Efficiency of YOLOv8 and Zero-Shot OWL-ViT on a Kitchen Utensil Detection Task

Michael Hamaty
San Jose State University

December 17, 2025

1 Introduction

Object detection plays a central role in modern computer vision, though the need for large, labeled datasets is a massive obstacle in making custom applications. While state-of-the-art models like YOLOv8 achieve impressive results on standard benchmarks, their performance in low-data regimes is often insufficient. Conversely, OWL-ViT proposes "zero-shot" capabilities, promising detection without explicit fine-tuning. This report investigates the sample efficiency of the YOLOv8 architecture by fine-tuning a small model (YOLOv8n) on increasingly larger subsets of a "Common Kitchen Utensils" dataset. I specifically analyze the trade-off between dataset size (k) and mean Average Precision (mAP). Furthermore, I benchmark these fine-tuned models against two general baselines: out-of-the-box YOLOv8n pretrained on COCO, and a zero-shot OWL-ViT model.

2 Objective

The primary goal of this project is to explore the effectiveness of few-shot fine-tuning compared to zero-shot inference in the domain of kitchen utensils. The project aims to achieve two key objectives:

1. **Quantify Sample Efficiency:** Determine how detection performance scales as the number of training examples per class (k) increases from 1 to 32.
2. **Identify the "Crossover Point":** Establish the dataset size required for a lightweight, fine-tuned model (YOLOv8n) to match or exceed the performance of a massive, zero-shot foundation model (OWL-ViT).

3 Experimental Design

3.1 Dataset Description

The study utilizes the *Common Kitchen Utensils* dataset withdrawn from Roboflow, comprising 10 classes: *bowl, fork, glass, knife, mug, pan, plate, spatula, spoon,* and *whisk*. The dataset contains approximately 8,000 labeled images. To strictly control for sample size, I implemented a custom stratified sampling strategy, creating four distinct training subsets where $k \in \{1, 4, 16, 32\}$ images are selected per class.

3.2 Model Architectures

- **YOLOv8n (Fine-Tuned):** A lightweight Convolutional Neural Network (CNN) designed for real-time detection. This model is fine-tuned on the processed k-shot datasets.
- **YOLOv8n (COCO Pretrained):** The same architecture but without fine-tuning, used to assess the transferability of standard pre-trained weights to this specific domain.
- **OWL-ViT (Zero-Shot):** An Open-Vocabulary Vision Transformer trained on web-scale image-text pairs. It detects objects using natural language prompts (e.g., "a photo of a spatula") rather than fixed class labels.

4 Implementations

4.1 Dataset Pre-processing and Splitting

To ensure rigorous evaluation, a custom test split was generated comprising 20% of the total dataset, stratified by class to ensure all the models are evaluated on the exact same unseen data. The remaining 80% of the data formed the pool from which the k-shot training sets were sampled. All images were resized to a uniform dimension of 640×640 pixels for consistency and computational efficiency during training.

4.2 YOLOv8n Fine-Tuning Configuration

The fine-tuning process utilized the Ultralytics YOLOv8 implementation. The training configuration was standardized across all k values to isolate the impact of dataset size:

- **Epochs:** 10
- **Batch Size:** 16
- **Optimizer:** defaults from Ultralytics YOLOv8
- **Augmentation:** Mosaic augmentation was enabled to improve robustness in the low-data regime.

4.3 Zero-Shot Baseline Configuration

For the OWL-ViT baseline, I utilized the `google/owlvit-base-patch32` model. Since OWL-ViT requires text prompts, I constructed simple prompts in the format "a photo of a <class>" for each of the 10 classes. For the standard YOLOv8n (COCO) baseline, I mapped COCO classes to our dataset labels where possible to keep mapping as consistent as possible. Classes without direct COCO equivalents (e.g., "whisk", "spatula") were expected to yield near-zero performance.

5 Comparison and Analysis

5.1 Quantitative Results

Table 1 summarizes the performance of the fine-tuned YOLOv8n models across different k values.

Table 1: YOLOv8n Performance vs. Training Set Size (k)

k (Images/Class)	Total Train Images	mAP@0.5	mAP@0.5:0.95	Train Time (min)
1	10	0.030	0.015	15.3
4	40	0.083	0.043	15.3
16	160	0.317	0.192	15.6
32	320	**0.460**	**0.278**	15.9

As shown in Figure 1, there is a logarithmic relationship between dataset size and model performance. At $k = 1$ and $k = 4$, the model fails to learn effective features, performing similarly to a random guesser. A significant performance jump occurs at $k = 16$, suggesting a learning threshold where the model begins to generalize.

