NutSure: Nut Detection and Classification Using YOLOv8

Collantes, Eric Jerard A.
BSCS-3A
Technological University of the Philippines – Manila
Manila City, Philippines
ericjerard.collantes@tup.edu.ph

Privado, Veronica Anne C.
BSCS-3A
Technological University of the Philippines – Manila
Manila City, Philippines
veronicaanne.privado@tup.edu.ph

***Abstract*– Traditional nut processing and classification methods are labor-intensive, time-consuming, and prone to human error, leading to poor quality control, mislabeled products, compromised food safety, and potential regulatory violations. NutSure addresses these issues by utilizing artificial intelligence (AI), data analytics, and computer vision techniques to detect and classify unshelled nuts from images or real-time video capture. By leveraging deep learning algorithms and models like the YOLOv8 and Convolutional Neural Networks (CNNs), NutSure achieves exceptional results and performance in detecting and classifying unshelled nuts. This project utilizes a preprocessed dataset from Kaggle, which minimizes the need for aggressive data augmentation and preprocessing. YOLOv8 helps by doing computations efficiently while still delivering reliable and accurate results for nut detections and classifications across five classes: Almond, Cashew, Peanut, Pistachio, and Walnut. This project includes a Python-based application that utilizes the library Tkinter for the application's graphical user interface (GUI). This application enables the users to upload their images or use their webcam for real-time unshelled nut detection and classification. The trained model shows exceptional performance, achieving a mean Average Precision (mAP50) of 92.1%, with high precision and recall metrics.**

NutSure not only addresses the gaps in existing nut processing systems but also opens possibilities for further enhancements, including real-time sorting, quality assessment, and broader applications in the food industry.

Keywords— CNN, nut detection, nut classification, YOLOv8, nuts, deep learning, object detection, computer vision, real-time classification.

I. INTRODUCTION

As demand for healthier foods continues to grow, nuts have become increasingly popular as a nutritious snack option, offering plant-based protein, anti-oxidants, vitamins, and a variety of culinary uses [1][2]. Among the many varieties, peanut, almond, walnut, cashew, and pistachio are the most popular for their health benefits and versatility [2]. However, these nut classes show subtle distinctions in shape, texture, and color, which pose challenges for existing computer vision datasets and automated classification systems. Classifying nuts efficiently is a challenge in the food industry, where identifying and sorting various types of nuts are important to ensure specific benefits, quality control, accurate labeling, allergen control, and regulatory compliance. [3]. The traditional methods for classifying nuts such as manual sorting by workers or mechanical systems using vibrating sieves, or weight-based sorting are labor-intensive, time consuming, and prone to errors, especially with nuts that have similar visual features like size, weight, shape, and color [1]. These can lead to mislabeled products, compromised food safety, and potential regulatory violations. [4].

In response to these challenges, an innovative solution that is also durable and scalable is needed. Deep learning, through convolutional neural networks (CNN), can automate the classification process with high accuracy, consistency, and speed. It is known for its advanced image processing, pattern recognition, and computer vision [5]. By using convolutional neural networks (CNN), limitations of traditional methods can be overcome as patterns and differences between nut types can be identified [5]. In addition, YOLOv8, a widely used deep learning framework, provides the tools to develop efficient models for object

detection and automating classification with high accuracy and speed [6], which is an ideal choice for this NutSure nut classification project.

The researchers have focused on developing an AI model using YOLOv8 to classify 5 nut types under different conditions. The researchers sourced a dataset of preprocessed unshelled nut images from Kaggle, manually annotated, and used it for training the model. The training process was conducted locally using PyCharm, where Python scripts optimized the convolutional neural network architecture. The system is designed to detect the nut, analyze, and classify it through convolutional neural networks. By automating the classification process, the system seeks to enhance efficiency, quality, and consistency, addressing the limitations and challenges of traditional methods and offering a practical solution for the food industry.

II. BACKGROUND OF THE STUDY

A. Nut Detection and Classification

Nuts are an excellent source of antioxidants, vitamins, and protein that can be beneficial to human health. As people become more obsessed with a healthier lifestyle today, nuts have been in demand and with it, is the increase in production [1][2]. Processing nuts are time-consuming and labor-intensive, from ensuring that they are sorted accurately, to packaging. In addition, it is prone to error that can lead to inconsistency of the quality [1]. Food industry is now using modern technology, and with the increasing demand for nuts [7], the need for efficient and accurate methods to classify nuts has become crucial.

Combining the modern technology used in the food industry with artificial intelligence and computer vision provides an innovative solution for NutSure nut classification. By using the appropriate algorithms and machine learning models, NutSure nut classification can be improved in terms of accuracy, efficiency, consistency, and quality. This project makes use of deep learning to address the gaps in the current nut classification system, to improve quality control, accurate labeling, allergen control, and regulatory compliance.

B. Convolutional Neural Networks (CNN)

Convolutional Neural Network (CNNs) is a deep learning advanced algorithm that is well-known for its capabilities and background with tasks such as image classification, detection, and segmentation [8]. CNN is composed of multiple layers; convolutional, pooling, and fully connected, which all contribute a unique function to the CNN [9][10]. These layers extract features of the images, down sample spatial dimensions, process output feature maps and connect every neuron from the previous layer to the next layer [10] which is for processing grid-like data.

CNN has already made contributions and practical applications in the real world such as autonomous driving, surveillance systems, disease diagnosis, tumor detection, and biometric authentication [10]. The ability of CNN to analyze visual data through automatically learning features such as edges, textures, and shapes from images makes it highly effective for distinguishing objects with subtle differences [10], such as nut classes.

C. Kaggle

Kaggle is a web-based platform that offers high-quality datasets and a collaborative environment that can be used for data science or machine learning projects [11]. In this project, the dataset was gotten from Kaggle and the unshelled nut images were already preprocessed and cleaned. In addition, the dataset also consisted of images that were augmented already, which lessened the need for aggressive data augmentation.

Since Kaggle is widely known by many users, many are contributing and providing a wide range of preprocessed datasets. The accessibility and availability of these datasets help in efficiently developing models that give accurate results [11]. By utilizing Kaggle and all its features, the development of the project became more efficient and yielded accurate and effective results.

D. Object Detection

Object detection is a computer vision used for identifying what and where certain objects are in a data such as image or video [12]. This project is focused on nut classification, however, object detection can be useful in the future changes and improvements. This can help in the future if intended to integrate real-time sorting or analysis of nuts, and make this project more useful and versatile. By integrating object detection, this project can address more problems outside classifying and detecting classes of nuts, increasing its contribution in the food industry.

E. YOLOv8

YOLOv8 or You Only Look Once algorithm is a deep learning model that utilizes both computer vision and machine learning to detect and classify objects [13]. It includes a Python package and a CLI-based implementation, which makes it very accessible and easy to use. YOLOv8 is the result of improvements and changes in its previous versions which makes it faster and more accurate. Moreover, it has a better reputation in handling multiple tasks compared to its previous versions. With these improvements, it has become efficient with the usage of resources as it balances accuracy and computational load [14].

