Amazon Warehouse Bin Size Verification Using Image Processing And Machine Learning

Priya Govindarajulu

Master of Science in Data Analytics, San Jose State University

Data 270: Data Analyst Process

Dr. Eduardo Chan

May 17, 2024

Modeling

Amazon is known for its automation systems, which use robots to handle tasks such as counting items in the bins, moving pods around the warehouse, and packing the items by classifying them. This "Amazon Warehouse Bin Size Verification using Image Processing and Machine Learning" project uses machine learning models to count the items in the bins using the robot's images. The object detection and counting of the objects in the images will help robots understand that they carry the correct bin, which helps to improve accuracy and reduce the errors of handling wrong bins. The data is collected using the robot's camera in the Amazon warehouse and gathered from the Amazon Bin Image Dataset (Amazon,2024). This data registry has 536,435 images and their respective metadata in JSON format. The 10000 images are sampled from the dataset using stratified sampling, and it has images with quantity of bin items such as 385 images from quantity zero, 1615 images from quantity one, and 2000 images from quantities two to five. The CSV file is created from a JSON file to include all the details related to items in the bins, such as length, name, height, width, and target variable quantity, with an image file name for connecting to the image dataset. The data preprocessing steps such as improving image contrast by Contrast Limited Adaptive Histogram Equalization, Laplacian of Gaussian used for image smoothing to reduce noise, edge detection, and bilateral filtering to reduce the noise created by the tape, which helps to prevent the item's from falling while moved by robots. The four machine and deep learning models You Only Look Once, Convolutional Neural Network, Faster Region-based Convolutional Neural Network, Contrastive Language–Image Pretraining developed in Google Cloud Vertex AI and stored in Google Cloud Storage for this project and this report mainly focuses on the deep learning You Only Look Once(YOLO) model. High computational resources are used to develop this model using Python frameworks such as TensorFlow and PyTorch. This project leverages deep learning techniques for object detection to identify and count bin items in Amazon warehouse environments.

Model Proposals

The You Only Look Once(YOLO) model has been chosen for its capability to detect objects in a real-time environment. This deep learning model is known for its rapid speed since it looks at the image only once. YOLO is very efficient in handling real-time streaming data and provides object counts quickly with higher accuracy. Research by Mamdouh et al. (2021) used the YOLO model for olive fruit fly detection and counting. This research aims to enhance olive crop production by reducing the damage caused by flies. The deep learning YOLO framework is used to detect and count the flies. The labeling for this model has been done manually using the LabelImg tool, and they have identified both male and female flies as one class. To increase the dataset, they performed data augmentation techniques like flipping and rotating images, and the authors introduced negative samples of butterfly images, which helped to reduce the false rate. This framework achieved a high mean Average Precision(mAP) of 96.68%, outperforming all existing pest detection systems. This model's precision is 0.84, with a high recall of 0.97. This framework's requirements for computational resources like GPU are very high for processing, which is not feasible for real-world implementation. This paper's goal aligns well with this project, where the bin items must be detected and counted to identify the quantity in bin images.

Dirir et al. (2021) researched counting objects in smart cities to manage transportation systems. Deep learning frameworks like YOLO and correlation filters like Channel and Spatial Reliability trackers(CSRT) are used to detect and count objects. This paper aims to improve the urban transportation system by tracking real-time data. The videos are used to train the models, which feature varying conditions such as quality of the images, angle view, speed of moving vehicles in motion, and testing object density impacts. The video stream input is segmented into frames and sent to the YOLO model to detect cars in the frames. After detecting the object, the object counting algorithm counts the objects that move and cross the line drawn in the frame. Due to the integration of YOLO with CSRT, the authors used accuracy performance metrics to evaluate this paper, and the results showed a 96%

average accuracy in counting objects. This paper has limitations for high-density objects, and poor image quality under different weather conditions might decrease the accuracy. This paper uses the YOLO model to detect the objects for counting and uses the same class of cars for all the vehicles.

The study conducted by Song et al. (2021) addresses the challenges in robot object detection systems by providing an improved YOLOv5 object detection method. This study was motivated by increased demand for robots' accuracy in detecting objects in current industry settings. The robotic arm with the camera is set up for training and picking up objects by detecting them. The researchers focused on modifying the existing YOLOv5 framework by adjusting network layers and changing the training procedure of the model, which might increase the accuracy and reduce the load on computational power. The improved YOLOv5 model results showed exceptional performance with a mean Average Precision of 99.43% with precision and recall rates of 99.38%. The limitations include specialized hardware setup and advanced calibration methods to train the robots. The test data is not diverse, with enough objects to represent the real-world environment settings.

Inbar et al. (2023) proposed wastewater treatment analysis using deep learning models like Faster R CNN and YOLOv5 to monitor sludge in wastewater using object detection techniques. This paper aims to improve the treated water's quality by analyzing microscopic images of the sludge samples. The sludge samples are collected from wastewater treatment plants, and microscopes are used to capture the sludge visuals. The images are annotated using bounding boxes to show the biological components in the sludge, and the Roboflow framework is used for annotation. The histogram equalization is applied in the data preprocessing, and both YOLOv5 and Faster R CNN models are trained using these preprocessed and annotated images. Without the data preprocessing step, YOLOv5 achieved a mAP of 0.67, which is 15% better than the Faster R-CNN model, and with histogram equalization preprocessing, the mAP of the YOLOv5 model increased to 0.77. The quality of the input should be better for these models to perform better, but in real-time, the input image quality

significantly varies, which might reduce the model's performance. Integrating these deep learning models in existing wastewater treatment plants is complex.

The research by Anish et al. (2023) addresses the challenges of using object detection systems to automate assistance for visually impaired individuals. The goal is to improve the quality of life for these persons using this advanced object detection technique, the YOLO algorithm. This study used the YOLO model for object detection and bounding box prediction with class probability in a single evaluation. The system used images with objects available in the room for training and achieved an overall accuracy of 94% with high-level object detection. The need for high-quality image data and computational resources might limit its real-time implementation where only low resources are available.

