Optimizing Data Augmentation Strategies for YOLOv8-Based Object Detection in Robotic Applications: A Case Study with the Toyota HSRC

Colin Gibbons-Fly

Table of contents

Introduction

Context and Lab Overview

The Robocanes Lab at the University of Miami is a research team dedicated to advancing research in the fields of robotics and artificial intelligence. The team specializes in at-home tasks such as having a robot clean or maneuver items, while also conducting research in a variety of other fields, including Human Robot Interaction (HRI) and Computer Vision (CV).

Research Focus

In this report, we will be analyzing the lab's ongoing research on the impact of integrating open-source CV models for object detection in relation to localization and mapping.

Significance of Open-Source Models

The benefit of having open-source models for computer vision tasks has greatly accelerated research in the field for several reasons. First, open-source models democratize access to cutting edge technology, allowing a wider range of individuals and organizations to participate [1]. Second, these models are cost-effective, as organizations would likely have to spend substantially more on software development if proprietary options were the only available choice [1]. Third, transparency in open-source models makes it easier to identify and address biases in AI systems, enhancing accountability and trust in computer vision applications [1][2]. Finally, open-source models offer flexibility and customization, providing developers the ability to modify and adapt the code to suit specific needs [1].

Role of Data Augmentation

Data augmentation plays a critical role in robotics and computer vision by enhancing model performance through artificially expanding datasets. This expansion allows models to learn from a broader range of scenarios, leading to better adaptability and reduced errors in real-world applications [3]. Data augmentation also addresses data scarcity, an issue often encountered in robotics and computer vision due to time, cost, and ethical constraints, by creating new training examples from existing data, effectively increasing dataset size and diversity without additional data collection [3][5]. Recent advancements, such as the RoVi-Aug framework, leverage data augmentation to facilitate the transfer of skills across different robots, enabling skills learned on one robot to be applied to another with different characteristics [4]. Finally, data augmentation improves generalization by combating overfitting, introducing randomness and variability into the training data to create more robust models that perform well in diverse real-world conditions, which is crucial for applications like autonomous driving [3].

Objective

The objective of this report is to evaluate how different levels of augmentation affect the performance of our YOLOv8 model deployed on the Toyota HSRC, with a particular focus on determining the optimal augmentation strategy for localized object detection.

Dataset Preparation and Labeling

Dataset Collection

Data was collected manually using an iPhone. Pictures were taken while carefully replicating the angle and eye level of the Toyota HSRC, and additional photos were captured from higher eye levels to ensure variety. To capture different perspectives, images were taken at varying depths, some close to the object and others from a greater distance. Variations in photo quality were also introduced to simulate the occasional blurring caused by the robotic camera in motion. To address lighting conditions, photos were taken with the lights on and off, accounting for light pollution. Lastly, many images included multiple objects, allowing the model to learn to detect and localize more than one object simultaneously.

Dataset Details

The curated dataset comprises a total of 650 images, which are divided into a training set of 520 images (80%) and a test set of 120 images (20%).

Classes

- ginger
- crate
- 2010_poster
- 2016_poster
- exit
- fire_alarm
- light_horiz
- light_plain
- light_plus
- light_vert

Data Augmentation Strategy

Dataset Versions

Version	Augmentations
baseline	None
min_aug	2 outputs per training example, Flip: Horizontal, 90° Rotate: Clockwise,Hue: Between -5° and +5°,Brightness: Between -5% and +5%
mod_aug	2 outputs per training example, Flip: Horizontal, 90° Rotate: Clockwise,Crop: 0% Minimum Zoom, 20% Maximum Zoom,Shear: ±10° Horizontal, ±10° Vertical,Hue: Between -10° and +10°,Brightness: Between -10% and +10%,Exposure: Between -10% and +10%
heavy_aug	2 outputs per training example, Flip: Horizontal, 90° Rotate: Clockwise, Crop: 0% Minimum Zoom, 25% Maximum Zoom, Shear: ±15° Horizontal, ±15° Vertical, Hue: Between -20° and +20°, Brightness: Between -20% and +20%, Exposure: Between -15% and +15%, Blur: Up to 2.5px, Noise: Up to 0.3% of pixels
geo_aug	2 outputs per training example; Flip: Horizontal, 90° Rotate: Clockwise, Crop: 0% Minimum Zoom, 25% Maximum Zoom, Shear: ±10° Horizontal, ±10° Vertical
pho_aug	2 outputs per training example; Hue: ±10°; Brightness: ±10%, Exposure: Between -10% and +10%, Blur: Up to 2px, Noise: Up to 0.25% of pixels

Hypotheses and Discussion

Hypothesis

Given the well-documented benefits of data augmentation in enhancing the robustness and generalization of computer vision models, our initial hypothesis was that increasing the level of augmentation would lead to improved classification results. In other words, as more augmentation transformations are applied, the model should better generalize to varied real-world conditions, thereby boosting its performance on the testing set.

