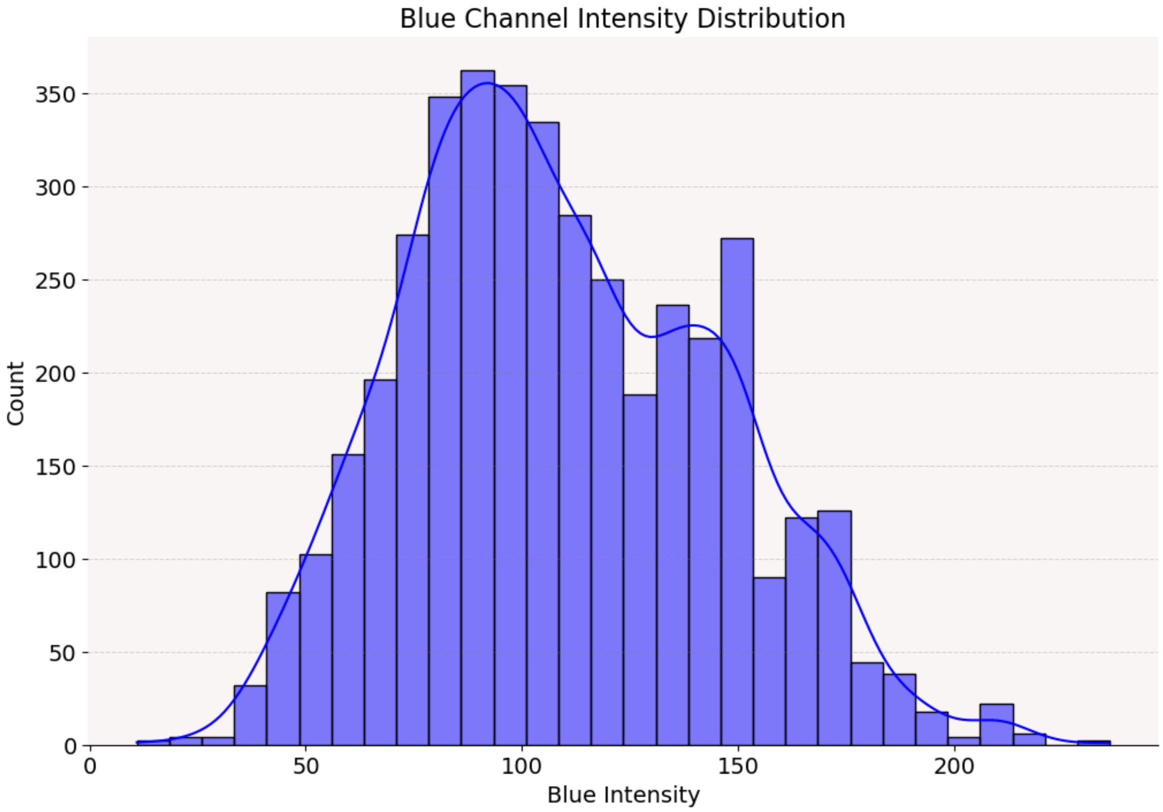


6. What are the average Blue channel intensities among the images?

In [18]:

```
# Set the size of the plot
fig, ax = plt.subplots(figsize=(12, 8))

# Plot the histogram for the Blue channel
sns.histplot(rgb_df['Blue_Intensity'], bins=30, color='blue', kde=True)
```

```
#Set Title, x & y axis names, face color
plt.title('Blue Channel Intensity Distribution', fontsize=16)
ax.set_xlabel("Blue Intensity",fontsize=14)
ax.set_ylabel("Count",fontsize=14)
ax.tick_params(labelsize=14)
ax.set_facecolor('#FCF6F5')

#Remove lines of grid perimeter to make more appealing.
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Set background of grid to custom color.
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.3

#Display plot
plt.show()
```



7. How many total Images are there in training?

In [19]:

```
# Make a counter to count images
total_images = 0

# Loop through each folder inside the dataset directory
for folder in os.listdir(dataset):
    folder_path = os.path.join(dataset, folder)
    #check to see if path is a valid directory
    if os.path.isdir(folder_path):
        # Count the number of image files in the folder
        num_images = len([f for f in os.listdir(folder_path) if os.path.isfile(
        # Add the count to the total
        total_images += num_images

# Output the total number of images
print(f"Total number of images: {total_images}")
```

```
Total number of images: 4170
```

Section 3: Inferential Analysis

This dataset does not make sense to do an inferential analysis on the images, as it probably won't reveal any meaningful differences among pixels right off the bat. That would require an advanced deep learning approach to extract deep heuristics and learn on those heuristics. We could do an analysis on RGB intensity, but I don't think that would help us moving forward.

Section 4: Analysis

Analysis

The first thing I have done in my analysis was analyze the tabular data since it was only names of the classes. After inspecting the head, I noticed that class names were all different in that some were title case, many were lower case. I first put everything in title case to clean it up and make it uniform. I also noticed there were a few duplicates which was revealed to me in class visualization. The duplicate classes that were listed were speed limit of 40 & 50 kilometer per hour signs and bicycles crossing signs. I aggregated them for modeling so it picks up everything in the right class. Because there were roughly 8 "40 kilometer per hour' pictures stored separately in a different folder than the majority, the graph displayed the results very strangely, so this was my solution to fix the problem.

I first modeled the distribution of my data to get an idea of class balance. There is a class imbalance among the images, with "Watch Out For Cars" signs having the most with 446 images and "Keep Left" signs having the least with 2 images. Next, I observed the distribution of my image aspect ratios. The majority of images in the dataset have an aspect ratio of 1.1. I then examined the distributions of image heights and widths. The majority of images have a mean image width of roughly 140 pixels, while the majority of images of a mean image height 130 pixels. I observed that there are 4170 images total within the data. I finally observed the intensity of red, blue, and green within the images. The images seem to have more red intensity on average when compared to green and blue intensity.

> Overall, this EDA has been a fun learning experience, learning about the os library and understanding how to navigate image data. Next, I will prepare my data for machine learning!

Resources / References

https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/

https://neptune.ai/blog/data-exploration-for-image-segmentation-and-object-detection

https://jacobheyman702.medium.com/examples-of-eda-for-image-analysis-4d7770924fb5

https://github.com/henrhoi/image-classification/blob/master/feature_extraction_and_exploratory_data_analysis.ipynb

https://www.kaggle.com/code/tarunpaparaju/plant-pathology-2020-eda-models/notebook

https://www.kaggle.com/code/nickyazdani/object-detection-from-scratch-using-tensorflow

https://www.kaggle.com/code/boulahchichenadir/cnn-classification