In YOLOv8, different model sizes cater to different needs. There is YOLOv8n for nano, YOLOv8s for small, YOLOv8m for medium, YOLOv8l for large, and YOLOv8x for extra-large. The model size has an inverse relationship with inference time and a direct relationship with mean Average Precision (mAP). The dataset for the unshelled nut categorization in this study was trained using the YOLOv8n model [15].

F. Python

Python programming language is simple, versatile, and has extensive libraries which is the reason it is used in many projects [16][17]. It has been useful since its first version until its latest version, but it has become powerful and useful in various domains including image classification [16]. The versatility of Python contributed significantly to the efficient development, training, and testing of the nut classification model.

Many libraries from Python such as ultralytics, tkinter, cv2, labelimg, scikit-learn are helpful in classification projects [18], making it possible for this project to classify unshelled nuts accurately. It is seamless with many deep learning models, and its seamless integration with YOLOv8 also contributed in the development of deep learning models of nut classification to be efficient [19].

III. REVIEW OF RELATED LITERATURE

Convolutional Neural Networks (CNNs) have significantly advanced image classification tasks within the food industry, particularly in food classification. Their ability to automatically extract hierarchical features [5][10] from images has led to improved accuracy in identifying various food items, enhancing quality assessment and inventory management processes. For instance, CNNs have been effectively applied in classifying fruits and vegetables, contributing to more efficient food production and distribution systems [20].

The introduction of the You Only Look Once (YOLOv8) object detection architecture has further changed image classification and object detection tasks. YOLOv8 combines speed and accuracy, making it suitable for real-time applications in the food industry. Studies have demonstrated its efficacy in food classification and quality control, where YOLOv8's efficient detection framework enables reliable performance in identifying multiple object classes simultaneously [13][21]. Its versatility has proven effective across diverse datasets, which makes it an ideal choice for food industry applications.

However, challenges persist in classifying visually similar objects, such as different types of nuts, due to subtle variations in shape, texture, and color. YOLOv8 addresses these challenges by using advanced anchor-free techniques, improved loss functions, and a dynamic feature pyramid structure [6][13]. These enhancements allow the model to focus on small but critical differences, improving its ability to differentiate between similar classes. Despite these advantages, the development of more specialized datasets remains essential for further enhancing classification accuracy in real-world scenarios [22].

Recent advancements in fine-grained object detection using YOLOv8 have introduced techniques such as semantic part localization and multi-scale feature extraction. These methods improve performance in tasks requiring precise identification of subtle visual differences. For nut classification, YOLOv8's fine-grained capabilities can be further enhanced by integrating clustering techniques and adaptive thresholds, ensuring better generalization and accuracy in diverse environments [23].

Moreover, the combination of YOLOv8 with multimodal data sources, such as hyperspectral or textural data, has demonstrated significant promise in agricultural applications. By incorporating diverse data, YOLOv8 models can better capture intricate differences between objects, enabling more reliable and accurate classification systems. This approach offers new opportunities for improving automated nut classification systems [24].

While YOLOv8 has substantially advanced object detection and classification tasks in agriculture, ongoing efforts to improve dataset quality and refine classification techniques remain critical. By addressing these challenges,

YOLOv8 can continue to serve as a foundation for developing reliable and efficient automated classification systems in the food industry.

IV. METHODOLOGY/EXPERIMENTS

This part outlines methods used in training, evaluating, and implementing the NutSure nut classification model. It includes collection, pre-processing, model training, model integration into application, and application development

A. Data Collection

The dataset for NutSure was sourced from Kaggle and Google. **1,000 standalone unshelled nut images** were obtained from Kaggle and **250 clustered unshelled nut images** were from Google.

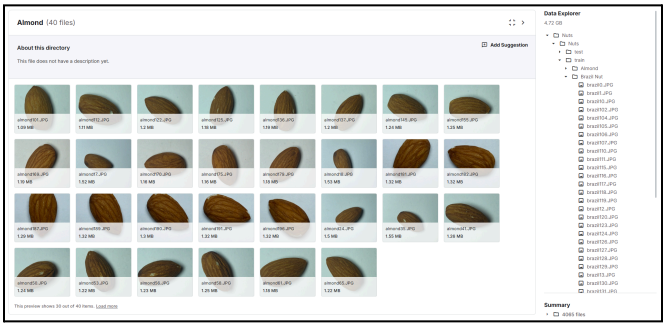

Fig. 1. Dataset from Kaggle

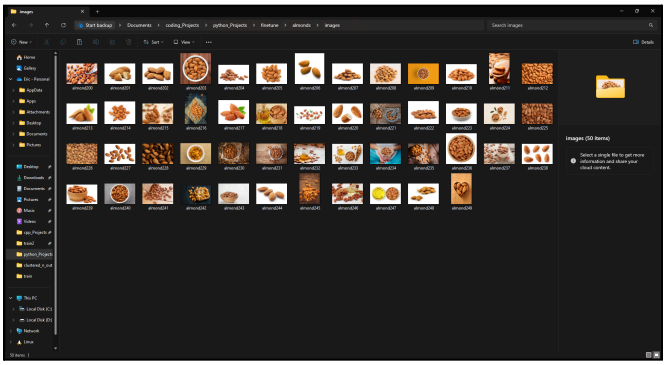

Fig. 2. Images sourced from google

The dataset comprises **1,250 images** of unshelled nuts, classified into 5 classes: almond, cashew, peanut, pistachio, and walnut. There are 200 images for each class from Kaggle and additional 50 images per class from Google.

B. Data Pre-processing

1. ***Renaming Images:*** The dataset from Kaggle with 1000 images were already labeled however, the 200 images manually downloaded from Google were not. The researchers made and used a Python script for automation of renaming images.

```python
import os

labels_dir = "finetune/labels"  # labels dir
keyword = "walnut"  # prefix
correct_class_number = 4  # class number for this class (e.g., peanut = 3)

for file_name in os.listdir(labels_dir):
    if file_name.startswith(keyword) and file_name.endswith(".jpg"):
        file_path = os.path.join(labels_dir, file_name)

        with open(file_path, "r") as file:
            lines = file.readlines()

        updated_lines = []
        for line in lines:
            parts = line.strip().split()
            if len(parts) > 0:
                parts[0] = str(correct_class_number)  # change prefix of file
                updated_lines.append(" ".join(parts))

        with open(file_path, "w") as file:
            file.write("\n".join(updated_lines) + "\n")

        print(f"Updated class numbers in: {file_name}")

print("Class number updates completed!")
```

Fig. 3. Python script for renaming images

2. ***Data Annotation:*** All of the 1,250 images, both from Kaggle and Google were annotated using labelImg, a tool from Python. The researchers boundboxed each unshelled nut in the image, labelling them with the correct class - almond, cashew, peanut, pistachio, and walnut. This step helped in creating a well-defined database for the model.