Null Hypothesis

Contrary to our expectation, an alternative hypothesis was that less augmentation might lead to better results. This could occur if excessive augmentation distorts key features or introduces too much variability, causing the model to diverge from the real data distribution.

Experimental Setup

Model Details

YOLOv8 is the latest iteration of the You Only Look Once family of object detection models and introduces several architectural improvements to enhance speed, accuracy, and efficiency. Its design is organized into three main components:

Backbone

YOLOv8 utilizes a custom CSPDarknet53 backbone for feature extraction. This backbone employs cross-stage partial connections to improve information flow between layers, thereby enhancing accuracy [6][7]. Key optimizations include:

- A reduced first convolutional kernel size in the stem
- An upscaled first convolutional kernel in the backbone bottleneck
- Replacement of the C3 block with a more efficient C2f module [8]

Neck

Instead of the traditional Feature Pyramid Network (FPN), YOLOv8 implements a novel C2f module for its neck structure. This module effectively combines high-level semantic features with low-level spatial information, improving detection accuracy—especially for small objects [7][8].

Head

The detection head is responsible for making predictions by employing multiple detection modules that predict bounding boxes, objectness scores, and class probabilities for each grid cell in the feature map. Notable innovations include:

- An anchor-free approach, eliminating the need for predefined anchor boxes
- A modified loss function that focuses on both bounding box location and classification confidence [6][8]

These architectural enhancements contribute to YOLOv8's improved accuracy, faster speeds, and reduced number of parameters compared to its predecessors, making it a highly efficient and advanced object detection model.

Rationale for Model Choice

Real-Time Performance: YOLOv8 excels in real-time object detection, achieving speeds of up to 80 frames per second while maintaining high accuracy [9][10]. This rapid processing is crucial for robotics applications that require instant feedback and decision-making.

Improved Accuracy: With an average precision (AP) of 53.9%, YOLOv8 demonstrates superior accuracy compared to earlier models and alternatives like Faster R-CNN [10]. This high level of accuracy is vital for precise object identification in dynamic robotic environments.

Efficient Architecture: The streamlined, single neural network design of YOLOv8 eliminates the need for complex post-processing steps [9]. This efficiency is particularly beneficial for robotics projects where computational resources are limited.

Flexibility and Adaptability: YOLOv8's anchor-free design allows for more accurate and efficient detection of objects in dynamic environments. Its support for multiple backbones also provides flexibility in tailoring the model configuration to specific use cases [11][12].

Enhanced Features for Robotics: Beyond basic object detection, YOLOv8 offers capabilities such as instance segmentation and pose estimation, which are valuable for advanced robotics applications including collaborative robotics and complex manipulation tasks [13].

Ultralytics API Configuration

The YOLOv8 model was configured using the Ultralytics API. For each dataset version, the model parameters were consistent except for the specific variations required by the dataset configurations. This streamlined setup allowed for rapid iteration and fine-tuning while maintaining consistency across experiments. The following parameters were held constant across all tests:

- Epochs: 200
- Learning Rate: 0.01
- Batch Size: 16
- Image Size: 640
- Confidence Threshold for Inference: 0.25

This controlled training procedure ensured that any observed differences in performance could be attributed to the varying levels of data augmentation rather than other training variables.