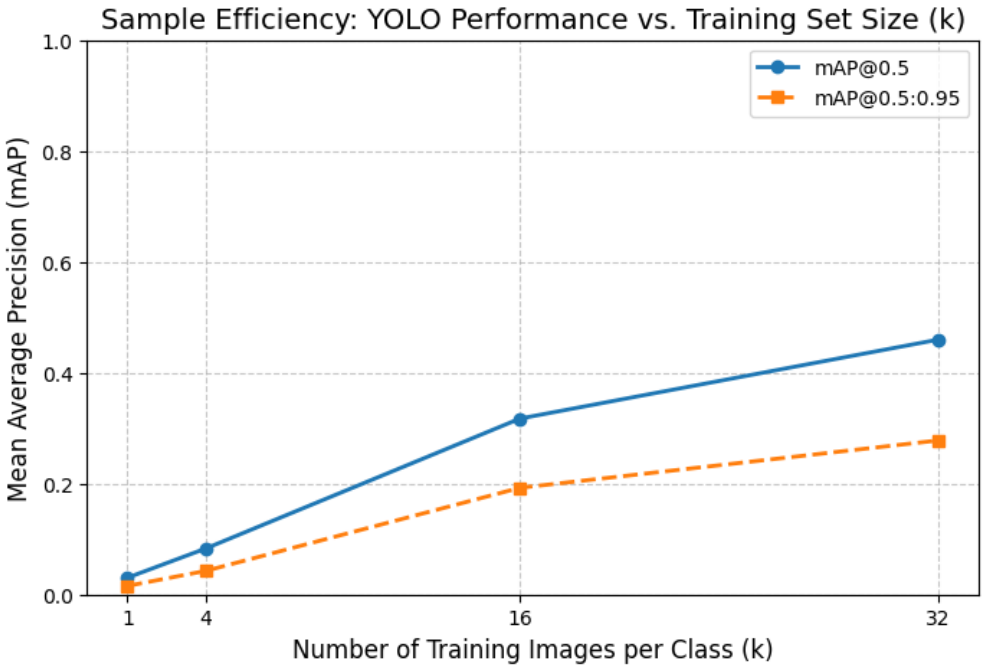

Figure 1: Sample Efficiency: mAP@0.5 performance scaling with number of training images per class (k).

5.2 Baseline Comparison

The fine-tuned results were compared against the two baselines. The COCO-pretrained YOLOv8n achieved an overall mAP@0.5 of **0.081**. This poor performance confirms that general pre-training is insufficient for specialized kitchen utensils, particularly for non-COCO classes like "whisk" and "spatula."

In contrast, the zero-shot OWL-ViT model achieved an mAP@0.5 of **0.460**. As shown in Table 1, the fine-tuned YOLOv8n matches this performance exactly at $k = 32$. This indicates that **32 labeled examples per class** is the break-even point where a small, specialized model catches up to a large foundation model in this specific domain.

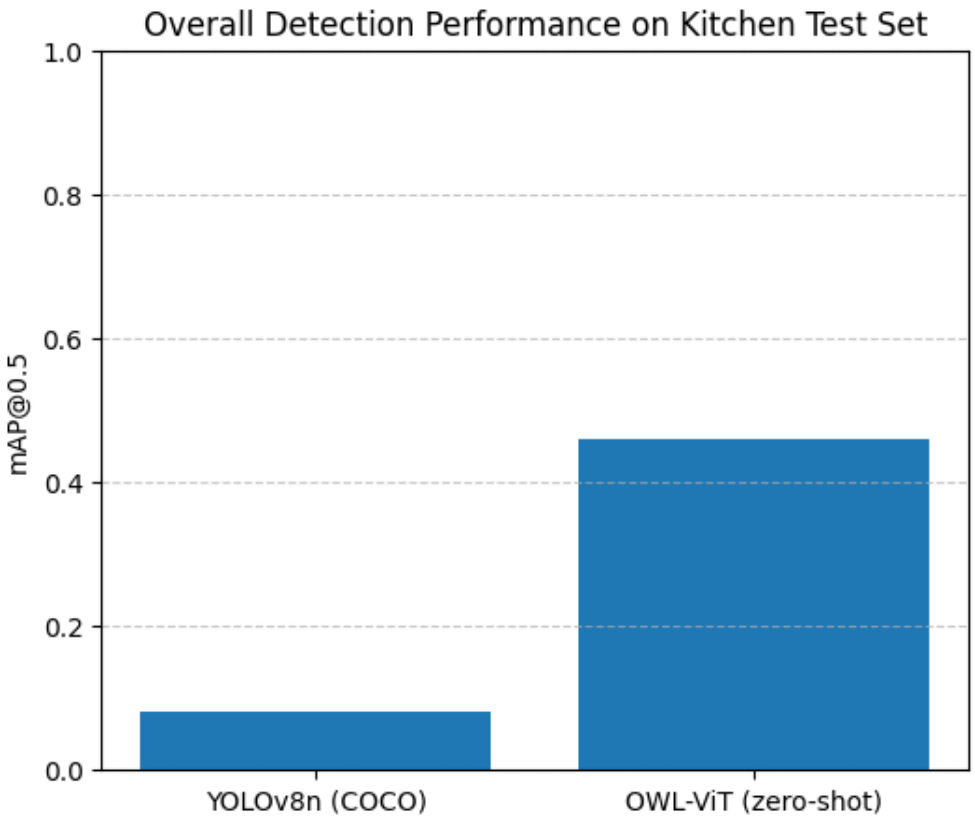

Figure 2: Performance comparison of Zero-Shot baselines (YOLO-COCO vs. OWL-ViT).