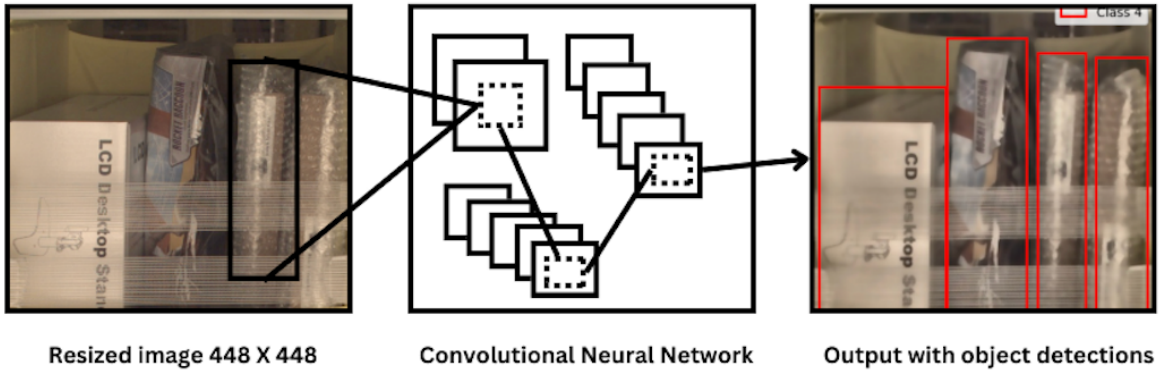
All the above research suggests that the YOLO model is suitable for object detection and counting tasks. It also works quickly and is ideal for real-time environments like Amazon warehouse settings. The performance metrics show that this model requires a high-quality image for better performance and high computational resources to maintain its performance.

You Only Look Once(YOLO)

The You Only Look Once algorithm is used to detect objects. As the name suggests, this YOLO algorithm processes the image once. This algorithm is known for its speed, especially when handling real-time data. The YOLO algorithm uses the single convolution network layer to perform bounding box prediction, identifying class probability within the predictions. This algorithm is proposed by Redmon et al. (2016), and Figure 1 shows the simple representation of the YOLO method for detecting objects. The bin images are resized to 416 X 416 or any suitable resolution based on computational resources. Then, a convolutional network runs on the image to predict the bounding box with object classification, and the predictions on the image are given as the output.

Figure 1

YOLO algorithm method for detecting objects.



Note. YOLO object detection system illustration, modified to demonstrate application using the Amazon Bin Image Dataset(Amazon,2024). Initially from "*You Only Look Once: Unified, Real-Time Object Detection*," by Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., 2016, retrieved from arXiv:1506.02640v5 [cs.CV], p. 1. DOI: https://doi.org/10.48550/arXiv.1506.02640.

YOLO Object Counting Algorithm

Dirir et al. (2021) proposed the Pseudocode of the proposed object counting algorithm by the YOLO model, shown in Figure 2. This pseudocode is an example of how to apply YOLO for object detection and counting tasks. The input is the frame from a video or static image; the output is the count of objects detected in the frame. The preprocessed images are sent through the YOLO model for object detection by using bounding boxes and classification of objects, which is the single class for this algorithm. The Channel and Spatial Reliability trackers technique is used along with the YOLO model to track the objects that cross the threshold line to ensure the same object is not counted twice. For each detected object by the YOLO model, the distance between the crossing line and the object is calculated, and if the value is less than the threshold, the object has not crossed the line. The Intersection over

Union(IoU) calculation checks if the object overlaps with tracked objects. The process loops through all the detected objects by the YOLO model and updates the counts.

Figure 2

YOLO Pseudocode for object counting algorithm.

Algorithm 1: Object Counting Algorithm V2

```
Input: frame, crossingLine
Output: counts
trackedObjects = []
counts = 0, threshold = 20, IoUMin = 0.2
detectedObjectes = YOLO(frame)
trackedObjects = CSRT.updateStatus(trackedObjects)
for object in trackedObjects do
    displacement = findDistance(object.position, crossingLine)
    if absolute(displacement) ≥ (threshold) then
        trackedObjects.remove(object)
    else
for object in detectedObjectes do
    displacement = findDistance(object.position, crossingLine)
    if absolute(displacement) < threshold then
        isNewObject = perfromIOU(object, trackedObjects, IoUMin)
        if isNewObject then
            trackedObjects.add(object)
            counts = counts + 1
        else
    else
return counts
```

Note. YOLO Algorithm for object counting, reprinted from "*An Advanced Deep Learning Approach for Multi-Object Counting in Urban Vehicular Environments*," by Dirir A, Ignatious H, Elsayed H, Khan M, Adib M, Mahmoud A, Al-Gunaid M, *Future Internet*. 2021, p. 10. ; 13(12):306. https://doi.org/10.3390/fi13120306.

YOLO Equations

The images are annotated for the bounding boxes, and annotation can be done manually or automated using software. If manual annotation is done on the images, it is called ground truth. The bounding box predicted and the ground truth difference can help us evaluate the Intersection over Union(IoU) localization task. Lakshmanan et al. (2021) stated the YOLO model's IoU and loss function equations. The IoU can be calculated by the area of predicted area to the ground truth overlap to the total area as given in Equation 1.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \tag{1}$$

The input images are converted into grids. The ground truth is assigned in these grid cells using the IoU, and predicted boxes will be within the same grid cells. Now that the pairings of ground truth and predicted boxes are in place, the loss can be calculated for the bounding box using Equation 2.

$$L_{\text{box}} = (x - \hat{x})^2 + (y - \hat{y})^2 + \left(\sqrt{w} - \sqrt{\hat{w}}\right)^2 + \left(\sqrt{h} - \sqrt{\hat{h}}\right)^2 \tag{2}$$

In Equation 2, the hat coordinates are for predicted boxes, and the ground truth coordinates are without the hat. The x and y are the bounding box's center coordinates, and the bounding boxes' width and height are represented using w and h. The final loss is calculated by finding and adding the bounding box loss from all the grids with the model's weighting factor like λ(box) =5, as mentioned by the author.