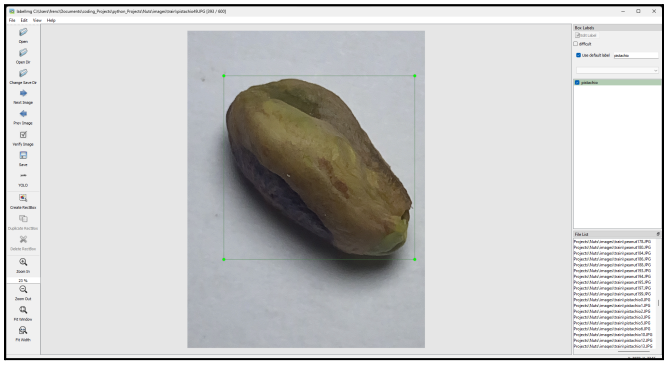

Fig. 4. Data annotation of pistachio using labelImg

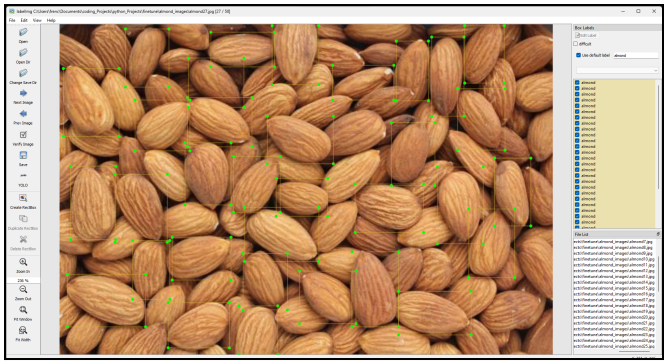

Fig. 5. Data annotation of almonds using labelImg

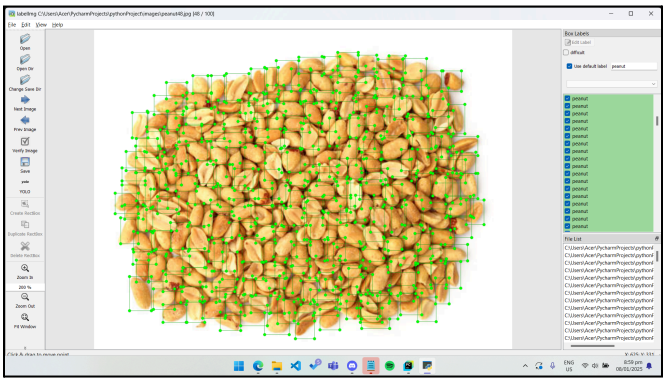

Fig. 6. Data annotation of peanuts using labelImg

While annotating the images, there is a text file created for every label (nut class) that contains the coordinates of the center of the bounding box. The height and width of the bounding box relative to the total image is outputted. The values inside this text file are crucial during the machine learning process.

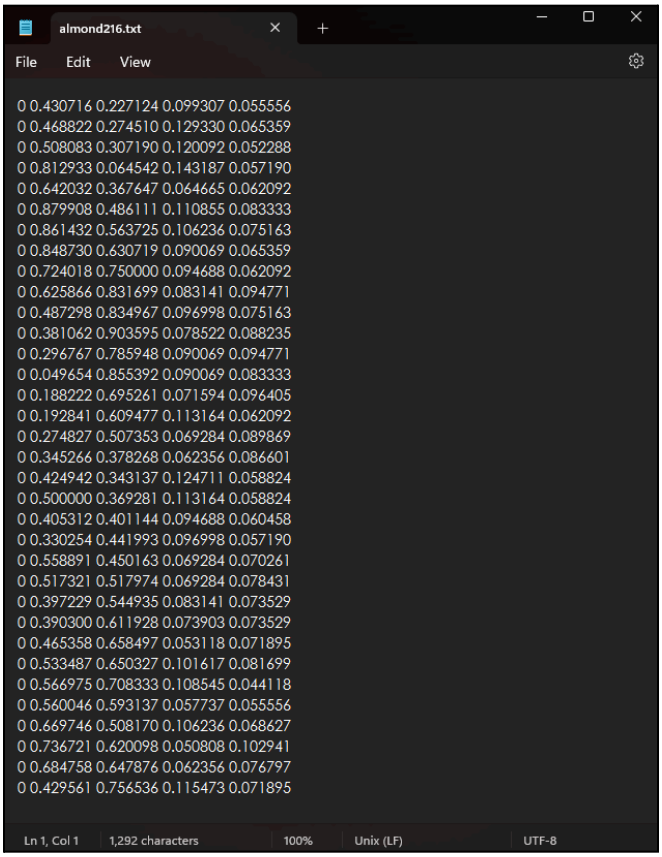

Fig. 7. Text file containing class and bounding box values for an image

3. ***Dataset Split:*** The annotated images were divided for training, validation, and test. The distribution was as follows:

	TRAIN	VALIDATION	TEST
KAGGLE (1,000 images)	600	200	200
GOOGLE (250 images)	150	50	50

Fig. 8. Dataset Split

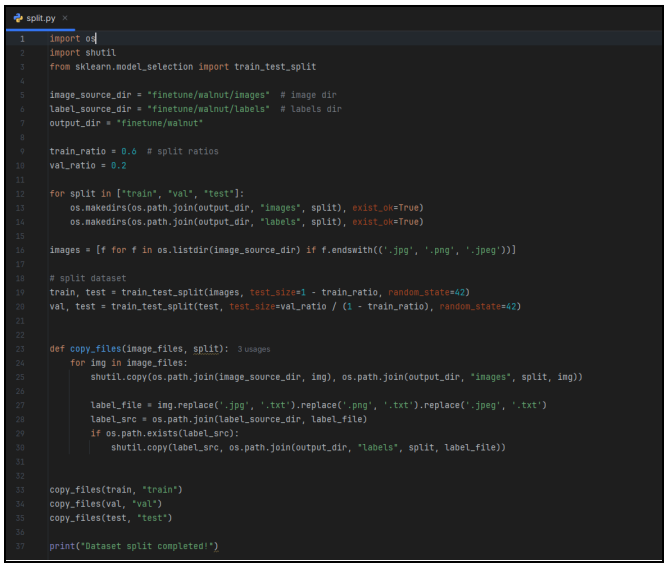

Fig. 9. Python code for splitting dataset

The 1,000 images from Kaggle were distributed with 120 images per class for training, with a total of 600 images. For validation, there were 40 images, with a total of 200 images. And for the test, there were 40 images, with a total of 200 images.

Meanwhile, the 250 images sourced from Google were distributed with 30 images per class for training, with a total of 150 images. For validation there were 10 images per class, a total of 50 images. And for the test, there are 10 images per class and a total of 50 images as well.

For the overall dataset, 60% (750 images) of the dataset were for training, 20% (250 images) were for validation, and the last 20% (250 images) were for testing.

C. Model Training

The model training for NutSure was conducted in **PyCharm** using the **YOLOv8 framework.** YOLOv8 was specifically chosen for the NutSure because of its reputation in object detection and image classification [6].

data.yaml

```yaml
train: images/train
val: images/val
test: images/test

nc: 5  # number of classes
names: ['Almond', 'Cashew', 'Peanut', 'Pistachio', 'Walnut']
```

Fig. 10. data.yaml

The **data.yaml** file is used to specify important configuration details for training and validating the model. It included the file paths for training, validation, test datasets, and classes.

