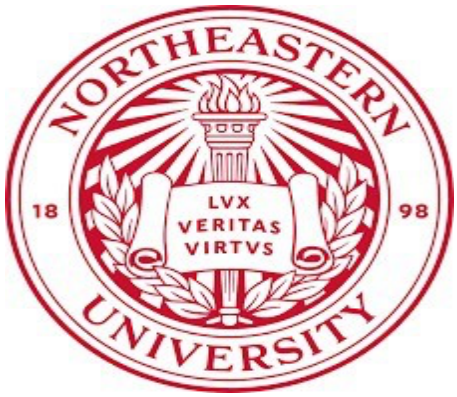


Discriminative Project Milestone #2

Group 5

Aditya Kumar

Meghana Rao

Simran Abhay Sinha

Sushmitha Sudharshan

kumar.aditya1@northeastern.edu
rao.meg@northeastern.edu
sinha.sim@northeastern.edu
sudharshan.s@northeastern.edu

Signature of student 1: Aditya Kumar (25%)

Signature of student 2: Meghana Rao (25%)

Signature of student 3: Simran Abhay Sinha (25%)

Signature of student 4: Sushmitha Sudharshan (25%)

Submission Date: 02/08/2026

ABSTRACT

This milestone focuses on building and evaluating a custom object detection system using the YOLOv8 deep learning framework. The main objective is to train a model that can detect multiple objects within a single image and correctly return both the object IDs and their spatial locations using bounding boxes. Instead of relying on a pre-existing multi-object dataset, a custom dataset was generated by combining randomly selected single-object images into composite images containing multiple objects. This approach allows precise control over object placement and ensures accurate bounding box annotations for training and evaluation.

The dataset was structured following the YOLOv8 format, with separate training and validation splits and a configuration file defining class names and dataset paths. The dataset contains multiple object classes represented using object ID labels. Before training, the dataset structure and image counts were verified programmatically to ensure correctness and compatibility with the YOLOv8 pipeline.

Several YOLOv8 model variants were trained, including Nano, Small, and Medium versions, to study the trade-offs between model size, accuracy, and computational cost. All models were trained using the same dataset and evaluation setup to ensure a fair comparison. In addition to the baseline models, an optimized version of the YOLOv8 Nano model was trained using enhanced data augmentation techniques and an adjusted optimizer configuration to improve performance without increasing model size.

Model performance was evaluated using standard object detection metrics, including mAP@0.5, mAP@0.5:0.95, precision, and recall. Results show clear differences in performance across model sizes, with larger models achieving higher detection accuracy. Based on validation results, YOLOv8 Small was selected as the best-performing model, providing strong accuracy while maintaining moderate computational requirements.

The selected model was tested on validation images and randomly selected samples. Inference results show that the model correctly detects multiple objects per image, assigns the correct object IDs, and produces accurate bounding box locations with high confidence scores. These results demonstrate that the trained model meets the milestone requirements for multi-object detection using a custom-trained YOLOv8 network.

1. Introduction

Object detection is a core task in computer vision where a model identifies what objects are present in an image and where they are located. In this project, the goal is to train a deep learning model that can detect multiple objects in a single image and return both object IDs and bounding box locations.

Instead of using a ready-made multi-object dataset, single-object images were combined to form new images containing multiple objects. This setup allows direct control over object placement, overlap, and labeling, while keeping the dataset aligned with the detection task.

The YOLOv8 architecture was used for this milestone due to its unified detection pipeline and real-time inference capability. The focus of this milestone is on custom training, correct annotation handling, and validating that the trained model can correctly identify and localize multiple objects in unseen images.

2. Dataset Preparation

The dataset was created as a grid-based multi-object detection dataset. Each image contains multiple objects formed by combining randomly selected single-object images into one composite image. Object placement was controlled so bounding box locations could be calculated during image generation.

The dataset directory was checked inside the notebook to confirm its structure. The root directory contains the data.yaml file, image and label folders, and a visualizations folder used to inspect generated samples. This check was done before training to ensure YOLOv8 could correctly load the data.