The confidence score is provided for each bounding box, indicating the object's confidence existing in that box. The research paper by Mamdouh et al. (2021) mentions Equation 3 to calculate the confidence score.

$$Confidence = P(object)\ \times IoU \tag{3}$$

Model Optimization. The IoU helps measure the overlap between the predicted and ground truth bounding boxes, and for this project, it is set at 0.50. The predicted bounding box overlaps the ground truth by at least 50%. In a warehouse environment like Amazon, where most items are obscured by boxes or noise like tapes, lower IoU helps to prevent missing the detections of the objects. This IoU

0.5 is slightly higher than the IoU 0.2 defined in Figure 3 YOLO algorithm. The Confidence score threshold is set at 0.2 to check whether predicted objects exist in the bounding box for this project.

Model Supports

Environment, Platform, and Tools

Proper hardware and software selection is required to create a deep learning model like YOLO, train it, deploy it, and prepare it for the test phase. The hardware setup for this model used Google Cloud Platform for computational engine and storage. The computational engine configuration is n1 standard 8 with 8v CPUs with 32 GB RAM, and additionally, NVIDIA L4 g2 -standard -8 is used with 1 GPU of 24GB GDDR6 GPU memory. The CPU is used for low computational tasks like model and test validation, and the GPU is used for high computational tasks like training the deep learning model. The Google Cloud storage stores collected data of size 30 GB with images and metadata. The total storage of 120GB is allocated for standard storage, where all the preprocessing, transformation, and model training data is stored. The archival storage of 120GB is allocated to store the data after the project completion and to maintain the backup.

The software requirements for this model development are Vertex AI in the Google Cloud, which is used for model creation, deployment, and fine-tuning. The Google Collab is used to write Python programs for data preprocessing, connect to train the models, and test the models for evaluation. Also, the Google Cloud SDK software is used to manage the resources in the cloud from the local system. GitHub is used for version control, and Google Docs is used for creating reports. Microsoft Office is used to create presentations and edit reports.

Table 1 describes the tools and Python libraries used for this project. The Scikit Learn library splits the data into train, test, and validation. The standard scaler normalizes the data, and metrics such as F1 score, precision, and recall are calculated using the Scikit-Learn metrics module. NumPy handles the mathematical calculation and assigns the random seed, and Pandas loads the CSV file data into the

data frame to prepare the data for the model. The OS library handles the path creation for data split, and the Shutil library handles file copy. The matplotlib creates metrics visualizations such as F1 graph and confusion matrix. The pillow library is used to open the images for processing, and the YOLOv5 model is used to detect objects in the images. The Pytorch library is used to load pre-train models and save the models, as well as image transformation like resizing and providing GPU support. The LabelImg tool annotates the images with ground truth bounding boxes.

Table 1

Tools and Libraries

Library	**Module**	**Method**	**Usage**
Pandas	Dataframe	read_csv	This method reads csv file into data frame for analyzing and manipulating the data.
NumPy	Random	Seed, array, float64	Provides support for multidimensional arrays, matrices and controlling the random number generation
Scikit-Learn	model_selection	train_test_split	To split the dataset into train, validation, and test.
	Preprocessing	StandardScaler	Features are standardized by making mean zero and standard deviation one.
	Metrics	Precision_score, recall_score, f1_score	To calculate performance metrics for the model.
OS	-	makedirs, path.join	Used for file system management and path manipulation to create folders.

Shutil	-	Copy	High level file operations such as copying files.
Matplotlib	Pyplot	show, subplots	For creating visualizations and displaying figures graphically.
Pillow	Image	Open	Helps to open image files that is manipulated by python.
YOLOv5	-	Train, detect	Training and detecting objects using YOLOv5 model.
Pytorch	Torch	Load, save, no_grad, cuda	Provides GPU support, loading and saving models and handling gradients.
	Torchvision	Models, transforms	To load pretrained models and for image transformation.
LabelImg	-	-	Image annotation tool used to add ground truth bounding boxes in the images.

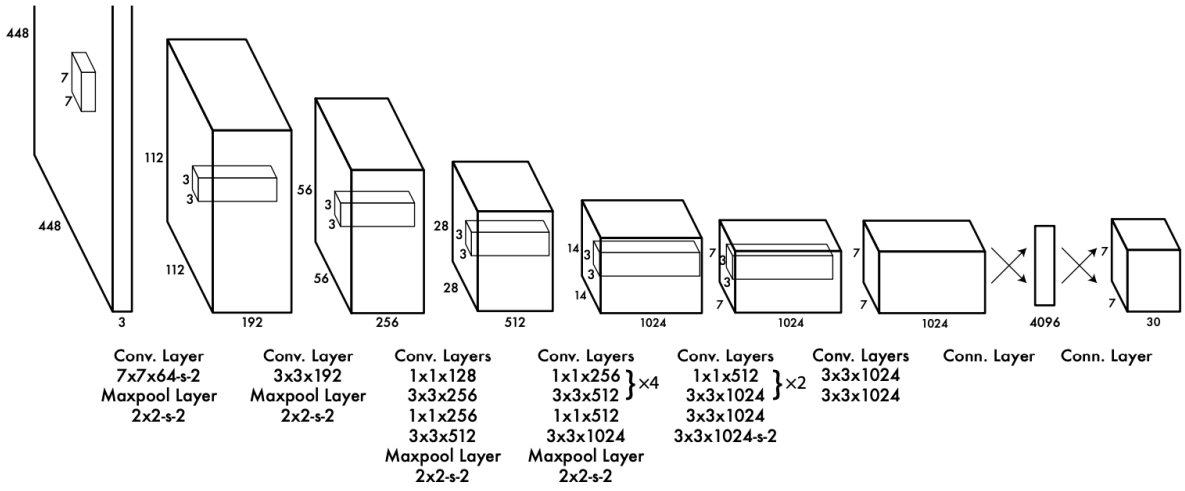
Data Flow and Model Architecture

The YOLO architecture, initially proposed by Redmon et al.(2016), has a single neural network layer(Figure 3). This neural network utilizes a series of convolutional layers and fully connected layers. The convolutional layers are the building blocks of the neural network, which process input images to create feature maps such as edges and texture. This architecture uses 24 convolutional layers, a deep-layered network with two fully connected layers, as shown in Figure 3. The authors have trained the model on low-resolution images and used high-resolution images for detection. The convolutional layers are altered to 1X1, which reduces the depth of the feature maps, compresses the model, and reduces

the computational cost without losing feature information. The fully connected layers convert the learned feature representation into final output predictions.

