SFU SIMON FRASER UNIVERSITY

DSP Image Processing Studio With Object Recognition and Image Segmentation Using YOLOv8

ENSC 429: Final Project Report

Group 1

Elaine Luu	(301392121)
Manraj Singh Rai	(301375061)
Clarence Wasilwa	(301429747)
Gurnek Ghatarora	(301394646)

Table of Contents

Abstract

This report presents the development and analysis of an image processing application which leverages Digital Signal Processing (DSP) techniques. Utilizing a pre-trained Convolutional Neural Networks (CNN) with the YOLOv8 model, the project implements key DSP concepts such as convolution, filtering, and sampling. Techniques like Gaussian blur, sharpening, and edge detection are employed to enhance image quality and accuracy through a variety of kernels (filters) convolved with image pixels. The results demonstrate the effective application of DSP in image processing, providing insights into workload distribution and individual contributions within the project.

Introduction

Convolutional neural network (CNN) is the most widely used deep learning model in feature learning for large-scale image classification and recognition. They are used in a variety of applications, such as autonomous vehicles, security camera systems, and others. The CNN model is composed of a convolutional layer, a pooling layer and a fully connected layer. These layers are stacked to form deep architecture that are used in automating the feature extraction process by leveraging concepts from various technological fields including digital signal processing (DSP) [1]. CNNs have revolutionized the field of image recognition and can then be used to tie in concepts of Digital Signal Processing in analyzing images. Certain features of these images may be in need of additional processing once they are identified. In this application we initially use the YOLOv8 model to perform preprocessing, followed by object classification [12]. We then allow the user to selectively apply additional processing to the image such as Gaussian blur, sharpen, edge detection and noise reduction. This allows the user to utilize this application to selectively identify and process the various objects in the image, as required for their purposes.

Methodology and Analysis

Our project focused on developing an app that uses DSP concepts to implement Image Processing. The model utilized in this project is a Convolutional Neural Network (CNNs), specifically the YOLOv8 model, for image segmentation and then the user can apply various image processing transformations, such as Gaussian blur, sharpening, denoising, histogram equalization, and edge detection with the help of OpenCV. Below we will go through some concepts that are essential to the functionality of our project.

Convolutional Neural Network (CNN) and Image segmentation

Upon capturing or opening an image, the YOLOv8 model processes the input to identify objects and to segment the image. The identified objects have masks applied over them so the user can precisely modify only the selected object. These objects also have boxes drawn around them and labels applied so the user knows what they are modifying when they later apply various transformations.

To enhance the accuracy of the masks being applied on each of the objects, we can perform erosion then followed by dilation with the help of the functions in OpenCV. Erosion is used to trim the unnecessary pixels around an object, typically considered noise. We then follow this with dilation which increases the bounds of the mask to fill in the noise which was removed. This is a common technique for refining masks [2]. All the items we identify are then saved separately in a list, and also removed from the overall mask of the image. This leaves us with a “background” mask which can be used if we wish to modify everything except for the identified items.

The segmented objects and their corresponding masks are visually represented on the original image, with bounding boxes and labels drawn for easy identification. This allows for targeted transformations such as Gaussian blur, sharpening, denoising, histogram equalization, and edge detection to be applied selectively to specific objects or regions within the image. For instance, applying Gaussian blur within the mask ensures that only the selected object is blurred, leaving the rest of the image unaffected.

This segmentation and masking process is primarily used to allow for precise image manipulation by allowing detailed control over individual elements. The YOLOv8 model's ability to perform both detection and segmentation makes it a powerful tool for various computer vision tasks, from simple image editing to complex scene analysis.

Gaussian Blur

When performing image processing, convolution is an essential tool for applying various filters to an image. Gaussian blurring is one of our functions that employs convolution where a Gaussian kernel (a type of low-pass filter), is used to smooth an image [8]. The original image is convolved with a Gaussian kernel, derived from the Gaussian formula:

$$Gxy = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Here, (x, y) are the coordinates of the kernel relative to the center pixel, and sigma (σ) is the standard deviation of the Gaussian distribution and controls the strength of the blur (lower the sigma, the less the image is blurred, since the kernel is smaller and has less weight applied to the pixels). The normalization factor ($\frac{1}{2\pi\sigma^2}$) ensures that the sum of all kernel values equals 1, preventing alteration to overall image brightness. The exponential term determines the weight of each pixel based on its distance from the kernel's center. Pixels closer to the center are given higher weights, while those farther away are given lower weights [8]. As the kernel slides across the image, it effectively averages the pixel values within the area covered by the kernel through element-wise multiplications and summations. This smooths out sharp edges and fine details, leading to a blurred appearance [8][9].