The dataset configuration was defined in a data.yaml file. This file specifies the dataset path, training and validation image directories, number of classes, and class names. The dataset contains 39 object classes, each represented by an object ID such as OBJ001, OBJ021, and similar identifiers. These class IDs are used directly by the model during training and inference.

Images were split into training and validation sets. The notebook explicitly counts the number of images in each split by scanning the visualization directories. The training set contains 100 images, and the validation set contains 20 images, giving a total of 120 generated multi-object images.

For every object placed in an image, bounding box annotations were generated and saved in YOLO format using normalized center coordinates, width, and height. Each bounding box is associated with one of the 39 class IDs defined in the dataset configuration.

Image resizing and augmentation were handled internally by the YOLOv8 training pipeline. No manual resizing was applied during dataset creation. The dataset was verified before training by printing directory contents, displaying the dataset configuration file, and confirming image counts for both splits.

3. Model Architecture and Training Process

The YOLOv8 object detection framework was used to train a custom multi-object detection model. YOLOv8 follows a single-stage detection approach where object classification and bounding box regression are performed in one forward pass. This allows the model to output object class IDs and their locations directly from an input image.

To understand how model size affects performance, multiple YOLOv8 variants were trained and compared. The models used were YOLOv8 Nano (n), YOLOv8 Small (s), and YOLOv8 Medium (m). These models differ in depth and number of parameters, which impacts detection accuracy, training time, and computational cost.

The Nano model was trained first as a lightweight baseline. It has the fewest parameters and is suitable for fast inference and low-resource environments. The Small model was trained as a balanced option, offering higher accuracy with moderate computational requirements. The Medium model was trained to evaluate how a larger architecture affects detection accuracy when more parameters are available.

All models were trained using the same dataset configuration file, image size, batch size, and number of epochs. The training process used early stopping with a patience value to prevent overfitting. During training, YOLOv8 automatically generated loss curves and performance plots, which were saved for later inspection.

After training each model, validation was performed using the YOLOv8 validation function. For each model, performance metrics were extracted directly from the validation results. These metrics include mAP at IoU 0.5, mAP at IoU 0.5–0.95, precision, recall, and the total number of model parameters. The path to the best-performing model weights was also recorded.

An additional experiment was conducted using an optimized version of YOLOv8 Nano. This model used stronger data augmentation during training, including color augmentation, rotation, translation, scaling, horizontal flipping, and mosaic augmentation. The optimizer was changed to AdamW, and the learning rate was adjusted. Training patience was increased to allow the model more time to converge. This setup was used to evaluate whether performance improvements could be achieved without increasing model size.

All training and validation results were stored programmatically and converted into a comparison table. This table was saved as a CSV file and printed in the notebook for reporting purposes. The trained models are capable of detecting multiple objects in a single image and returning their class IDs and bounding box locations.

Model Name	**mAP@0.5**	**mAP@0.5:0.95**	**Precision**	**Recall**	**Parameters**
YOLOv8 Nano	0.8269	0.8224	0.7665	0.7361	758,917
YOLOv8 Small	0.9868	0.9838	0.9559	0.9537	2,131,141
YOLOv8 Medium	0.9853	0.9836	0.9499	0.9609	3,798,277
YOLOv8 Nano (Optimized)	0.9550	0.9516	0.9012	0.8826	758,917

4. Model Performance Analysis and Interpretation

The comparison plots show how different YOLOv8 model variants perform across multiple evaluation metrics. The metrics compared include mAP@0.5, mAP@0.5:0.95, precision, and recall. These plots were generated using the validation results collected after training each model.