Figure 3

YOLO Architecture.



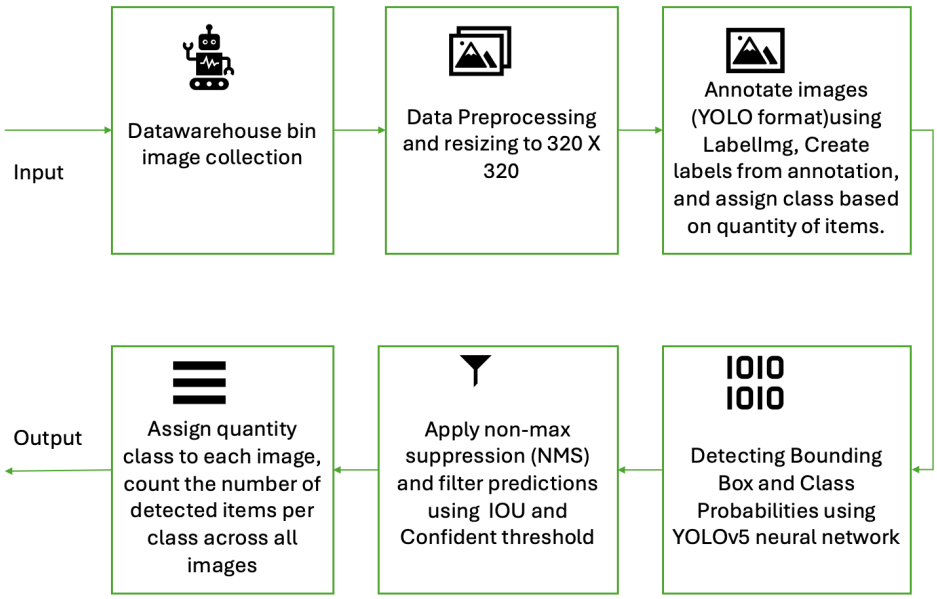
Note. YOLO architecture, reprinted from "*You Only Look Once: Unified, Real-Time Object Detection*," by Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., 2016, p. 3. , retrieved from arXiv:1506.02640v5 [cs.CV], DOI: https://doi.org/10.48550/arXiv.1506.02640.

With so many versions evolved in YOLO by modifying the network layer, YOLOv5 is one of the fastest models. It is known for its enhanced performance and efficiency in detecting objects. Pham et al.(2023) mentioned that this YOLOv5 model is implemented in the PyTorch platform, which makes it easy to use and deploy. This version replaced the focus layer with a convolutional layer that aims to improve the exportability of the model. This model uses a Path Aggregation Network, which improves low-level feature propagation and helps to capture the fine-grained details. These changes in the network layer with PyTorch Platform make it more suitable for object detection and counting. This project uses YOLOv5 with pre-trained weights to train the model for detecting and counting items in the bin images.

The YOLOv5 model flow diagram for this project is shown in Figure 4. The bin images taken by robots are collected from the Amazon warehouse with their JSON metadata. The Data Preprocessing is completed, and the images are resized to 320 X 320, optimizing them for the YOLOv5 model. The LabelImg tool is used to annotate the images in the YOLO bounding box format, which includes a class of the item, x and y center coordinates of the bounding box, and its length and width from the center. Since this project goal is to count the objects, the objects in the images are annotated under single item class. Every item in the image is enclosed with a bounding box and assigned to the class based on the number of items in the images. If there are four items in the image, class four is assigned. These labeled images are used to train the YOLOv5 model, and the model counts the items in the images using the bounding box and classifies items based on the visible items. The Non-Max suppression method is used to refine and filter the predictions using the confidence score and Intersection over Union(IoU) threshold. The model counts the number of items in each image and provides the total quantity predicted. These predictions are then sent to output for further analysis and visualization.

Figure 4

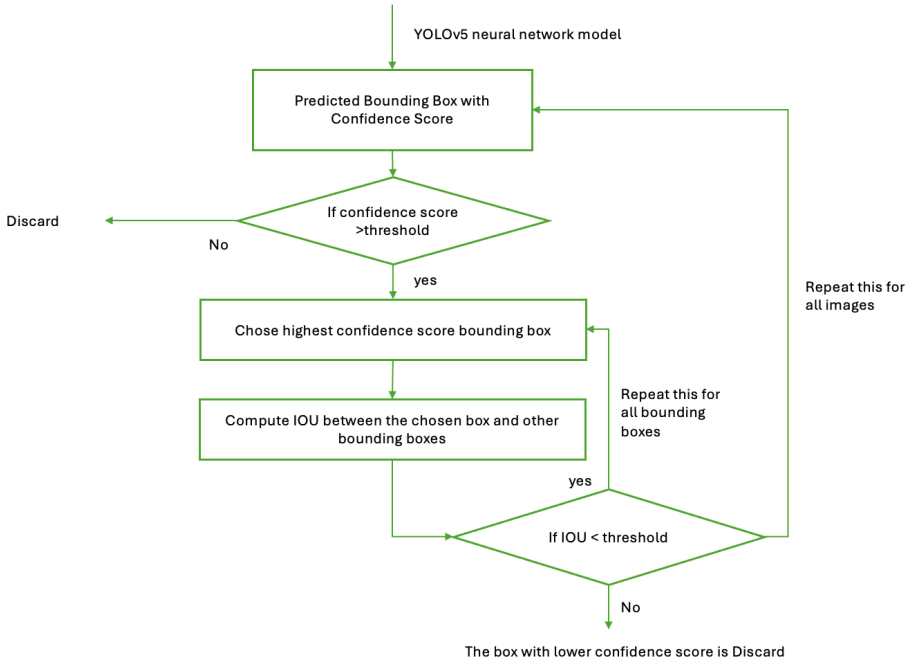
YOLOv5 flow diagram for counting bin items in Amazon warehouse bin images.