In training of this model, the pre-trained **yolov8n.pt model** was used by the researchers.

yoloV8.py

```python
from ultralytics import YOLO

# pre-trained YOLOv8n model
model = YOLO('yolov8n.pt')

model.train(
    data='data.yaml',
    epochs=50,
    imgsz=640,
    batch=16,
    workers=16
)
```

Fig. 11. Python code in PyCharm for model training

The model training begins by running the Python script in Figure 11. First, it was run using only the Kaggle dataset.

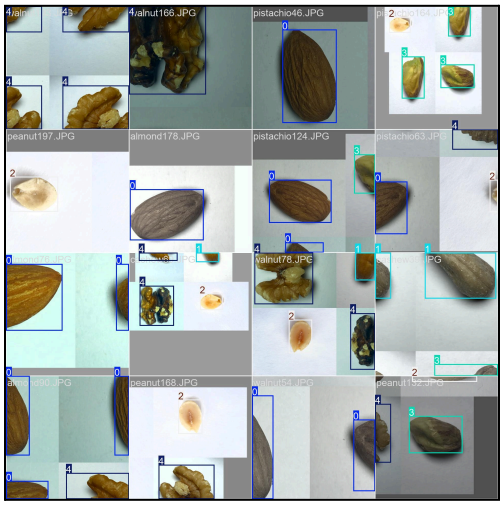

Fig. 12. The training batch using the Kaggle dataset

After the training was done, the researchers found that the model has exceptional accuracy and precision. However, when it was tested using images that have clustered nuts from Google, the model did not predict accurately.

Fig. 13. The model prediction on images outside the dataset

The researchers then decided to add more images in the dataset, specifically images that have clustered unshelled nuts. After gathering, labelling, and annotating the Google images, the previous model was loaded and then trained again using a dataset which now includes both the Kaggle and Google images.

yoloV8.py

```python
from ultralytics import YOLO

# first model from the Kaggle dataset
model = YOLO('runs/detect/train/weights/best.pt')

model.train(
    data='data.yaml',
    epochs=50,
    imgsz=640,
    batch=16,
    workers=16
)
```

Fig. 14. Second training using the previously trained model and the Google images

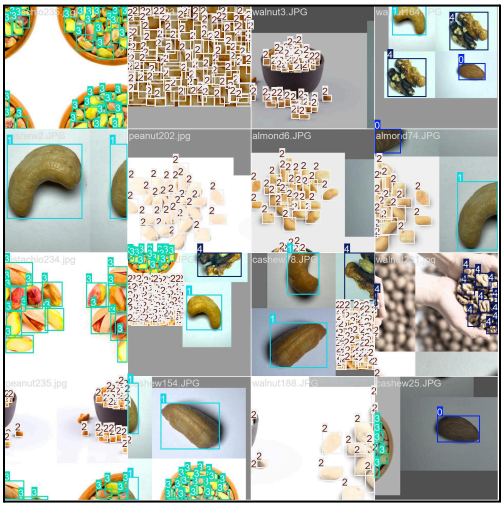

Fig. 15. The training batch using both the Kaggle and Google dataset

D. Application Development

The NutSure application was developed using Python and its libraries OpenCV and TKinter. The OpenCV was used for the camera features of the application. It helped with the real-time image processing and object detection. Meanwhile, the TKinter was used for the graphical user interface of the NutSure application.

testing.py

```
import tkinter as tk
from tkinter import filedialog, Label, Button, Frame
from PIL import Image, ImageTk
from ultralytics import YOLO
import cv2
from threading import Thread

model = YOLO('runs/detect/train2/weights/best.pt')  # model path

# Global variables
camera_on = False
cap = None


def load_image():  1 usage
    global img, img_path
    img_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
    if img_path:
        img = Image.open(img_path)
        img.thumbnail((550, 550))
        img_tk = ImageTk.PhotoImage(img)
        image_label.config(image=img_tk)
        image_label.image = img_tk
        status_label.config(text="Image loaded. Ready for prediction.")


def predict_image():  1 usage
    global img_path
    if img_path:
        results = model.predict(source=img_path, conf=0.7, save=False)
        annotated_img = results[0].plot()
        num_detections = len(results[0].boxes)

        annotated_img = cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB)
        annotated_img = Image.fromarray(annotated_img)
        annotated_img.thumbnail((550, 550))
        img_tk = ImageTk.PhotoImage(annotated_img)
        image_label.config(image=img_tk)
        image_label.image = img_tk
```

Fig. 16.1. Code for the NutSure application

testing.py

```
def start_camera():  1 usage
    global camera_on, cap
    camera_on = True
    cap = cv2.VideoCapture(0)

    def process_camera():
        while camera_on:
            ret, frame = cap.read()
            if not ret:
                break

            results = model.predict(source=frame, conf=0.5, save=False)
            annotated_frame = results[0].plot()
            num_detections = len(results[0].boxes)

            annotated_frame = cv2.cvtColor(annotated_frame, cv2.COLOR_BGR2RGB)
            frame_pil = Image.fromarray(annotated_frame)
            frame_pil.thumbnail((550, 550))
            frame_tk = ImageTk.PhotoImage(frame_pil)

            image_label.config(image=frame_tk)
            image_label.image = frame_tk

            if num_detections > 0:
                status_label.config(text=f"Live: {num_detections} detections found.")
            else:
                status_label.config(text="Live: No detections found.")

        cap.release()

    Thread(target=process_camera).start()


def stop_camera():  1 usage
    global camera_on, cap
    camera_on = False
    if cap is not None and cap.isOpened():
        cap.release()
    status_label.config(text="Camera stopped.")
```

Fig. 16.2. Code for the NutSure application

E. Model Implementation in Python

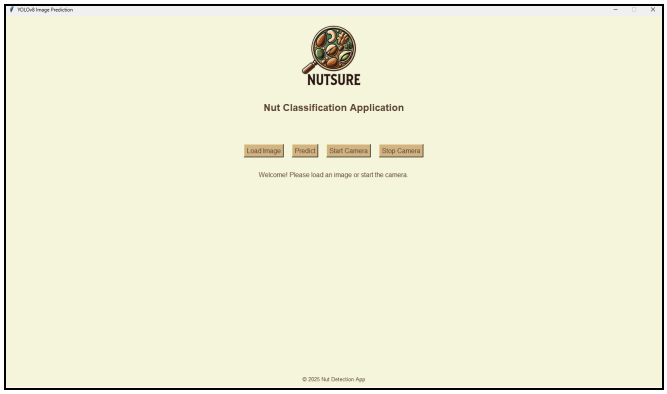

Fig. 17. NutSure main window

The NutSure application has two features: **detection from uploaded images** and **real-time object detection**. The detection from uploaded images or "load image" feature will let users upload images from their device and the application will detect the unshelled nuts in the image. Meanwhile, the real-time object detection or "start camera" feature uses a webcam to detect and classify what type of nut it is.