Sharpen

The sharpening process applies convolution with the intent to enhance the edges and details of an image. A sharpening kernel (a type of high-pass filter) is used to amplify high-frequencies while filtering out the low-frequencies. The kernel often includes both positive and negative values. When convolved with the original image, this kernel increases contrast at the edges and highlights fine details by sliding across the whole image, in the same fashion as the Gaussian kernel. Common sharpening kernels include the Laplacian filter and the standard high-pass filter. The Laplacian filter uses a kernel that generally includes a central positive value surrounded by negative values, in order to emphasize the rapid changes from high to low frequency (and vice-versa). The high-pass filter, on the other hand, preserves high-frequency components by simply subtracting a low-pass filtered version of the original image, thus enhancing the edges. Our kernel is as follows[10][11]:

0	-1	0
-1	5	-1
0	-1	0

Edge Detection

As mentioned earlier, edge detection is a crucial step in image processing that is used in our project. It involves identifying significant transitions in intensity within an image, which tend to correspond to the boundaries of objects. The technique we used is known as Canny Edge Detection. It has optimal performance in terms of noise reduction and accurate detection of edges. It involves multiple steps such as, noise reduction using a Gaussian filter, gradient calculation, non-maximum suppression, and edge tracking by hysteresis. The combination of these steps ensures that edges are detected with minimal noise and accurate localization [5]. Our model uses edge detection indirectly in the sharpening process. This is done by enhancing the edges which allows sharpening algorithms to make the transitions between different regions more pronounced.

Denoising

Denoising is used to remove noise from images while retaining the main details, to improve overall image quality. Our method undertakes an adaptive technique which bases the filter's strength on local characteristics of the image, mainly being local variance. This local variance indicates the level of detail or noise, so that the filter strength can be scaled accordingly. This in turn will effectively reduce noise in smoother areas while preserving details in areas with more sudden changes in frequencies (e.g. lots of details or lines in the image) [13].

Histogram Equalization

Histogram equalization is another key DSP technique performed by our model. Histogram equalization is typically used in image enhancement and involves intensity transformations that redistribute the pixel values based on their cumulative distribution, thereby enhancing the contrast. This is initially done by computing the histogram of the pixel intensities, which represents the frequency distribution of each intensity level. Then this histogram is then used to construct the cumulative distribution function (CDF). The CDF is then employed to map the original intensity values to new values such that the resulting histogram is approximately uniform, thereby enhancing the contrast of the image.

Results and Discussion

In this section, we practically demonstrate and discuss the functionality of our application as described above.

Gaussian Blur: In Figure 1, we can see how Gaussian Blur is performed by the application. It is applied on the entire image, except for one car. This makes the rest of the image appear blurry, with softer edges and reduced noise. This allows the user to highlight specific aspects of the image that they want to direct more focus towards.