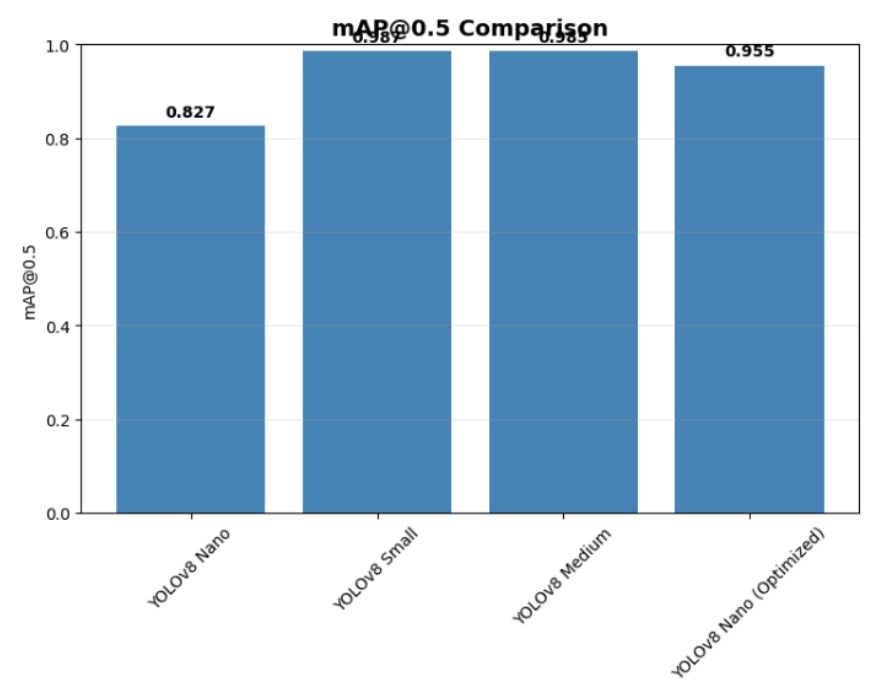
For mAP@0.5, the YOLOv8 Nano model shows the lowest score, while the Small and Medium models achieve much higher values. This indicates that larger models are more accurate at detecting objects when using a fixed IoU threshold. The optimized Nano model improves clearly over the base Nano model, showing that training configuration and augmentation have a strong effect even without increasing model size.

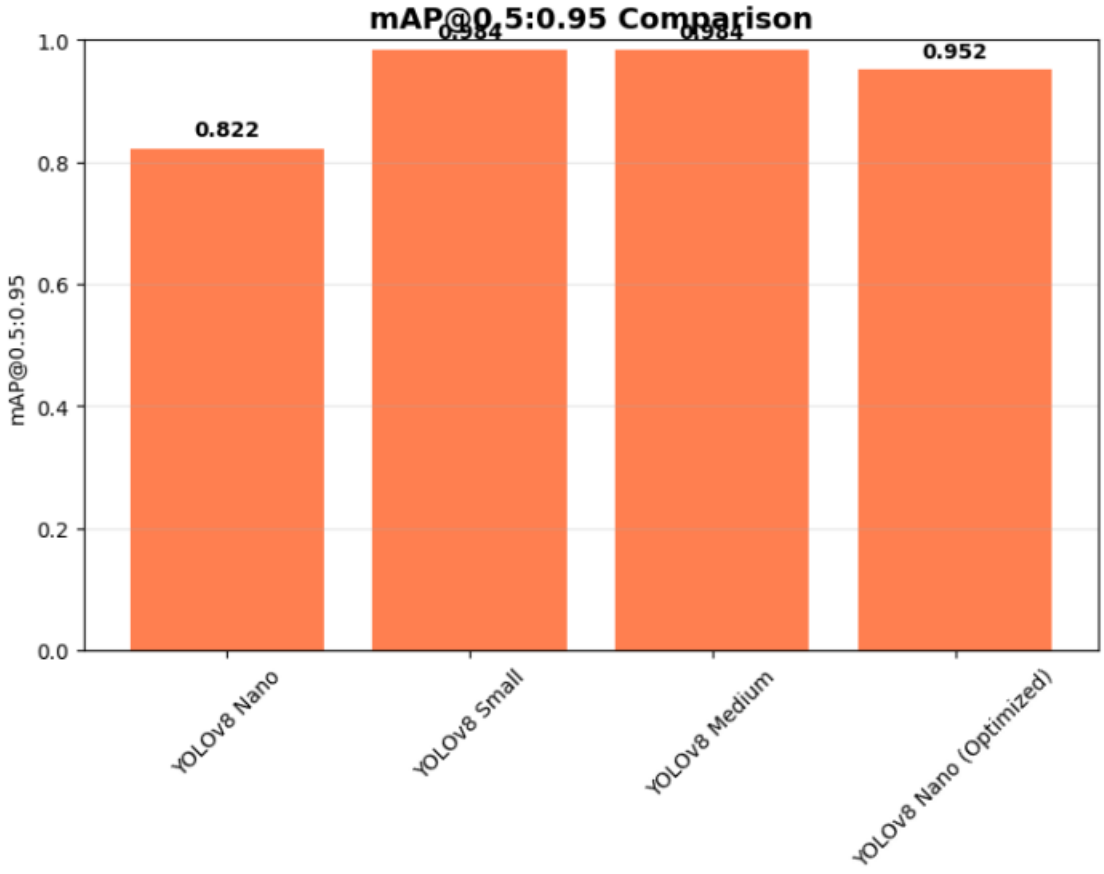
For mAP@0.5,0.95, a similar trend is observed. The Small and Medium models achieve the highest scores, showing consistent detection performance across stricter IoU thresholds. The optimized Nano model again performs better than the standard Nano model but does not reach the accuracy of the larger models.