Fig. 18. Classifying unshelled nuts using the 'load' feature

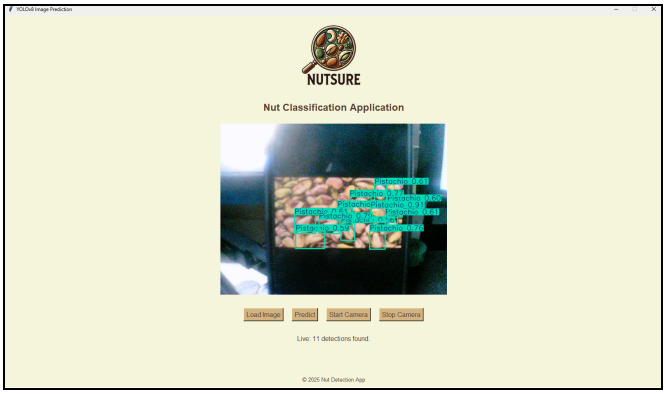

Fig. 19. Classifying unshelled nuts using the 'camera' feature

V. RESULTS

A. *Training results*

1. Kaggle Dataset

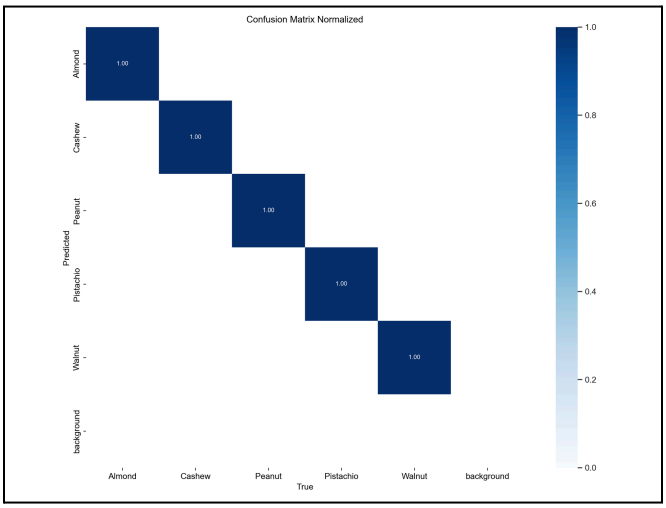

Fig. 20. Training Set Confusion Matrix for Kaggle Dataset

The model reached an accuracy **of 99.5%** and hit its peak performance at the 50th epoch for the training datasets for the standalone unshelled nut images (Kaggle dataset). This is highlighted in Figure 20.

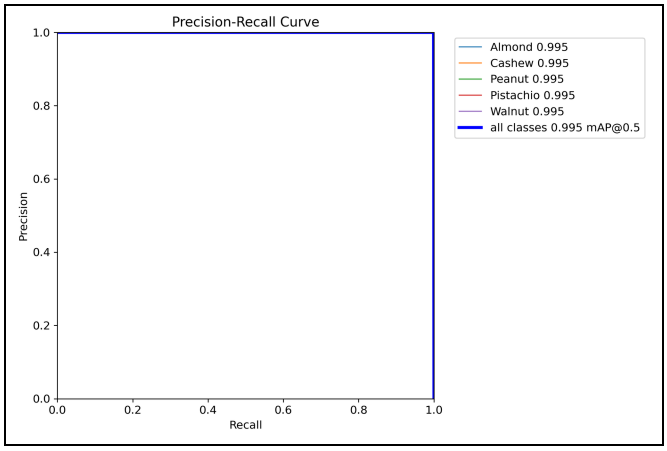

Fig. 21. Precision - Recall Curve for Kaggle Dataset

Precision - Recall Curve for Kaggle Dataset: The model achieved a mean Average Precision (mAP@0.5) of **99.5%**, demonstrating exceptional accuracy with minimal false negatives across all nut classes in the Kaggle training dataset.

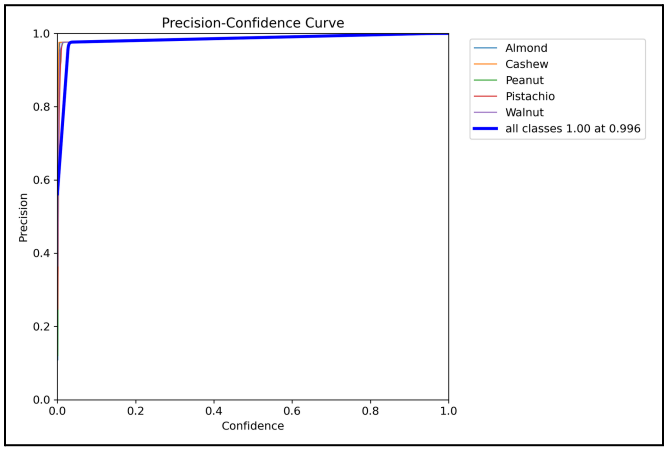

Fig. 22. Precision - Confidence Curve for Kaggle Dataset

Precision - Confidence Curve for Kaggle Dataset: The model achieved a near-perfect precision of **1.00** at a confidence threshold of **0.996**, demonstrating its ability to make highly accurate predictions with very few false positives across all nut classes in the Kaggle training dataset.

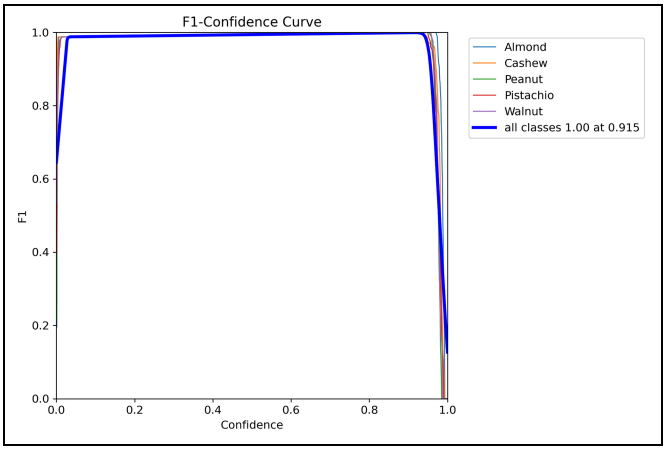

Fig. 23. F1-Confidence Curve for Kaggle Dataset

F1 - Confidence Curve for Kaggle Dataset: The model achieved an average F1 score of **1.00** at a confidence threshold of **0.915**, as shown in the figure. This indicates perfect precision and recall across all nut classes in the Kaggle training dataset, demonstrating the model's exceptional performance and reliability in classification tasks.