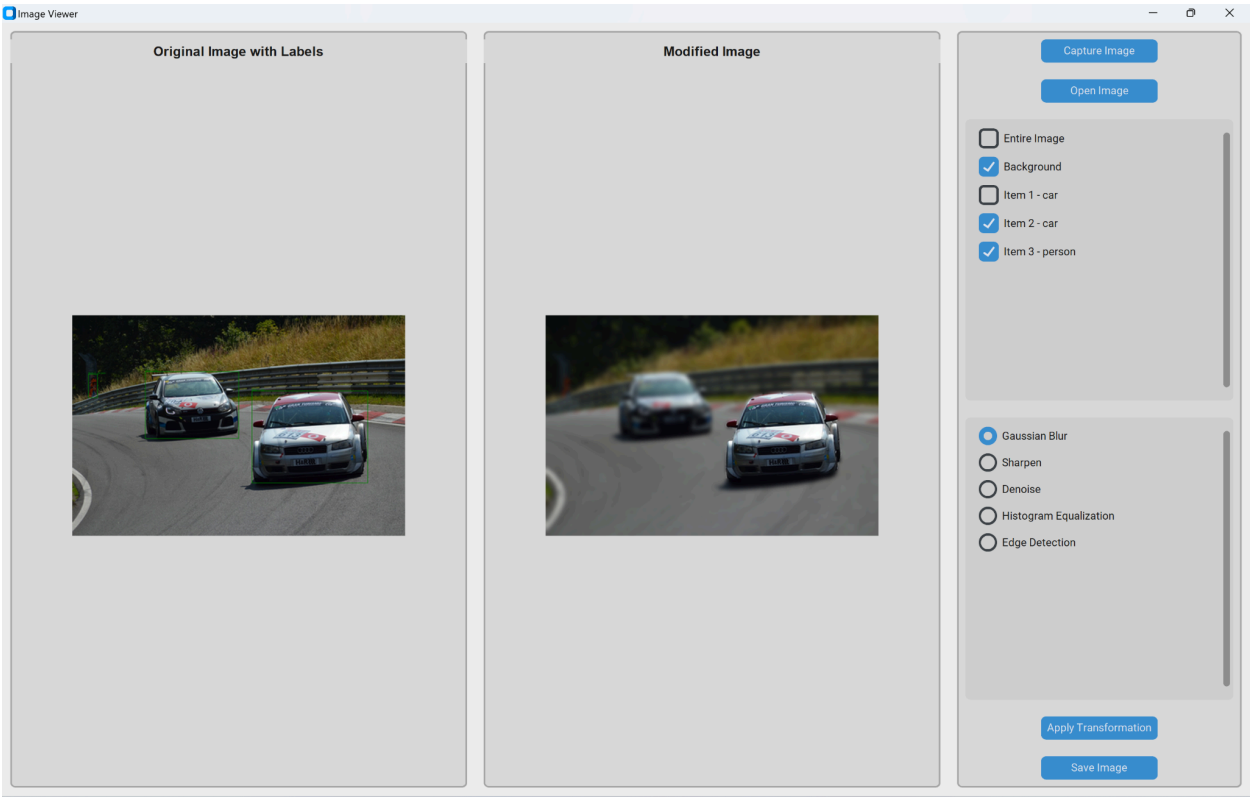

Figure 1: Gaussian Blur

Sharpening: In Figure 2 we can see the fur on the original image is quite muddled and difficult to discern details from. Upon applying our sharpening, we can see the edges more clearly highlighted now that they have been processed. This can be useful for highlighting additional details (i.e. the fur) that have not been clearly captured in the original image. These can then be more easily identified.