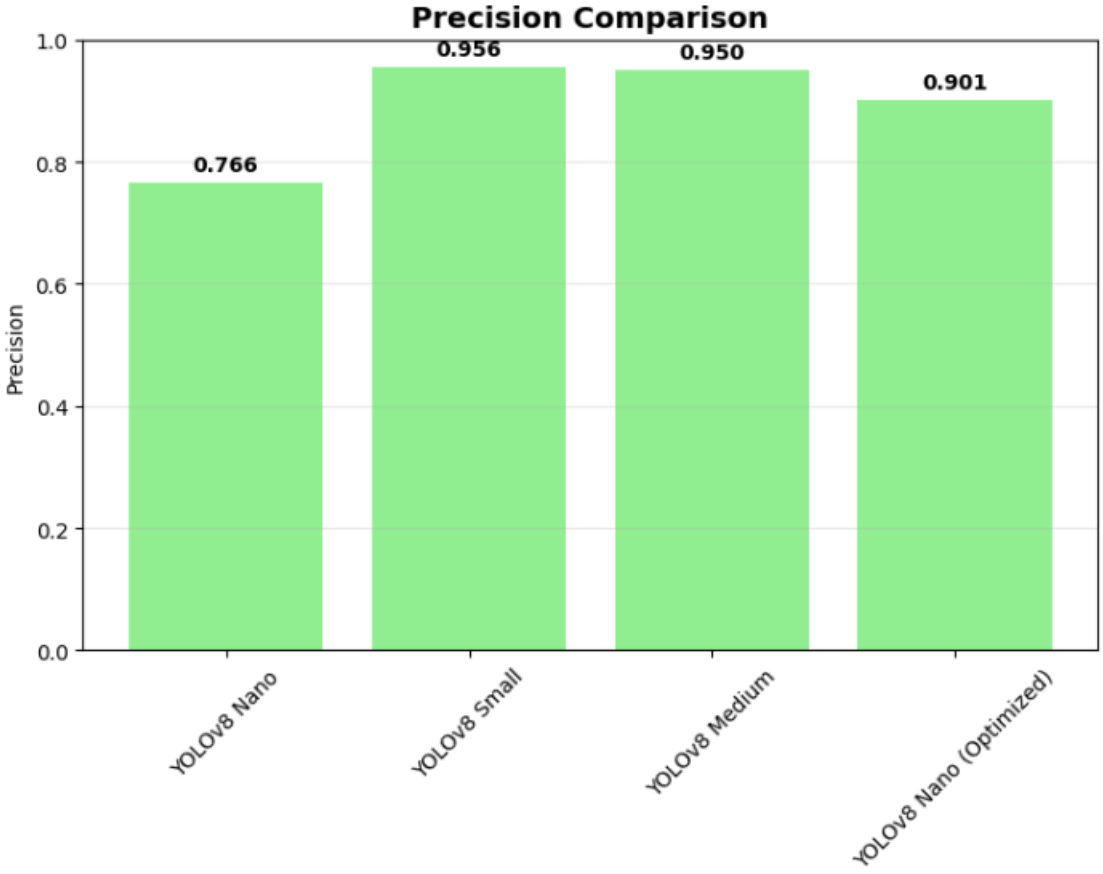
The precision comparison shows that the Small model achieves the highest precision, followed closely by the Medium model. This means these models produce fewer false positive detections. The optimized Nano model improves precision compared to the base Nano model, indicating that augmentation and optimizer changes help reduce incorrect detections.