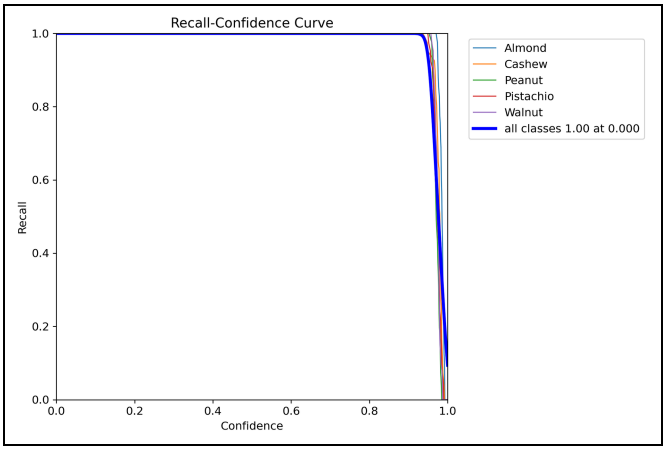

Fig. 24. Recall-Confidence Curve for Kaggle Dataset

Recall - Confidence Curve for Kaggle Dataset: The model achieved a recall of **1.00** across all nut classes (Almond, Cashew, Peanut, Pistachio, and Walnut) in the Kaggle training dataset at a confidence threshold of **0.000**. This indicates that the model successfully identifies all relevant instances of each class, ensuring no false negatives at the lowest confidence level.

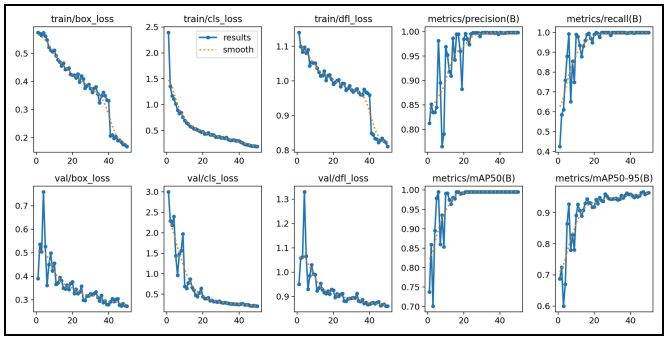

Fig. 25. Training and Validation Metrics for Kaggle Dataset

Training and Validation Metrics for Kaggle Dataset: The figure shows the model's performance over 50 epochs. Training and validation losses (box, classification, and DFL) decrease steadily, indicating improved accuracy in localization and classification. Precision, recall, and mAP@50 stabilize near **1.00**, demonstrating the model's exceptional ability to detect and classify unshelled nuts (Almond, Cashew, Peanut, Pistachio, and Walnut) with high reliability on the Kaggle dataset.

2. Overall dataset

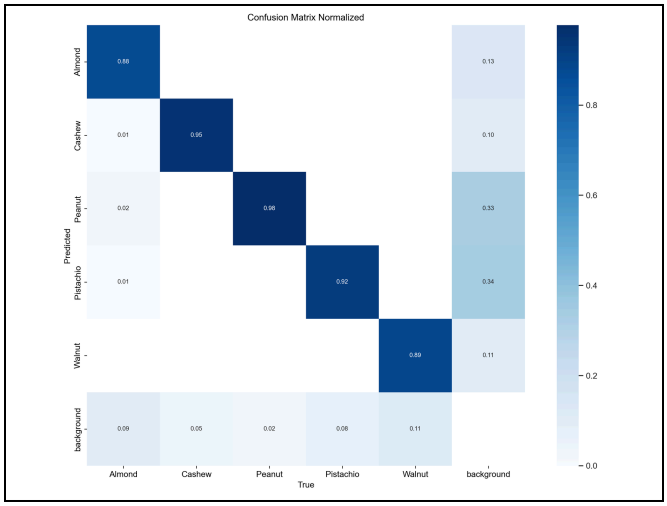

Fig. 26. Validation Set Confusion Matrix for Overall Dataset

The model achieved an overall accuracy of **93.5%**, with its peak performance observed at the **50th** epoch for both standalone and clustered unshelled nut images in the Kaggle dataset. This is shown in Figure 26, which shows the normalized confusion matrix. The matrix highlights the model's ability to accurately classify nut types, while

effectively differentiating them from the background, despite minor misclassifications.

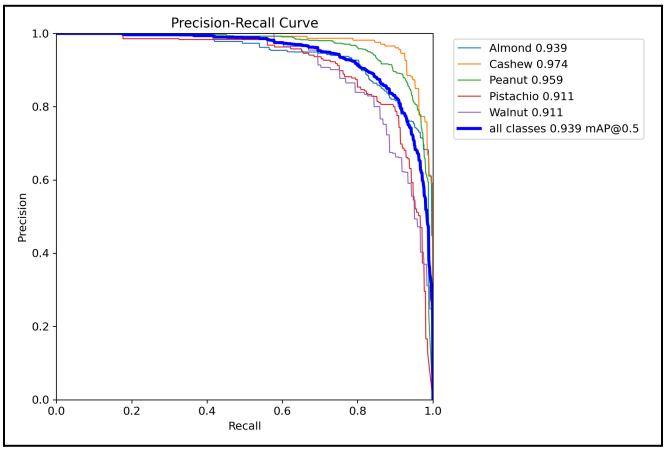

Fig. 27. Precision - Recall Curve for Overall Dataset

Precision-Recall Curve for Overall Dataset: The model achieved a mean Average Precision (mAP@0.5) of **93.9%** across all classes in the overall dataset, including standalone and clustered unshelled nut images. Class-specific performance shows Cashew with the highest mAP of **97.4%**, followed by Peanut at **95.9%**, while Pistachio and Walnut had slightly lower mAP values of **91.1%**. These results demonstrate the model's strong ability to maintain a balance between precision and recall across various input scenarios.

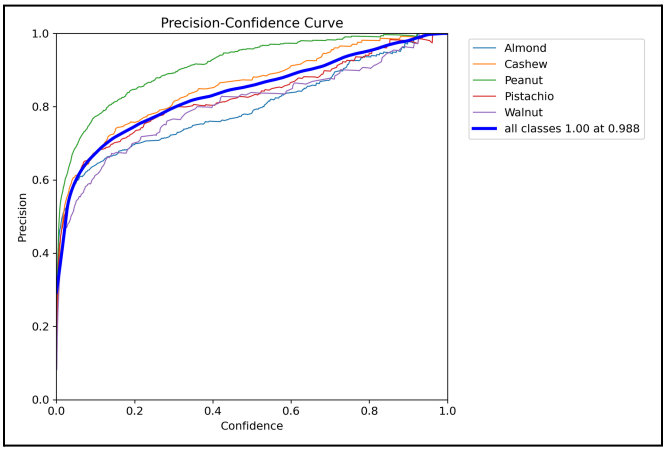

Fig. 28. Precision - Confidence Curve for Overall Dataset

Precision-Confidence Curve for Overall Dataset: The model achieved a precision of **100%** at a confidence threshold of **98.8%** across all nut classes. This indicates highly accurate predictions with minimal false positives in the overall dataset, including standalone and clustered unshelled nut images.