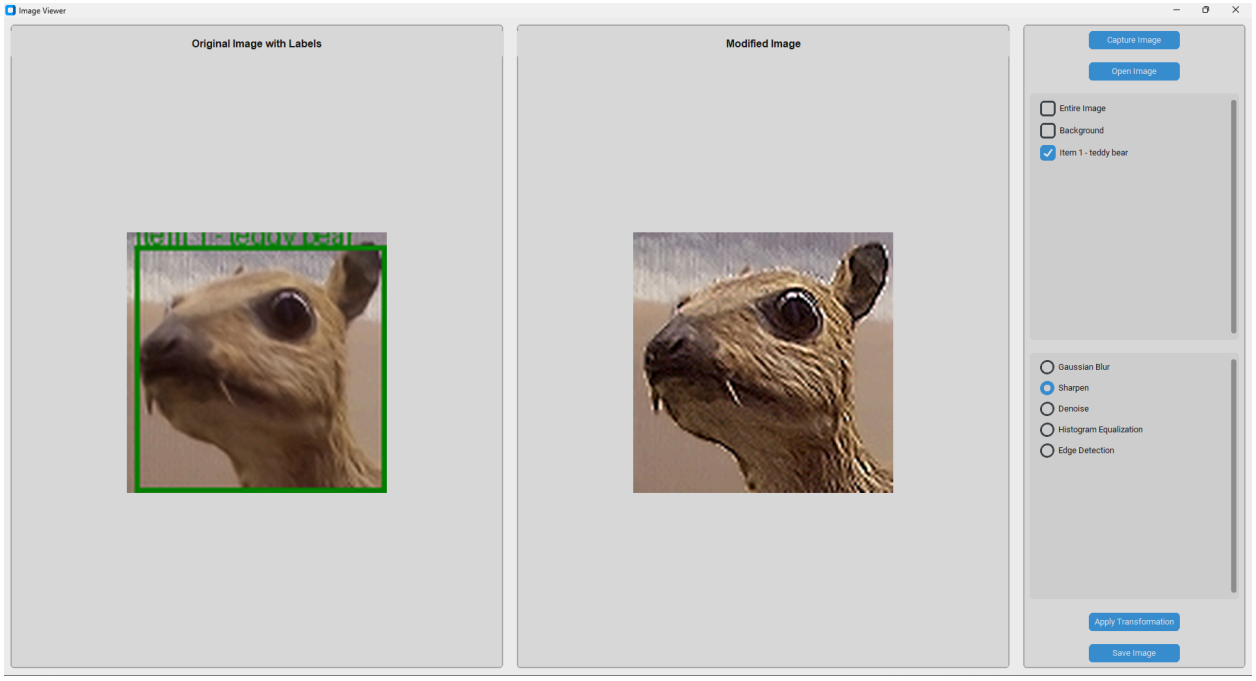

Figure 2: Sharpen

Denoising: In Figure 3 we can see that the original image has a significant amount of noise, with a grainy, “salt and pepper” appearance. Upon applying denoising, we can see that the less detailed parts of the image, such as the suit, are smoothed, and other features such as the hair, still retain most of their detail.