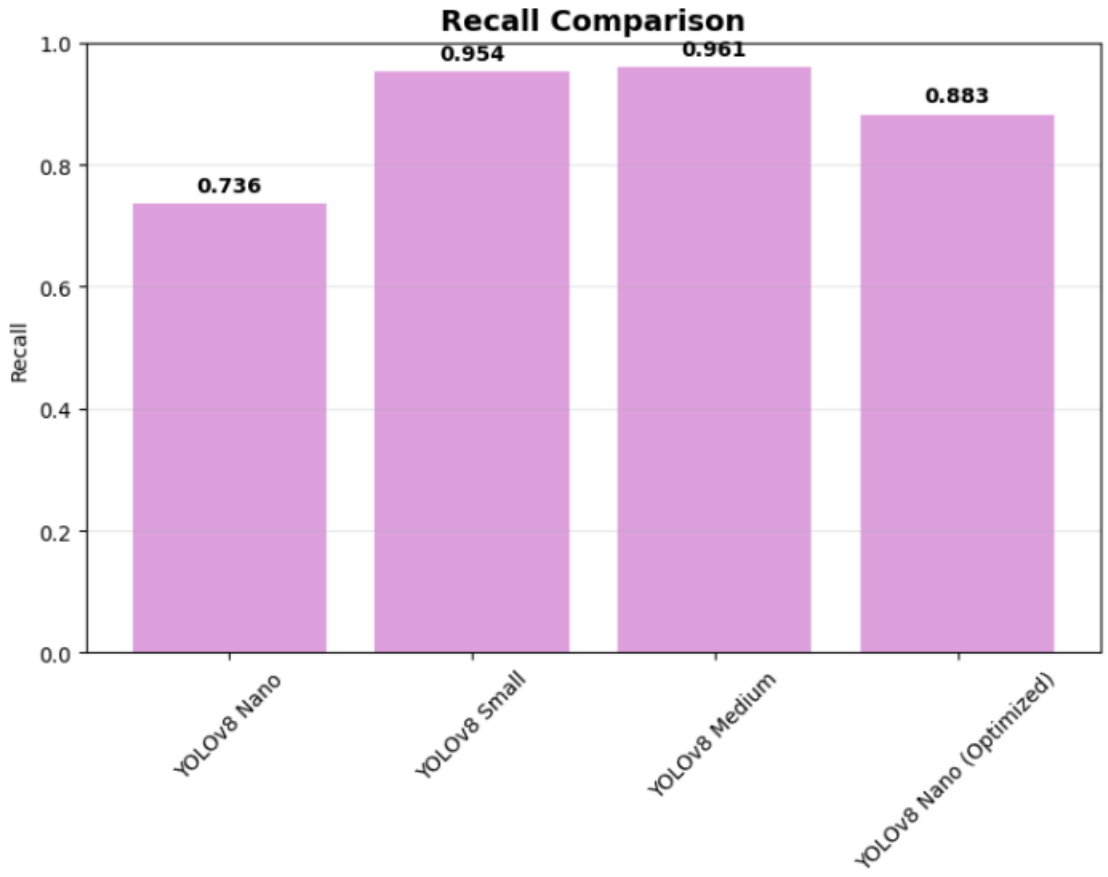
The recall comparison shows that the Medium model achieves the highest recall, meaning it detects a larger proportion of the objects present in the images. The Small model follows closely. The optimized Nano model improves recall over the base Nano model but still misses more objects compared to the larger models.