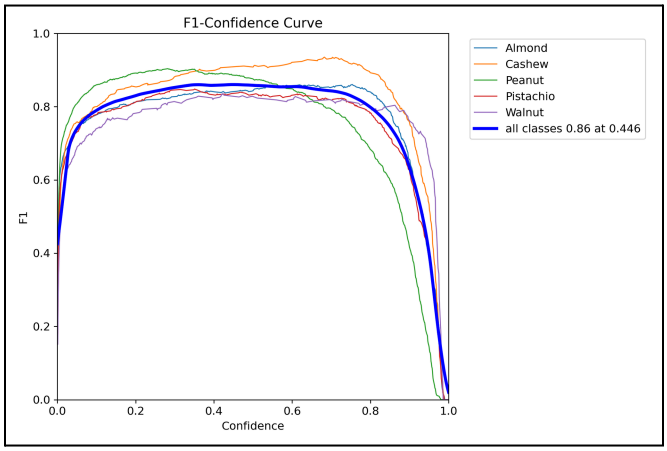

Fig. 29. F1-Confidence Curve for Overall Dataset

F1-Confidence Curve: The model achieved its highest F1 score of **86%** at a confidence threshold of **44.6%** across all classes in the **overall dataset**. This demonstrates a balanced trade-off between precision and recall for both standalone and clustered unshelled nut images.

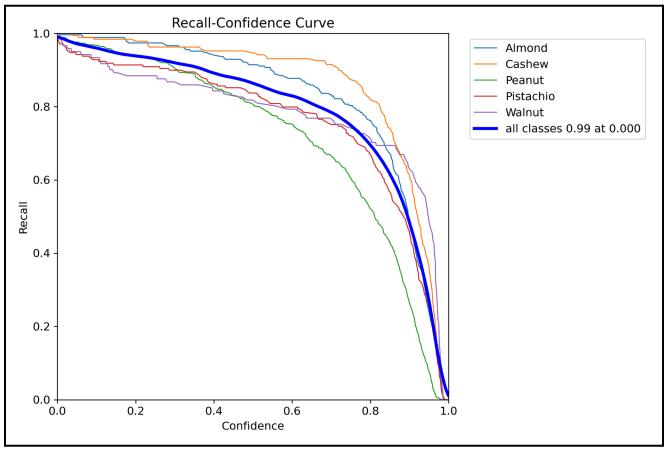

Fig. 30. Recall-Confidence Curve for Overall Dataset

Recall-Confidence Curve: The model attained a recall of **99%** at a confidence threshold of **0%**, indicating it successfully identified nearly all relevant instances across all nut classes in the overall dataset.

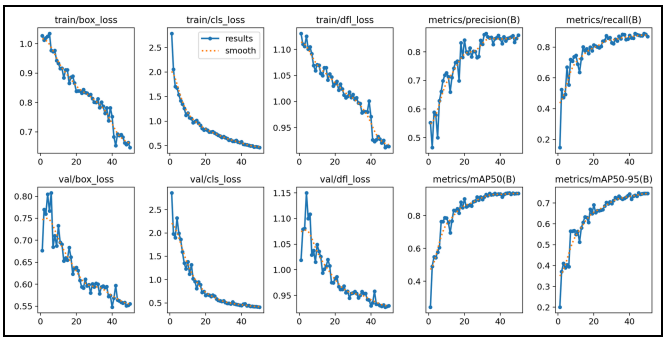

Fig. 31. Training and Validation Metrics for Overall Dataset

Training and Validation Metrics for Overall Dataset: The training and validation losses (box, classification, and DFL) steadily decreased over 50 epochs, indicating improved accuracy in localization and classification. Precision, recall, and mAP consistently increased, with mAP stabilizing near **1.00** for the IoU threshold of 0.50. This demonstrates the model's robust performance and generalization ability across the **overall dataset**, including both standalone and clustered unshelled nut images.

B. Testing results

1. Kaggle dataset

Testing accuracy: The model achieved an exceptional 99.5% accuracy on the test dataset from Kaggle. In addition, every class also achieved an almost perfect precision at 99.5% accuracy..

Key Metrics

Metric	Value
Box(P) (Precision)	0.998 (99.8%)
R (Recall)	1.0 (100%)
mAP50	0.995 (99.5%)
mAP50-95	0.963 (96.3%)

Fig. 32. General metrics for the Kaggle test dataset

Class-Specific Metrics

Class	Precision	Recall	mAP50	mAP50-95
Almond	0.997	1.0	0.995	0.926
Cashew	1.0	1.0	0.995	0.949
Peanut	0.997	1.0	0.995	0.957
Pistachio	0.997	1.0	0.995	0.988
Walnut	0.997	1.0	0.995	0.995

Fig. 33. Class-specific metrics for the Kaggle test dataset

2. Overall dataset

Testing accuracy: The second model achieved a 92.1% accuracy on the overall dataset that contains images from Kaggle and Google. On the class-specific metrics, the prediction accuracy varies, with the cashew class performing the best at 96.6% accuracy and pistachio performing the worst at 88.2% accuracy.

Key Metrics

Metric	Value
Box(P) (Precision)	0.888 (88.8%)
R (Recall)	0.828 (82.8%)
mAP50	0.921 (92.1%)
mAP50-95	0.704 (70.4%)

Fig. 34. General metrics for the overall test dataset, including Kaggle's and Google's

Class-Specific Metrics

Class	Images	Instances	Box(P) (Precision)	R (Recall)	mAP50	mAP50-95
Almond	50	197	0.92 (92.0%)	0.759 (75.9%)	0.916 (91.6%)	0.733 (73.3%)
Cashew	50	205	0.924 (92.4%)	0.896 (89.6%)	0.966 (96.6%)	0.789 (78.9%)
Peanut	50	668	0.896 (89.6%)	0.906 (90.6%)	0.954 (95.4%)	0.681 (68.1%)
Pistachio	50	315	0.809 (80.9%)	0.818 (81.8%)	0.882 (88.2%)	0.639 (63.9%)
Walnut	50	149	0.892 (89.2%)	0.758 (75.8%)	0.885 (88.5%)	0.677 (67.7%)

Fig. 34. Class-specific metrics for the overall test dataset, including Kaggle's and Google's

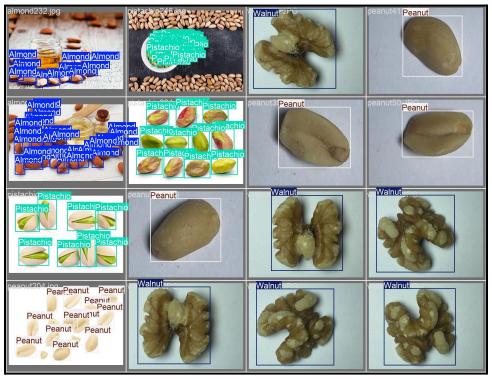

Fig. 35. Labels of the second batch in overall test dataset

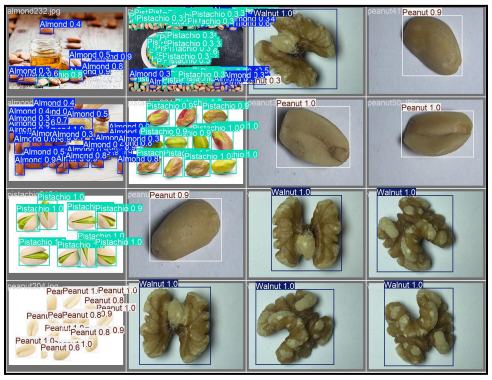

Fig. 36. Predictions of the second batch in overall test dataset

While these results look promising and accurate, the model **contains limitations** such as sometimes struggling to

disregard objects that are not nuts. If the model is given an image without nuts, it sometimes boundboxes random objects as nuts.