Results and Analysis

Training Results

Version	box_loss	cls_loss	dfl_loss
baseline	0.520	0.300	0.879
mid	0.453	0.250	0.863
mod	0.503	0.266	0.880
heavy	0.483	0.254	0.873
geo	0.457	0.238	0.864
pho	0.369	0.204	0.834

Precision, Recall, mAP50

Version	precision	recall	mAP50	mAP50-95
baseline	0.907	0.911	0.942	0.788
mid	0.832	0.932	0.921	0.757
mod	0.857	0.949	0.942	0.774
heavy	0.854	0.971	0.949	0.787
geo	0.905	0.880	0.926	0.764
pho	0.883	0.908	0.934	0.762

Validation Results

Version	box_loss	cls_loss	dfl_loss
baseline	0.688	0.370	0.936
mid	0.716	0.357	0.940
mod	0.708	0.357	0.936
heavy	0.698	0.336	0.948
geo	0.723	0.339	0.946
pho	0.730	0.370	0.957

Insights and Analysis

In analyzing the performance metrics of the object detection project, several key points emerge. The Baseline and Geo models demonstrate strong precision, achieving values above 0.90 (0.90734 and 0.90511 respectively). In contrast, the Heavy and Mod models show lower precision values (approximately 0.85375 and 0.85702), suggesting that these configurations may be sacrificing some precision for other performance gains.

When evaluating recall, the Heavy model stands out with an exceptionally high value of 0.97104, while the Mod model follows closely at 0.94913. The Baseline, Geo, and Pho models maintain a consistent performance with recall figures around 0.91, and the Mid model records a slightly lower recall of 0.93185. This disparity indicates that the Heavy model is particularly effective at detecting true positives.

Looking at the mAP50 metric, the Heavy model again leads with a score of 0.94937, closely trailed by the Baseline and Mod models, both around 0.942. The Mid model, however, lags behind with a mAP50 of 0.92053. In terms of mAP50-95, which provides a more comprehensive evaluation by averaging precision across different IoU thresholds, both the Baseline and Heavy models perform best with scores of 0.78844 and 0.78734, respectively. The remaining experiments range from about 0.75733 (Mid) to approximately 0.76196–0.764 for Geo and Pho.

Overall, these metrics point to distinct strengths and trade-offs among the models. The Pho model, while achieving the best training losses, suffers from higher validation losses in box and DFL metrics and slightly lower precision, indicating potential overfitting. The Heavy model, with its excellent recall and mAP50, excels at detecting true positives but at the expense of some precision. In contrast, the Baseline and Geo models offer a more balanced performance with high precision and competitive validation losses, with Baseline having a slight edge in terms of validation box loss and mAP50-95.

In conclusion, if the primary goal is to maximize recall and overall detection accuracy, the Heavy model appears most promising, despite its compromise in precision. However, if a balanced approach between precision and recall is preferred, the Baseline (or possibly Geo) configuration would be more appropriate. Additionally, the Pho model's performance suggests that further regularization may be needed to improve its generalization on unseen data.

References

[1] King, A. (2024, May). Op-ed: The benefits of open source AI. Fieldston News.https://fieldstonnews.com/home/2024/05/op-ed-the-benefits-of-open-source-ai/

[2] Moesif. (n.d.). Open source AI.https://www.moesif.com/blog/technical/api-development/Open-Source-AI/

[3] Cosmo Tech Blog. (n.d.). What is data augmentation in machine learning?https://techblog.cosmobc.com/what-is-data-augmentation-in-machine-learning/

[4] TechXplore. (2024, October). Augmentation algorithm transfers skills across robots.https://techxplore.com/news/2024-10-augmentation-algorithm-skills-robots.html

[5] IBM. (n.d.). Data augmentation.https://www.ibm.com/think/topics/data-augmentation

[6] YOLOv8.org. (n.d.). YOLOv8 architecture.https://yolov8.org/yolov8-architecture/

[7] Labellerr. (n.d.). Understanding YOLOv8 architecture, applications, and features.https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/

[8] Viso.ai. (n.d.). YOLOv8 guide.https://viso.ai/deep-learning/yolov8-guide/

[9] Labelvisor. (n.d.). YOLOv8 vs. Mask R-CNN: In-depth analysis and comparison.https://www.labelvisor.com/yolov8-vs-mask-r-cnn-in-depth-analysis-and-comparison/

[10] estack. (n.d.). AI for industrial automation: Answering YOLOv8 applications in Cat AI.https://www.restack.io/p/ai-for-industrial-automation-answer-yolov8-applications-cat-ai

[11] Keylabs. (n.d.). Under the hood: YOLOv8 architecture explained.https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/

[12] Ultralytics. (n.d.). YOLOv8 vs. YOLOv5.https://docs.ultralytics.com/compare/yolov8-vs-yolov5/