Overall, the plots show a clear trade-off between model size and detection performance. Larger models achieve higher accuracy, precision, and recall, while optimization techniques allow smaller models to improve performance without increasing parameter count. These results support the use of different YOLOv8 variants depending on accuracy requirements and computational constraints.




mAP@0.5:0.95 Comparison
mAP@0.5:0.95
1.0
0.8
0.6
0.4
0.2
0.0
0.822
0.984
0.984
0.952
YOLOv8 Nano
YOLOv8 Small
YOLOv8 Medium
YOLOv8 Nano (Optimized)


Precision Comparison
Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.766
0.956
0.950
0.901
YOLOv8 Nano
YOLOv8 Small
YOLOv8 Medium
YOLOv8 Nano (Optimized)



5. Best Model Selection

After training and validating all model variants, the best-performing model was selected based on mAP@0.5:0.95. This metric was chosen because it reflects detection performance across multiple IoU thresholds and gives a more complete view of localization quality.

The validation results for all trained models were stored in a results table. The model with the highest mAP@0.5:0.95 score was identified programmatically by comparing this metric across all entries.

Based on this comparison, YOLOv8 Small achieved the highest mAP@0.5:0.95 score and was selected as the best-performing model. It also showed strong performance across all other evaluated metrics.

The selected model achieved an mAP@0.5 of 0.987 and an mAP@0.5:0.95 of 0.984, indicating accurate object detection and bounding box localization. The precision score of 0.956 shows a low rate of false positives, while the recall score of 0.954 indicates that most objects present in the images were detected.

The trained weights for the best model were saved automatically during training and stored at runs/detect/yolov8s_detection/weights/best.pt. This model was later used for inference on multi-object images to output object class IDs and bounding box locations. The best model information was also saved to a text file to keep a record of the selected architecture and its performance metrics.

6. Inference Results on Multi-Object Images

The selected YOLOv8 Small model was used to run inference on five validation images. These images contain multiple objects arranged in a grid, matching the dataset generation setup. Inference was performed using a confidence threshold of 0.25.

For each image, the model outputs bounding boxes, class IDs, and confidence scores. The annotated images show that the model correctly detects multiple objects within a single image and assigns the appropriate object IDs to each detected item.

In Sample 1, the model detected more than ten objects in a single image. All detections were returned with high confidence scores, mostly above 0.95. The bounding boxes align well with the object boundaries, and multiple different class IDs are present in the same image.

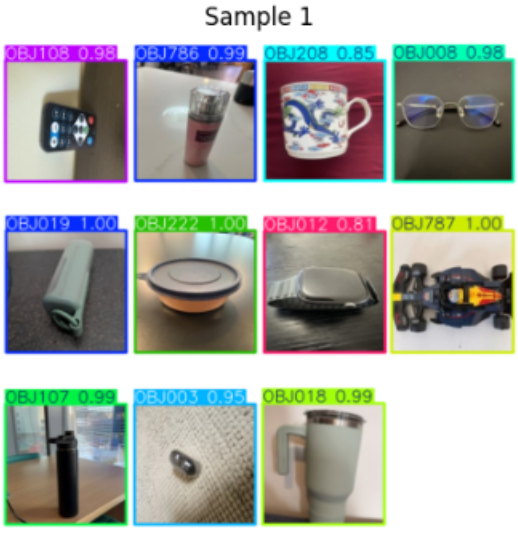
In Sample 2, the model detected repeated object classes along with distinct objects. The output includes multiple detections of the same class when the object appears more than once. Confidence scores remain high, indicating consistent recognition across similar objects.

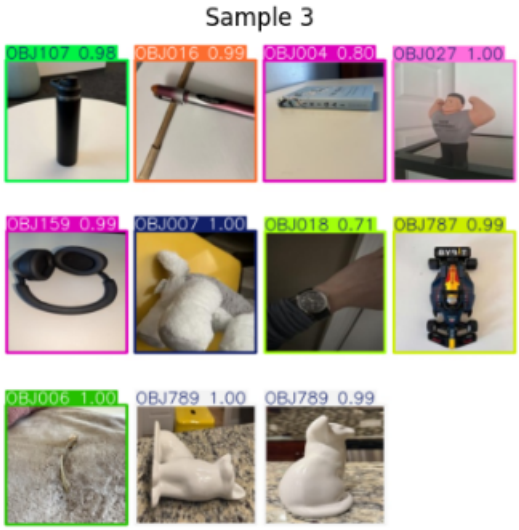
In Sample 3, the image contains a larger number of objects with some visual similarity. The model still detects most objects correctly. A small number of detections appear with lower confidence values, which typically correspond to smaller objects or objects with partial overlap.