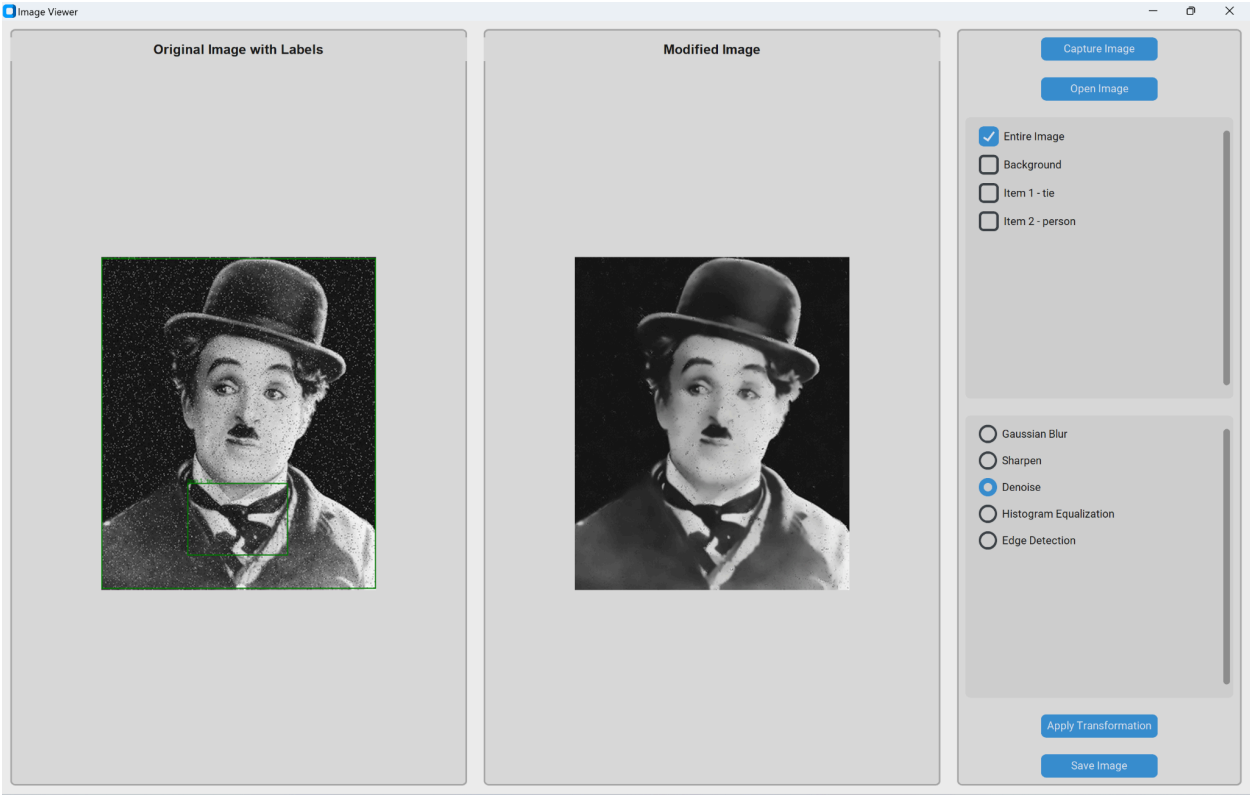

Figure 3: Denoising

Histogram Equalization: In Figure 4, we see how Histogram Equalization is used to improve the contrast of the image. In the original, the background has poor contrast and has a different intensity than the rest of the image. After applying the Histogram Equalization, it results in a more balanced intensity distribution, making the features of the image more visible and distinguishable. This is particularly useful in improving the visibility of objects in low-contrast images.

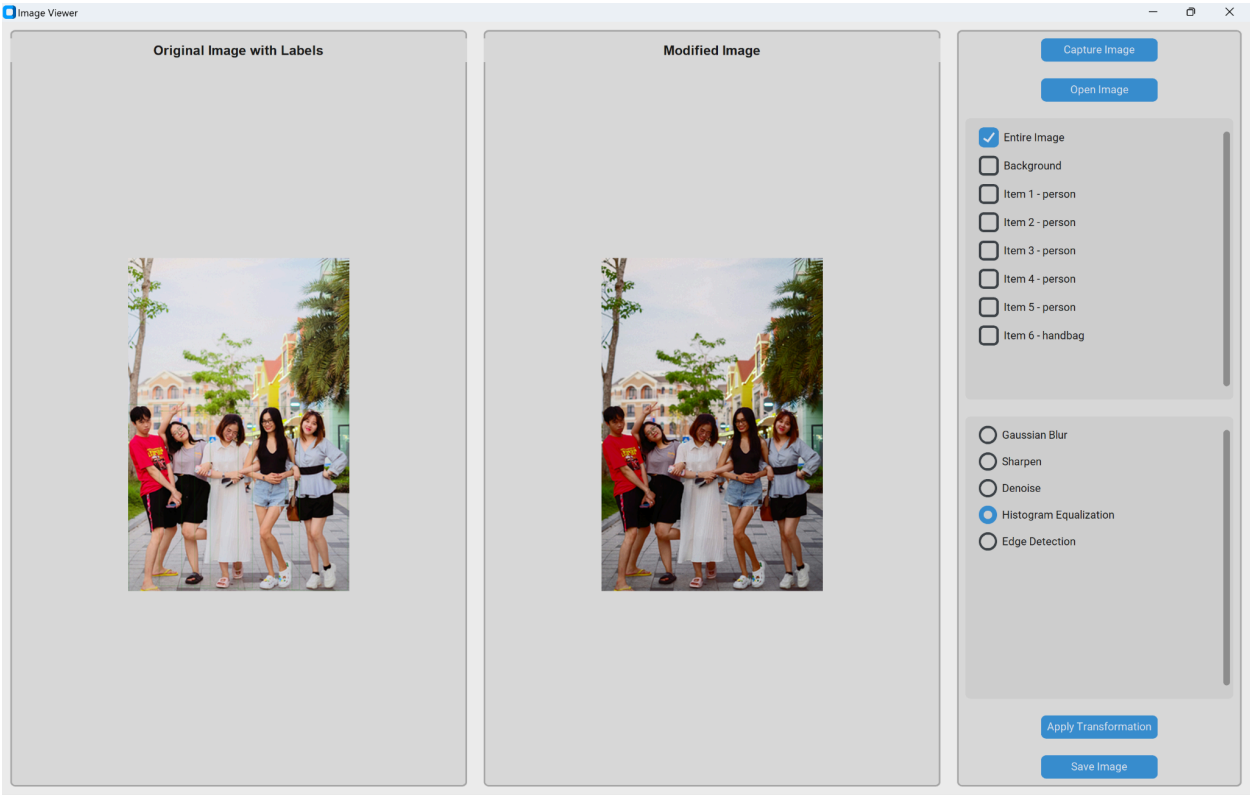

Figure 4: Histogram Equalization

Edge Detection: Figure 5 shows how the application carefully performs edge detection based on the image that is processed by the user. The edge-detected image clearly outlines the boundaries of objects, making them easy to recognize. This is beneficial for when the user wants to highlight specific segments of the object they have applied the edge detection on.