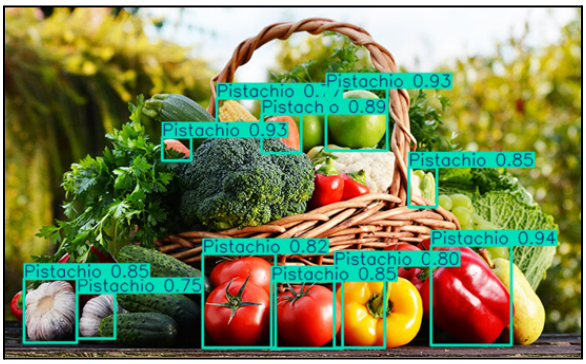

Fig. 37. Predictions of the model given an image without unshelled nuts

VI. CONCLUSION

This nut classification project not only addressed the need for efficient and accurate nut classification in the food industry, but also the potential of deep learning, CNN, and YOLOv8 in changing classification processes. By using the Convolutional Neural Network and the YOLOv8 object detection model, NutSure has shown its potential to change the quality control and processing in the food industry.

The NutSure system achieved a validation accuracy of 94% and a testing accuracy of 92.1%, showing effectiveness in distinguishing between the world's most popular peanuts; almond, cashew, peanut, pistachio, and walnut. This was able to happen because of the thorough collection of data, annotation, and iterated training of the model until it gave a pleasing result.

And as the demand for automation in the food industry grows, the NutSure system not only shows the high accuracy and reliability for classifying unshelled nuts but also paves the way for future advancements and enhancements in quality control and processing.

VII. RECOMMENDATION

For future projects similar to the nut classification system, the researchers recommend incorporating more clustered shelled or unshelled nut images into the dataset, helping the model to have more accurate predictions and little to no false positives. This will make the system acceptable and usable for real-world scenarios where processing nuts are often grouped together. In addition, future projects should introduce background or negative classes to help the model distinguish nuts from irrelevant objects in images or real-time captures. Lastly, to be more valuable in the food industry, future researchers can add more features in the existing system. They can add a feature where the system can detect if a nut is fresh or damaged to contribute to the quality control sector of the nut industry.

VIII. REFERENCES

[1] K. Corp and M. Solutions, “Screening Nuts for Quality and Classification”, [Online]. Available: https://blog.kason.com/blog/making-the-grade-sorting-nuts-for-quality-and-classification

[2] V. Kiprop, “The most popular nuts in the world,” WorldAtlas, Dec. 13, 2018. https://www.worldatlas.com/articles/the-most-popular-nuts-in-the-world.html

[3] Grocery Manufacturers Association et al., Industry Handbook for Safe Processing of nuts. 2016. [Online]. Available: https://ucfoodsafety.ucdavis.edu/sites/g/files/dgvnsk7366/files/inline-files/261404.pdf

[4] N. Munguia, “What is food fraud?,” FSNS, Mar. 21, 2024. https://fsns.com/what-is-food-fraud/

[5] M. Soori, B. Arezoo, and R. Dastres, “Artificial intelligence, machine learning and deep learning in advanced robotics, a review,” Cognitive Robotics, vol. 3, pp. 54–70, 2023, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667241323000113

[6] Keylabs, “Under the hood: YOLOV8 architecture explained,” *Keylabs: Latest News and Updates*, Mar. 19, 2024. https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/#:~:text=YOLOv8%20is%20a%20state%2Dof,objects%20in%20real%2Dtime%20scenarios.

[7] “NuTs Market Booming regional demand and supply analysis,” *Journal*. https://vocal.media/journal/nuts-market-booming-regional-demand-and-supply-analysis

[8] “An Introduction to Convolutional Neural Networks (CNNs),” *DataCamp*, Nov. 14, 2023. https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[9] R. Yamashita, M. Nishio, R. K. Gian DO, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” Insights Into Imaging, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

[10] K. Kalra, “Convolutional neural networks for image classification,” *Medium*, Jul. 14, 2023. [Online]. Available: https://medium.com/@khwabkalra1/convolutional-neural-networks-for-image-classification-f0754f7b94aa

[11] “What is kaggle and what is it used for?,” *Coursera*, Sep. 09, 2024. https://www.coursera.org/articles/kaggle.

[12] B. K, “Object detection Algorithms and libraries,” *neptune.ai*, Apr. 15, 2024. https://neptune.ai/blog/object-detection-algorithms-and-libraries.

[13] “Under the hood: YOLOV8 architecture explained,” *Keylabs: Latest News and Updates*, Mar. 19, 2024. https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/#:~:text=Key%20Takeaways-,YOLOv8%20is%20a%20state%2Dof%2Dthe%2Dart%20deep%20learning,and%20applications%20in%20computer%20vision.

[14] G. Boesch, “YOLOV8: A Complete Guide [2025 Update],” *viso.ai*, Dec. 06, 2024. https://viso.ai/deep-learning/yolov8-guide/#:~:text=YOLOv8%20is%20the%20newest%20model,easy%2Dto%2Dimplement%20framework.

[15] Ultralytics. blog.roboflow.com. (n.d.). GitHub -ultralytics/ultralytics at GitHub. https://github.com/ultralytics/ultralytics?ref=blog.roboflow.com

[16] “Top 6 Advantages of Python over Other Programming Languages.” https://www.invensis.net/blog/benefits-of-python-over-other-programming-languages

[17] “What is Python? Executive Summary,” *Python.org*. https://www.python.org/doc/essays/blurb/

[18] “The Python Standard Library,” *Python Documentation*. https://docs.python.org/3/library/index.html

[19] J. Solawetz, “What is YOLOv8? A Complete Guide.,” *Roboflow Blog*, Oct. 30, 2024. https://blog.roboflow.com/what-is-yolov8/#:~:text=As%20opposed%20to%20other%20models,coding%20experience%20than%20prior%20models.

[20] K. Kamr, "Fruits and Vegetables Image Classification," GitHub, 2024. [Online]. Available: https://github.com/khaledkamr/image-classification. [Accessed: 10-Jan-2025].

[21] G. Redmon and A. Farhadi, "YOLOv8: Real-Time Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024.

[22] "Mastering Fine-Grained Image Recognition: Techniques and Challenges," Toolify AI, 2024. [Online]. Available: Toolify AI.

[23] X. Chen et al., "Fine-Grained Object Detection Using YOLOv8," *International Journal of Computer Vision and Image Processing*, vol. 45, no. 7, pp. 88–101, 2023.

[24] "A Review of YOLOv8 Applications in Smart Agriculture Using Multimodal Data," Preprints, 2024. [Online]. Available: Preprints.