Based on the training, the YOLOv5 model predicts a bounding box for each item in the images with a confidence score in which the model correctly identified the object within a bounding box. Figure 5 shows the flowchart for the Non-Max Suppression Process after the YOLOv5 model predicts the bounding boxes for this project. This model's confidence score threshold is set at 0.2, and if the confidence score is less than the threshold, that bounding box will be discarded. If it exceeds the threshold, the image with all the chosen bounding boxes is moved for further processing. Among these boxes, the highest confidence score bounding box is chosen. IoU is computed between this chosen box and other predicted bounding boxes to find how accurate the bounding boxes are. If IoU is greater than the 0.5 threshold set, the box with a lower confidence score is discarded, and this process is repeated until all the predicted bounding boxes are checked and repeated for all images. This Non-Max Suppression process helps the model to improve object detection accuracy using the Confidence score and IoU threshold of the bounding boxes.

Figure 5

YOLOv5 flowchart for Object detection and counting using Non-Max Suppression Process.



The images collected from the warehouse settings are noisy and hard for the human eye to distinguish manually. The items need to be visible, and multiple items are packed in cardboard boxes or wrapped with covers. The tape at the front of the bin, which helps the items stay in place while robots move them, adds additional noise. Hence, data preprocessing has been done on these images to reduce the noise and improve clarity for better object detection by the models before resizing the images for the model's requirement. The data preprocessing steps such as improving image contrast by Contrast Limited Adaptive Histogram Equalization, Laplacian of Gaussian used to reduce noise, edge detection, and for image smoothing, and the noise created by tape is reduced by applying bilateral filtering, which smooths the noise at the front while retaining the items for detection at the back. Figure 6 shows the original image 00055.jpg collected from the warehouse, with images transformation after CLAHE, LOG, and bilateral filtering.

Figure 6

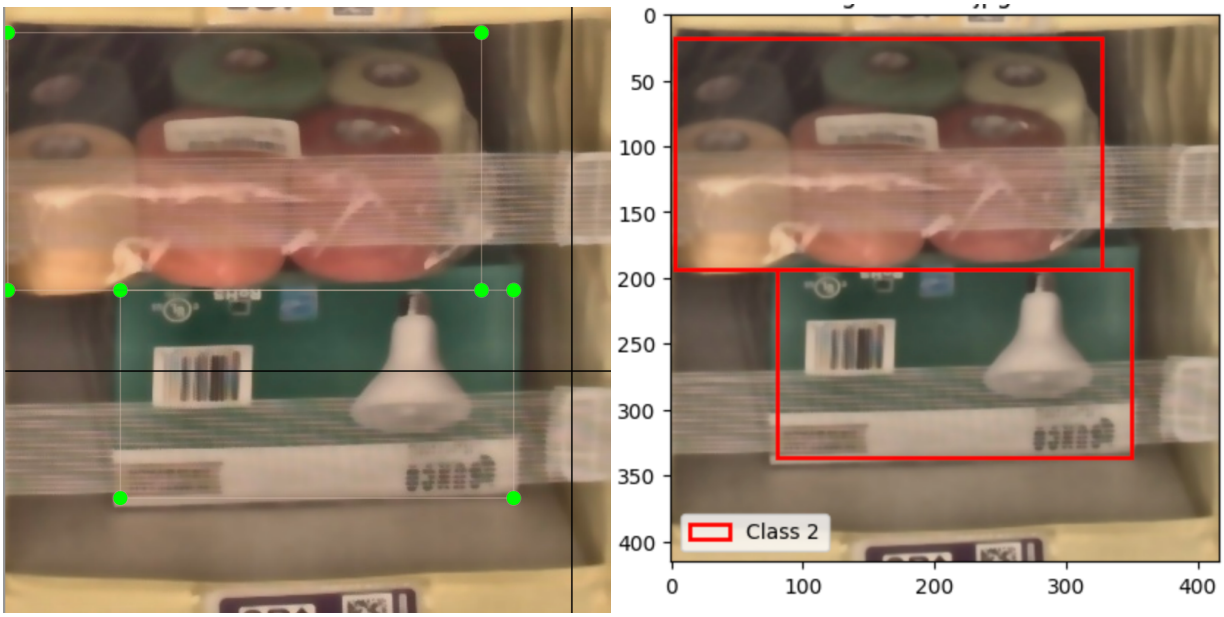
Data Preprocessing steps for bin images.

(a)Original image-00055.jpg (b) After CLAHE (c) After LOG and bilateral filtering

The LabelImg tool is used to annotate all the images with the YOLO model's bounding box format, and the green annotations are how the objects are labeled in the images, as shown in Figure 7. The red annotations are after predictions by the YOLO model, with respective counting done and classes assigned to identify the items in the bins.

Figure 7

Bounding box before and after for labeling the bin images.



(a)LabelImg Tool to annotate items- image 00025.jpg, (b) Images with Predicted Bounding box and counting class.

Model Comparison and Justification

Both Faster R CNN and YOLO models are developed for this project. Table 2 compares these models in terms of architecture, data types, computational needs, and other characteristics, with their strengths and limitations. The Faster R-CNN model's architecture has two phases: initial region proposal and convolutional processing using CNN. The YOLO model simplifies its architecture with a single pass of the images with integrated CNN. Both models handle image data for object detection and classification, and videos can also be used as data type for these models and images. YOLO performs better at various data sizes, but Faster R-CNN performs better for larger datasets. YOLO is less prone to overfitting compared to Faster R-CNN due to its model's simplicity, and YOLO trains faster than Faster R-CNN.

Both models require high computational resources, and GPUs are recommended for data training and testing. However, the YOLO model needs more GPU resources due to its real-time tracking

and single-process network layer compared to Faster R-CNN. Both models require preprocessing, such as image annotation, for object detection, but Faster R CNN needs this annotation to be done accurately for models to perform better. Suppose proper annotations are done for Faster R CNN. In that case, the results will be highly accurate and precise object detection, but training the model is slower due to its complexity. The YOLO model is rapid due to its model's single pass nature, which makes it suitable for real-time object detection, but it trades off accuracy for its speed.