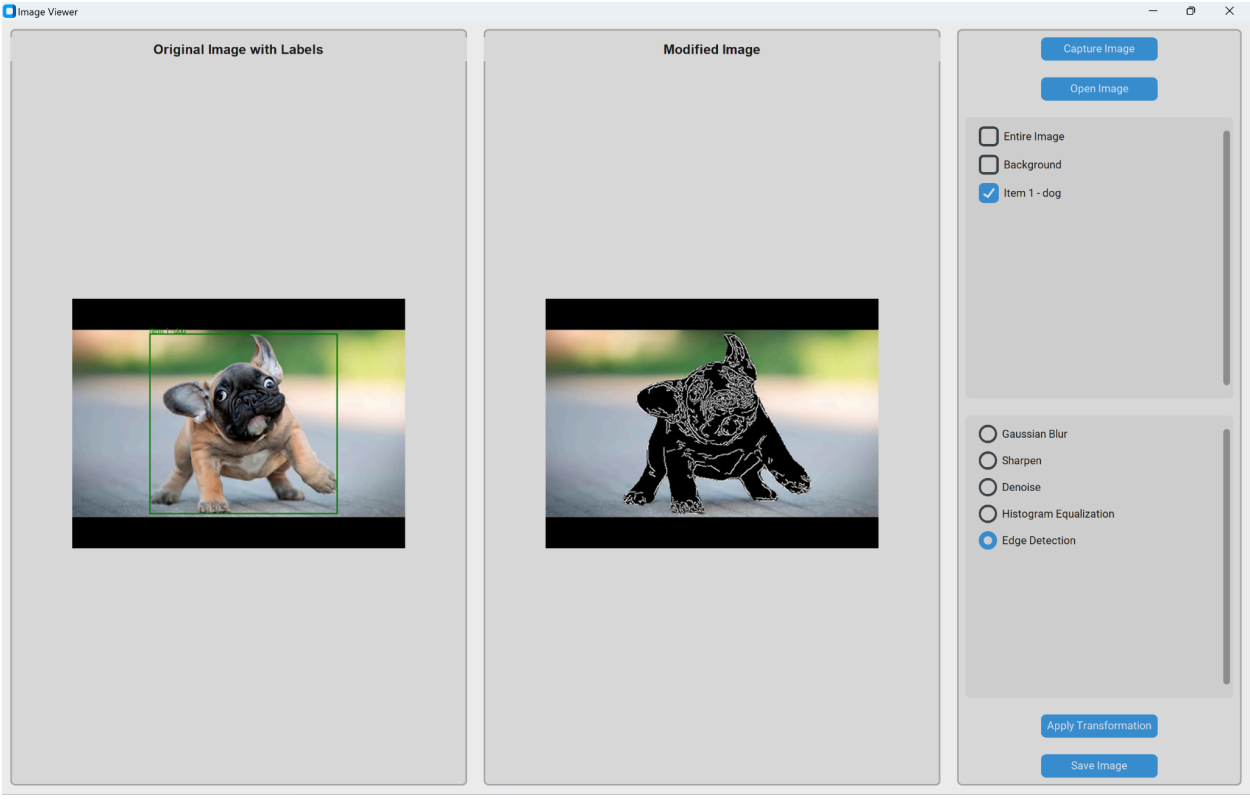

Figure 5: Edge Detection

Workload and Quality

The workload was evenly distributed, and the application has a good level of features for the 1-month timeframe. The frontend was created with custom-tkinter and is minimal, with an intuitive and simple-to-use layout, which can be seen in the Appendix. In terms of the actual technical aspects of the project, it accomplishes a good variety of tasks.

We use the YOLOv8 segmentation model for object detection and segmentation, allowing for the identification and masking various sections of the image which contain objects. The application processes images captured from a webcam or loaded from local files. The YOLOv8 model performs object detection and segmentation, generating bounding boxes, class labels, and precise segmentation masks for each detected item.