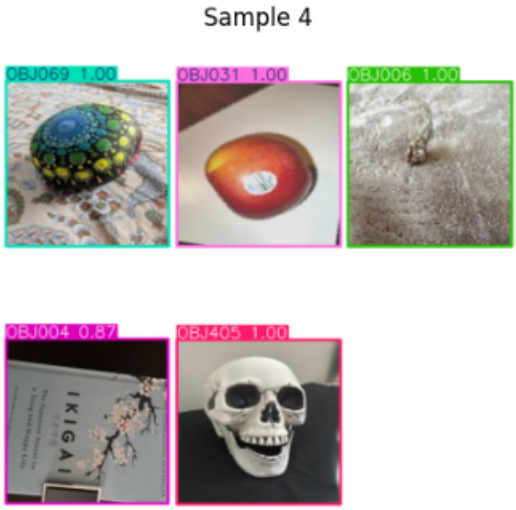
In Sample 4, the model detects fewer objects, and all detections show confidence values close to 1.00. Bounding boxes are tight and correctly localized around each object.

In Sample 5, the model again identifies multiple objects and assigns the correct class IDs. Confidence values remain high, with only one detection falling below 0.8.

Across all samples, inference speed is consistent, with preprocessing, inference, and postprocessing times remaining low per image. The results show that the trained model can take an image containing multiple objects and return the object IDs and bounding box locations for each detected object.







7. Random Validation Image Testing

After selecting YOLOv8 Small as the best model, extra testing was done by running inference on two randomly selected validation images. This was done to confirm the model's predictions on images it was not manually chosen for.

The class name list was loaded from the dataset data.yaml file using YAML parsing. This was used to map predicted class IDs back to the object ID names like OBJ029, OBJ005, and others. The model was loaded from the saved best checkpoint at:

runs/detect/yolov8s_detection/weights/best.pt

All validation images were collected from the validation folder, and two images were randomly sampled from the total validation set. Predictions were run using a confidence threshold of 0.25. For each image, the model returned bounding boxes, class IDs, and confidence values. The output was printed in a readable format showing object name and confidence percent, and an annotated image was displayed with bounding boxes drawn.

Random Image 1: val_0017.jpg

The model detected 13 objects in this image. Most detections were high confidence and clustered close to 99%. Some objects appeared multiple times in the image, and the model returned repeated detections for the same class name where relevant.

Detections printed from the notebook:

- OBJ029 (99.69%)
- OBJ005 (99.29%)
- OBJ229 (99.22%)
- OBJ789 (99.21%)
- OBJ405 (98.83%)
- OBJ107 (98.21%)
- OBJ069 (97.51%)
- OBJ005 (97.08%)
- OBJ005 (96.87%)
- OBJ003 (96.74%)
- OBJ095 (94.17%)
- OBJ788 (93.91%)
- OBJ095 (70.79%)

This sample shows the model can detect many objects in one image and output the object IDs for each detection.

Random Image 2: val_0003.jpg

The model detected 5 objects in this image. Four detections were near 100% confidence, and one detection had a lower confidence score.

Detections printed from the notebook:

- OBJ005 (99.97%)
- OBJ787 (99.72%)
- OBJ069 (99.62%)
- OBJ229 (99.56%)
- OBJ012 (77.43%)

This sample shows the model outputs object IDs consistently and still reports detections with lower confidence when the object is less clear.

These random tests show that the trained model outputs multiple detections per image, and each detection includes the object ID and the location shown through the bounding box overlay.

8. Conclusion

In this milestone, a multi-object detection system was successfully trained using YOLOv8. A custom dataset with multiple objects per image was generated and correctly annotated. Different YOLOv8 model sizes were trained and compared, and YOLOv8 Small was selected based on validation performance. Inference results on validation and random images show that the model can identify object IDs and locate multiple objects within a single image. The project meets the milestone requirements for dataset generation, model training, and multi-object detection output.