Table 2

Comparison between Faster R-CNN and YOLO

Characteristic	Faster R-CNN	YOLO
Architecture	Two Stage Process(Region Proposal and CNN)	Single Shot detection(integrated CNN)
Data Type Handles	Images, Videos	Images, Videos
Data Scale Efficiency	Excels with Larger datasets	Performs well with both smaller to larger dataset
Risk of Overfitting	Highly possible due to model's complexity	Generally robust against overfitting
Preprocessing demands	Extensive need for detailed and accurate annotations.	Moderate and requires bounding boxes
Duration of Training	Longer due to detailed stages for analysis	Quicker due to its single pass architecture
Space requirements	High(to store features from multiple stages)	Low (compact model)
Computational needs	High (GPU recommended)	High(GPU recommended) and more than Faster R-CNN due to

		its real-time detection capabilities.
Main Strengths	Superior accuracy and precise object detection	Rapid processing suitable for real time environments
Key Limitations	Slower and complex model training	The finer details might be missed while doing the object localization.

Model Evaluation Methods

The model evaluation metrics such as Accuracy, Precision, Recall, F1 score, mean Average Precision, and confusion matrix are used to evaluate the Faster R-CNN and YOLO models. For models with object detection tasks, when a project involves more than one class, like quantity zero to five in this project, the accuracy can be calculated using mean Average Precision (Pham et al.,2023, p.15). As mentioned in the research paper by Pham et al.(2023), the metrics are described with a calculation formula.

Accuracy

Accuracy is the most straightforward metric used for the model evaluation. It measures the ratio of correctly predicted observations to the total observations in the dataset. High accuracy indicates that models can distinguish across all the samples. However, this is not a reliable indicator where class distribution is imbalanced, like different quantities of items in this project. The formula to calculate accuracy is given in equation 4, and the TN, TP, FP, and FN means True Negatives, True Positives, False Positives, and False Negatives of the predicted values, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Precision

Precision calculates the accuracy of positive predictions. It reflects the correct positive identification proportion. This is crucial in scenarios where the false positive error is more severe than the false negative error. The precision is calculated using equation 5, and the TP and FP are true positives and False Positives, respectively.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

Recall

Recall measures the model's ability to identify all positive cases correctly, and it is known for its sensitivity. A higher recall indicates that the model captures a large proportion of positive cases, helping to ensure that only a few positives are missed. The recall is calculated using equation 6 and TP, FN are true positives and false negatives, respectively.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

F1 Score

The F1 score considers both false negatives and false positives and helps to find the mean of precision and recall. This is a balanced metric, especially if we have an uneven class distribution, like the bin quantity class in this project. The F1 score is very high, and this indicates that the model performs well for false positives and false negatives. The F1 score calculation is given in Equation 7.

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

Mean Average Precision(mAP)

The mean average precision is the key performance indicator for object detection models, where multiple objects must be detected within the same image. The mAP provides a metric that averages precision across all classes and multiple IoU thresholds. It gives the model effectiveness across different class overlap levels and detection thresholds. The area under the recall curve is the average

precision, as shown in Equation 8. The mean Average Precision for six classes in this project can be calculated using the formula below, as shown in equation 9.

$$AP = \int_0^1 p(r)dr \tag{8}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \tag{9}$$

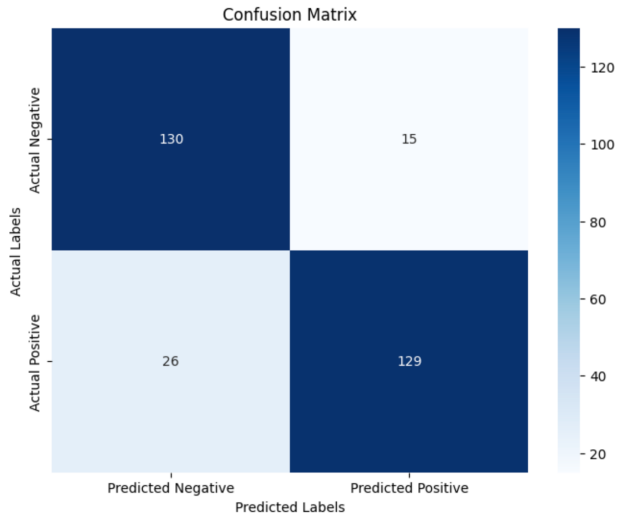
In equation 9, the number of classes is N, and AP is the average precision for class i.

Confusion Matrix

The confusion matrix specifies the model's performance, especially classification models. As shown in Figure 8, each row represents the actual class values, and each column represents the predicted class values by the object detection model. This matrix helps to assess the model's errors visually and shows the classes that predicted correctly, which can be used to refine the model's performance.

Figure 8

Structure of confusion matrix.



These performance metrics help validate and evaluate both models and help compare them to understand how they perform for this project.

Model Validation and Evaluation

You Only Look Once(YOLO)

Accuracy. The accuracy of the YOLO model is calculated by using Equation 4, and the result is shown in Figure 9. The model shows an approximate validation accuracy of 43% and a test accuracy of 41%. The validation suggests that almost half of the validation data is classified by the model, and test accuracy is slightly lower than the validation accuracy. The drop in accuracy indicates that the model is having difficulty detecting unseen data. This traditional accuracy is not a reliable indication of object detection models where multiple object detection in a single image is involved. Even though accuracy can be calculated for the YOLO model, the mean Average Precision is the proper metric for evaluating the model's performance, especially in this project where multiple classes of different item quantities are used.

Figure 9

Accuracy of YOLO model.

```
YOLO model validation accuracy is 0.43456
YOLO model test accuracy is 0.41436
```

Precision. As shown in Figure 10, the precision for validation data is 0.46, which indicates that the predicted labels across all classes were correct and showed moderate precision. The precision for quantity zero is very high, at 78%, suggesting excellent prediction when no items are present. For quantities two to five, the precision is lower than that of quantities zero, indicating a mix of correct and incorrect precisions. Figure 11 shows the precision for test data, which is 0.448; 44.8% of the predicted labels of all quantities were accurate for the unseen data. In the test data, the precision of quantity zero increased, indicating this model handles the bin with no items well with high accuracy among other item quantities. The precision of the remaining quantities in the test data is similar to the validation results.

Figure 10

Precision, Recall, and mAP for the YOLO model's validation data.

Class	Images	Instances	P	R	mAP50	
all	990	990	0.46	0.77	0.556	0.555
Quantity 0	990	33	0.781	0.97	0.965	0.965
Quantity 1	990	169	0.616	0.806	0.782	0.781
Quantity 2	990	185	0.365	0.692	0.439	0.439
Quantity 3	990	216	0.322	0.704	0.348	0.348
Quantity 4	990	182	0.277	0.742	0.316	0.315
Quantity 5	990	205	0.399	0.707	0.484	0.483

Figure 11

Precision, Recall, and mAP for the YOLO model's test data.