Techniques, such as erosion and dilation, are applied to refine these masks, ensuring accurate representation of the objects. These refined masks enable the use of transformations, such as Gaussian blur, sharpening, denoising, histogram equalization, and edge detection, to specific objects or regions within the image.

The application supports multiple transformations, allowing users to select and apply them to either the entire image or specific detected objects. The user interface provides a clear distinction between the original image with bounding boxes and labels and the modified image after transformations. Users can save the transformed images, making this tool practical for various applications in image processing and computer vision.

There are a few aspects we could improve in a future iteration of the project. The project has been created in a way that allows for the application to be easily extended to add additional filters and transformations for the image processes. We also used a general purpose YOLOv8 model, and could instead use one that is trained to specific data to make the project more specialized.

Overall, the project achieves a robust and flexible solution for object recognition and segmentation, demonstrating significant technical capability within the given timeframe. The features implemented provide a comprehensive tool for image analysis and manipulation, suitable for a variety of practical applications.

Discussion of Individual Contributions

The group was split into two teams of two. Elaine and Clarence worked on the UI development and figuring out how to use the CNN to identify the various objects being detected in the images as well as how to parse them accordingly. Then when the various items in the image were masked, they also learned how to refine these masks to minimize unnecessary pixels getting modified when the user applied the various DSP image processing techniques.

Gurnek and Manraj worked on the implementation of the various filters used to process the images. They also researched the algorithms behind them and learned how to effectively ensure they work as intended.

All four members worked collaboratively on both the report as well as presentation, and overall each member contributed approximately 25% of the content to the project as a whole.

Conclusion

In conclusion, convolutional neural networks (CNNs) have become the foundation of learning in image classification and recognition. CNNs versatility and effectiveness allows for the model to be applied to different real-world scenarios such as autonomous vehicles and security cameras.
The CNN architecture consists of the convolution, pooling and fully connected layer which enables automated feature extraction, leveraging concepts from many fields such as digital signal processing (DSP).

The scope of our project focuses on how we use DSP to analyze images. We utilize a pre-trained CNN model, YOLOv8, to identify objects within images. This model allows us to apply various image processing techniques and presents no difficulty upon utilization as the model simplifies the initial stages of image analysis. By employing filters such as Gaussian blur, sharpen, edge detection and noise reduction, further analysis of identified objects can be performed.

We are able to successfully incorporate CNNs in object identification with the detailed analysis provided by DSP techniques through this approach. This combination conveys how deep learning and DSP work together in modern technological applications.