5.3 Class-Level Analysis

Figure 3 illustrates the per-class Average Precision (AP) for the best performing fine-tuned model ($k = 32$).

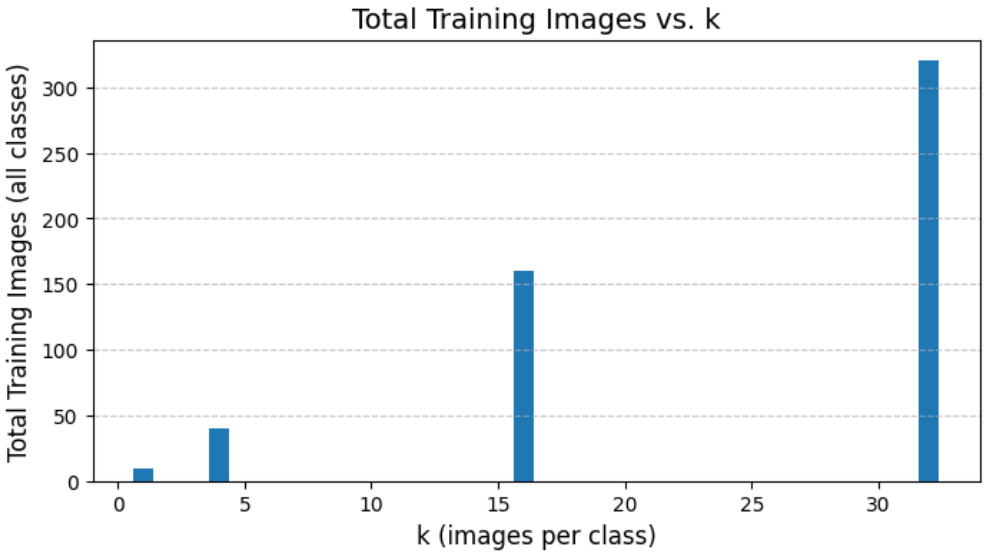

Figure 3: Per-Class AP for the best fine-tuned YOLO model ($k = 32$).

Distinct performance tiers are observable:

- **High Performers:** Classes like "Pan" and "Mug" achieved high AP scores. These objects tend to be large, rigid, and distinctively shaped, making them easier for the CNN to feature-map.
- **Low Performers:** "Glass" consistently performed poorly. This is likely due to the transparent nature of the object, which lacks strong texture cues for the convolutional filters.

5.4 Discussion: OWL-ViT Failure Cases

While OWL-ViT performed strongly overall, it exhibited near-zero AP for "pan" (0.03) and "spatula" (0.09). I hypothesize three primary reasons for this failure:

1. **Open-Vocabulary Mismatch:** OWL-ViT is trained on web-scale, iconic imagery. The visual distribution in our dataset (cluttered, overhead kitchen views) likely represents a significant distribution shift from its training data.
2. **Shape Ambiguity:** Spatulas are thin, elongated objects often occluded by hands or other dishes. Without specific training on these occlusions, the zero-shot model struggles to localize the object boundaries accurately.
3. **Prompt Sensitivity:** I utilized simple prompts ("a photo of a spatula"). Ambiguous classes often require prompt engineering (e.g., "a metal spatula used for frying") to activate the correct attention heads in the Transformer, which I did not performed in this study.

6 Result

The experiments demonstrate that fine-tuning a lightweight model is a highly data-efficient strategy for this domain. While the zero-shot OWL-ViT model provides a strong baseline (mAP 0.46), the YOLOv8n model was able to match this performance with only 320 total labeled images (32 per class) and approximately 45 minutes of training time on a single GPU. Furthermore, standard pre-trained models (YOLO-COCO) proved ineffective (mAP 0.08), highlighting that "generic" object detection weights do not transfer well to specific tools without at least some fine-tuning.

7 Conclusion

This project successfully quantified the sample efficiency of YOLOv8 for kitchen utensil detection. I identified that for this specific task, **32 images per class** represents the crossover point where the cost of labeling data yields a model as capable as a state-of-the-art zero-shot Transformer. Future work could explore a "human-in-the-loop" approach, using OWL-ViT to auto-label the first 32 images to bootstrap the YOLO training process, thereby combining the zero-shot capabilities of Transformers with the inference speed of CNNs. I initially hoped to implement this type of approach within this project after taking about 150 of my own photos, but discovered that such tools are not yet readily available in existing frameworks.