Class	Images	Instances	P	R	mAP50	
all	1000	1000	0.448	0.797	0.554	0.553
Quantity 0	1000	40	0.835	1	0.992	0.992
Quantity 1	1000	168	0.558	0.792	0.77	0.769
Quantity 2	1000	208	0.367	0.784	0.467	0.467
Quantity 3	1000	179	0.224	0.693	0.259	0.259
Quantity 4	1000	200	0.296	0.79	0.348	0.348
Quantity 5	1000	205	0.407	0.727	0.486	0.486

Recall. As shown in Figure 10, the overall recall for the validation data is 0.769, which suggests that the model correctly identified 76.9% of all actual instances, demonstrating good detection capabilities. Each quantity shows good recall greater than 70%, with quantity zero having the highest recall of 0.97, which shows effective detection of actual instances across all quantities. The overall recall for test data is 0.797, as shown in Figure 11, which is higher than the validation overall recall value and indicates a high ability to detect relevant classes. The quantity zero achieved a perfect recall score 1 in the test data, where all the actual classes are correctly identified. The quantity one has a perfect recall of 0.79, showing strong detection capability. The quantity two, three, four, and five have recall values of 0.78, 0.69,0.79 and 0,727, respectively.

Mean Average Precision(mAP). As shown in Figure 10 for the validation data, the mAP50 is 0.557, and the model's precision at a 50% IoU threshold is about 55.7%, which is reasonable for object detection tasks. Quantity zero has the highest mAP50 of 96.1%, indicating outstanding localization accuracy for this class. However, the mAP decreases when the quantity increases, with 0.35 mAP50 for

quantity three, which is very low and indicates notable challenges in accurate localization. The test data, as shown in Figure 11, shows that the model achieves a mean average precision of 55.4% at an IoU threshold of 50%, a moderate and overall balanced detection capability at this threshold level. Like validation data, the test data also showed a higher mAP50 for quantity zero, 0.992, and the lowest mAP50 for quantity three, 0.259.

F1 score. The overall F1 score of the test and validation remains consistent at 56%, which shows that the mode's performance is consistent across different datasets, as shown in Figures 12 and 13. For quantity zero, the F1 scores are high in both datasets, which indicates that no-item bins are more accessible to detect. There is stable performance for quantities one and five with moderate F1 scores—quantities two to four show lower F1 scores, especially in the test data. The difficulty in detecting these items might be because of overlapping or closely placed items and less image quality.

Figure 12

F1 score for Validation data.

```
F1 Scores for Validation Data:
Quantity 0: 0.86
Quantity 1: 0.69
Quantity 2: 0.48
Quantity 3: 0.44
Quantity 4: 0.40
Quantity 5: 0.50
Overall: 0.56
```

Figure 13

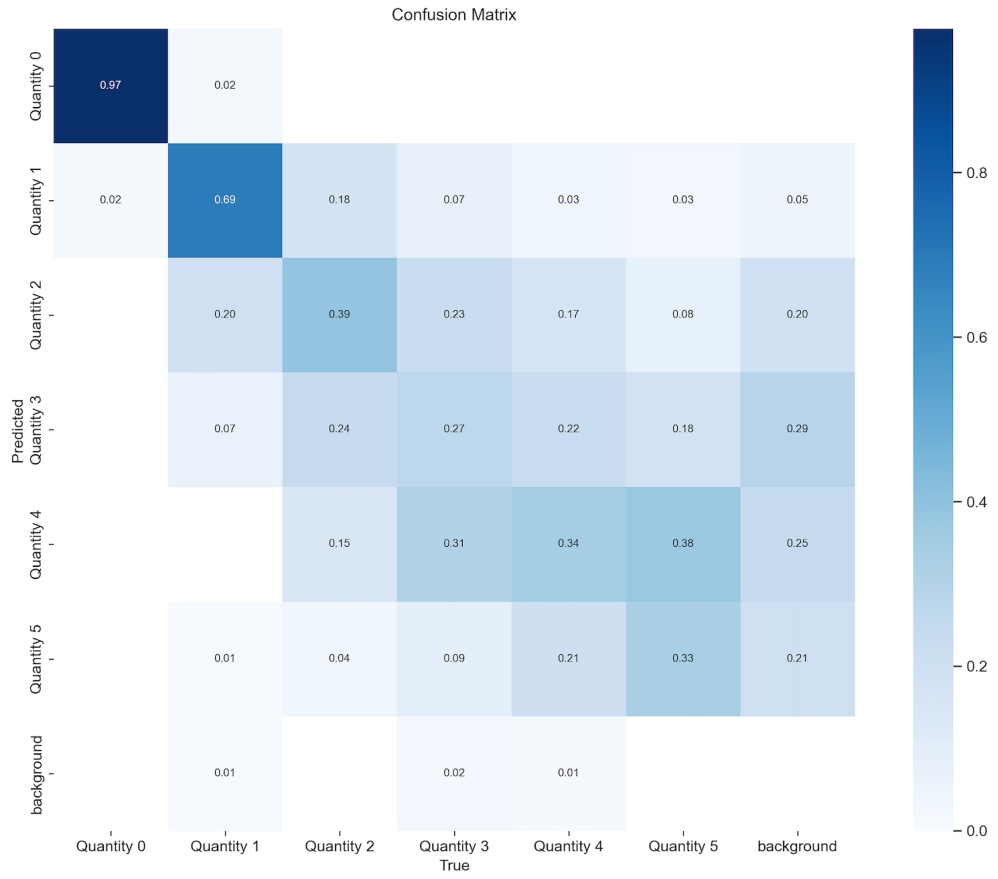
*F1 score for Validation data***.**

```
F1 Scores from the model for test data:
Quantity 0: 1.00
Quantity 1: 0.60
Quantity 2: 0.40
Quantity 3: 0.35
Quantity 4: 0.35
Quantity 5: 0.40
Overall: 0.56
```

Confusion Matrix. The confusion matrix, as shown in Figure 14, represents a classification model's performance in categorizing images by the number of items, ranging from zero to five. The model's accuracy in predicting items is shown in the diagonal values; for quantity zero, it is 0.97. For Quantity 1, only tiny fraction values are misclassified as quantity 1. Class quantity one is mostly correctly classified at 0.69, but there is notable confusion with quantities two and four. There is a pattern where higher quantities, such as from two to five, are often confused with the next lower or higher quantity. The organization of bin items in the warehouse makes precise counting challenging.