Appendix

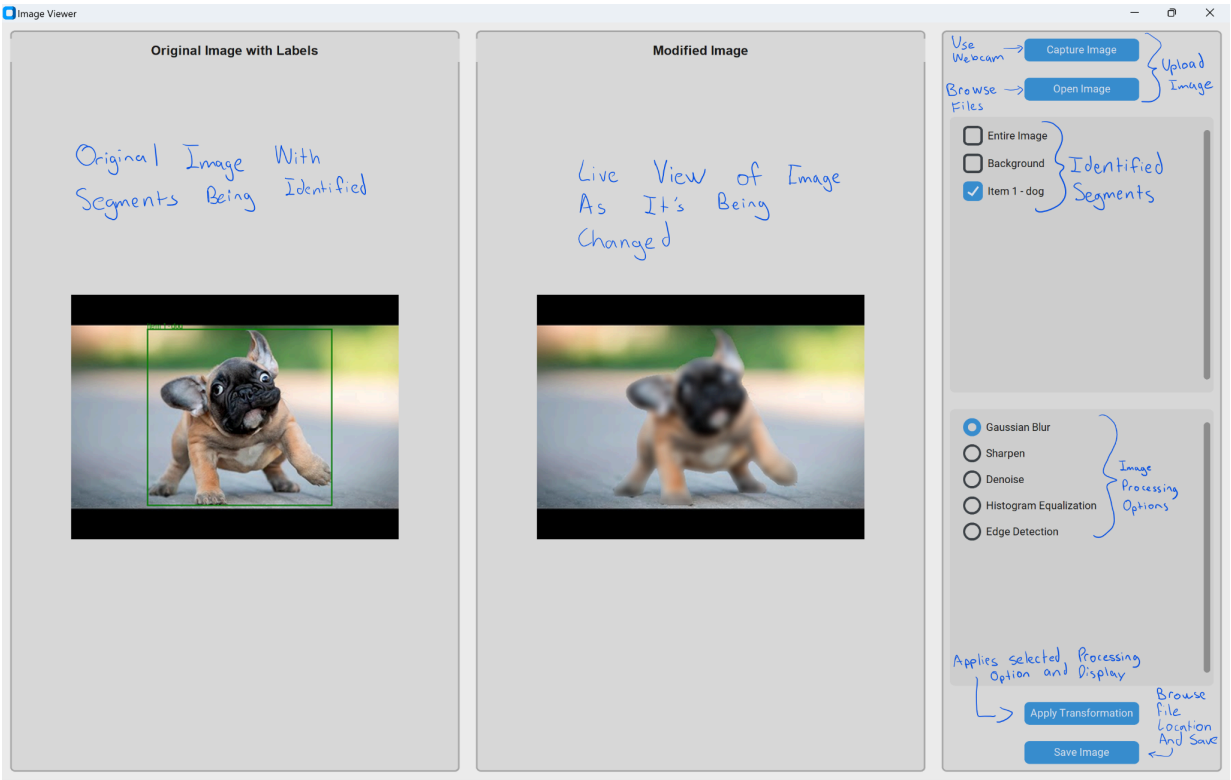

Figure 6: Labeled application

The application is easy and intuitive to use. The steps to use it are as follows:

1. Select the image you wish to process via local files or webcam
2. Allow the CNN to identify items in your image and to segment it
3. Select what item you wish to process.
4. Select how you would like to process it
    a. You can select one, or multiple options on any of the identified segments.
    b. Select “Apply Transformation”
    c. The updated image will be window in the center of the application
5. Click “Save Image”

References

1. J. Gu et al., “Recent advances in convolutional neural networks”, *Pattern Recognition*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013 [accessed July 18, 2024].

2. OpenCV, “Morphological Transformations”, *OpenCV-Python Tutorials*. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html [accessed July 25, 2024].
3. A. V. Oppenheim, R. W. Schafer, and J. R. Buck, "Discrete-Time Signal Processing," 2nd ed. Prentice-Hall, 1999 [accessed July 21, 2024].
4. R. Thakur and K. Khare, “High Speed FPGA Implementation of FIR Filter for DSP Applications,” *International Journal of Modeling and Optimization*, pp. 92–94, 2013, doi: https://doi.org/10.7763/ijmo.2013.v3.242 [accessed July 21, 2024].
5. R. C. Gonzalez and R. E. Woods, "Digital Image Processing," 4th ed. Pearson, 2018 [accessed July 21, 2024].
6. A. K. Jain, “Fundamentals of Digital Image Processing”, Pearson, 1989 [accessed July 21, 2024].
7. S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, 1997 [accessed July 21, 2024].
8. R. Boomgaard, “6.1 Gaussian Smoothing and Gaussian Derivatives”, *Image Processing and Computer Vision*. https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/LectureNotes/IP/LocalStructure/GaussianDerivatives.html [accessed July 24, 2024].
9. Udacity, USA. *Example Gaussian Filter.* (Feb. 23, 2015). Accessed: July 24, 2024. [Online Video]. Available: https://youtu.be/-AuwMJAqjJc?si=mTj3zaTYPujxNUJo
10. V. Powell, “Image Kernels”, *Explained Visually*. https://setosa.io/ev/image-kernels/ [accessed July 24, 2024].
11. R. Lini, “Laplacian of Gaussian Filter (LoG) for Image Processing”, *Medium*. https://medium.com/@rajilini/laplacian-of-gaussian-filter-log-for-image-processing-c2d1659d5d2#:~:text=The%20Laplacian%20of%20an%20image,edge%20detection%20and%20feature%20extraction. [accessed July 24, 2024].
12. G. Jocher, A. Vina, “Data Preprocessing Techniques for Annotated Computer Vision Data”, *Ultralytics*. https://docs.ultralytics.com/guides/preprocessing_annotated_data/ [accessed July 25, 2024].
13. J. Li, W. Daia, “Adaptive Image Denoising Algorithm Based on Correlativity”, *ScienceDirect.* https://www.sciencedirect.com/science/article/pii/S1877705811053562 [accessed July 12, 2024].