Figure 14

Confusion Matrix of Test data for YOLO model.



Conclusion. The YOLO model exhibits the most robust recall rate across all classes, indicating it effectively detects the items. However, precision is notably lower, significantly when quantity increases, suggesting that this model struggles with items in closer proximity or overlapping items. The mAP50 for unseen data is 55.4% at 50% IoU, indicating that this model performs moderately in detecting objects at this threshold level. The confusion matrix suggests that employing more robust training methods might help reduce the confusion, especially in higher quantities.

Results

For this project, the performance of Faster R-CNN and YOLOv5 is evaluated using the same test data with the same data size of 10% in 10000 images. Both models have never seen this test data before, and the metrics obtained for this test data are used to evaluate them. As shown in Table 3, the YOLOv5 model performs better in all metrics than Faster R-CNN. The YOLOv5 model shows a higher

overall accuracy of 41% than the Faster R-CNN model of 31.5%, which indicates that YOLOv5 correctly identifies and classifies objects and is better than Faster R CNN. The YOLOv5 has the highest precision of 44.8% compared to Faster R CNN, which is 34%. The Yolov5 also has the highest recall of 79.7%, but Faster R CNN has a 39.3% recall, which indicates YOLOv5 model is more reliable for detecting various quantities. The YOLOv5 has a mean Average Precision of 55.4% and leads over Faster R-CNN at 38%, demonstrating better average precision across all recall levels. The Faster R-CNN has an F1 score of 31.2%, and YOLOV5 has a score of 56%. These results showcase that YOLOv5 is a more robust choice for varied and dynamic object detection environments like training robots in warehouse environments.

Table 3

Comparison between Faster R-CNN and YOLO Results

Model	**Overall Accuracy**	**Precision**	**Recall**	**mAP**	**F1 score**
Faster R-CNN	31.5%	34%	39.3%	38%	31.2%
YOLOv5	41%	44.8%	79.7%	55.4%	56%

Conclusion

This project uses robust deep-learning techniques, such as the YOLO model, to demonstrate object detection and counting in warehouse environments where accuracy and efficiency are essential. Improving the accuracy of detecting items in the bins and counting for the robots to implement automation tasks in the warehouse environment is crucial. The YOLOv5 model has proven to be superior in handling various automation tasks, particularly detecting different quantities of the bins with a Mean Average Precision of 55.4% at the 50% IoU threshold, which is 24% better than Faster R-CNN. The YOLO models show a high recall of 79.7% and effectively detect the objects in the images. The YOLOv5 is the most balanced model, maintaining the same performance across all classes. Overall, the performance of

YOLOv5 for this project is moderate, which is common while handling multiple objects in the same images, and additional enhancements like hyperparameter finetuning and increasing dataset size for training might increase the performance for the YOLOv5 model.

Limitations and Future Scope

The limitations of this project are noticeable: When the quantity increases, the precision decreases, and the confusion matrix indicates there is confusion in detecting the correct quantity when numbers are high. This is mainly because items are arranged and packed in bins. Additional techniques like fine-tuning hyperparameters such as epoch for more training on the images and new storage techniques to store items in a bins for easier detection can be implemented to improve the current model's performance.

The future scope is to increase the data size for model training and improve image clarity by implementing proper Datawarehouse bin organization techniques. The YOLO model will be combined with other deep learning models like CLIP to create better object detection systems. They are implementing this YOLO model in warehouse environments to compare the real-time performance and increase data augmentation techniques to improve the model’s performance.

References

Amazon. (2024). Amazon Bin Image Dataset. AWS open data registry. Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States (CC BY-NC-SA 3.0 US). Retrieved February 2024, from https://registry.opendata.aws/amazon-bin-imagery.

Anish, A., Sharan, R., Hema M, A., and Archana, T., *Enhancing Surveillance Systems with YOLO Algorithm for Real-Time Object Detection and Tracking*, 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2023, pp. 1254-1257, Doi: 10.1109/ICACRS58579.2023.10404710, IEEE full paper access: https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/8274769.

Dirir A, Ignatious H, Elsayed H, Khan M, Adib M, Mahmoud A, Al-Gunaid M. *An Advanced Deep Learning Approach for Multi-Object Counting in Urban Vehicular Environments. Future Internet*. 2021; 13(12):306. https://doi.org/10.3390/fi13120306.

Inbar, O., Shahar, M., Gidron, J., Cohen, I., Menashe, O., & Avisar, D. (2023). *Analyzing the secondary wastewater-treatment process using Faster R-CNN and YOLOv5 object detection algorithms*. Journal of Cleaner Production. https://doi.org/10.1016/j.jclepro.2023.137913.

Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical Machine Learning for Computer Vision*. O'Reilly Media, Inc.

Mamdouh, N., and Khattab, A., *YOLO-Based Deep Learning Framework for Olive Fruit Fly Detection and Counting,* 2021 IEEE Access, vol. 9, pp. 84252-84262, Doi: 10.1109/ACCESS.2021.3088075 , IEEE full paper access: https://ieeexplore.ieee.org/document/9450822.

Pham, T. N., Nguyen, V. H., & Huh, J. H.*Integration of improved YOLOv5 for face mask detector And auto-labeling to generate dataset for fighting against COVID-19*. 2023, Journal of Supercomputing*, 79*, 8966–8992. https://doi.org/10.1007/s11227-022-04979-2.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, *You only look once: Unified, real-time object detection,* 2016,arXiv:1506.02640v5 [cs.CV*],* Doi*:* https://doi.org/10.48550/arXiv.1506.02640 .

Song, Q., Li, S., Bai, Q., Yang, J., Zhang, X., Li, Z., & Duan, Z. (2021). *Object detection method for grasping robot based on improved YOLOv5*. Micromachines, 12(11), 1273. https://doi.org/10.3390/mi12111273.