# AdaptiVision: Scene-Adaptive Confidence Thresholding for Object Detection - Performance Analysis on COCO

Abhilash Chadhar

Axelera AI / Indian Institute of Space Science and Technology

abhilashchadhar@gmail.com

Kartik Chadhar

All India Institute of Medical Sciences (AIIMS), Delhi

kartikchadhar@gmail.com

April 13, 2025

## Abstract

Fixed confidence thresholds in object detection models limit robustness across diverse visual scenes. This paper investigates AdaptiVision, a post-processing technique designed to enhance detection by dynamically adjusting the confidence threshold for a YOLOv8n model based on scene complexity heuristics (object count, size variance, density) derived from initial low-confidence detections. Optional rule-based context awareness and class-specific biases can further modulate the threshold. We perform a rigorous evaluation on the COCO 2017 validation dataset (5000 images) using standard mAP metrics. The results show that while AdaptiVision increases the average number of detected objects per image (6.43 vs. 4.95) and improves average processing speed by nearly 1.5x compared to the standard fixed-threshold approach (0.024s vs. 0.034s on test hardware), it leads to a decrease in overall detection accuracy (mAP@[.50:.95]: 0.340 vs. 0.355 baseline; AR@[.50:.95]: 0.428 vs. 0.464 baseline). This performance degradation, despite identifying more objects faster, highlights the limitations of the current heuristic approach in effectively balancing precision and recall according to standard COCO metrics. The study underscores the challenge of designing adaptive heuristics that generalize well and the importance of comprehensive evaluation beyond simple object counts or speed metrics.

## 1   Introduction

Object detection—the task of identifying and localizing objects within an image—is a cornerstone of modern computer vision, enabling applications such as autonomous driving, robotics, surveillance, and image retrieval. State-of-the-art detectors, particularly those based on deep learning like the YOLO family [1, 2, 11, 12], typically assign a confidence score to each potential detection. Filtering these potential detections using a confidence threshold is crucial.

The common practice involves setting a single, fixed confidence threshold. While straightforward, this approach faces limitations: a threshold optimized for simple scenes might miss detections in complex ones (low recall), while a lower threshold increases false positives in simple scenes (low precision). This inflexibility motivates research into adaptive methods.

We analyze AdaptiVision, a post-processing methodology attempting to adapt the confidence threshold based on scene characteristics estimated directly from initial YOLOv8n [3] detection results using heuristics (object count, size variance, density). AdaptiVision operates as a lightweight layer applied after initial inference.

This paper presents a rigorous evaluation using standard COCO mAP metrics on the full COCO 2017 validation dataset (val2017, 5000 images) [4]. The core contributions are:

1. Analysis of a heuristic method for estimating scene complexity from initial detections.

2. Evaluation of a dynamic threshold mapping mechanism based on this complexity.

3. Assessment of optional rule-based context and class biases integration.

4. An experimental evaluation on COCO val2017 using standard mAP metrics, comparing AdaptiVision to a baseline.

5. An analysis of the trade-offs observed between object counts, processing speed, and standard accuracy metrics (mAP, AR).

This paper is structured as follows: Section 2 discusses related work. Section 3 details the AdaptiVision methodology. Section 4 outlines the experimental setup and results on COCO val2017. Section 5 discusses the implications and limitations, and Section 6 concludes the paper.

## 2 Related Work

The challenge of optimizing object detection performance across diverse scenes has motivated research beyond standard architectures with fixed parameters. Several avenues have been explored to make detection more adaptive and context-aware.

**Adaptive Non-Maximum Suppression (NMS):** Standard NMS uses a fixed Intersection over Union (IoU) threshold to suppress redundant bounding boxes. Recognizing that optimal suppression might depend on object density, methods like Adaptive NMS [5] propose adjusting the IoU threshold dynamically based on the density of nearby detections. Soft-NMS [6] offers an alternative by decaying the scores of overlapping boxes instead of eliminating them outright. While related to adapting parameters based on detection characteristics, these methods focus on the NMS stage, whereas AdaptiVision targets the initial confidence filtering threshold *based on broader scene complexity metrics rather than just local density for NMS*.

**Confidence Calibration:** Some research focuses on improving the reliability of the confidence scores output by detectors, ensuring they better reflect the true probability of a correct detection [10]. While valuable, calibration typically aims to improve the *interpretation* of existing scores rather than dynamically changing the *filtering threshold* based on scene properties *in response to overall scene complexity*.

**Context-Aware Object Detection:** Incorporating scene context can significantly improve detection accuracy. Early approaches often used graphical models or statistical co-occurrence information. More recent deep learning methods, such as Relation Networks [7], explicitly model object relationships using dedicated network modules (e.g., attention mechanisms, graph neural networks) to learn contextual dependencies. These approaches often require significant architectural changes and additional training data. AdaptiVision incorporates a simpler, optional form of context using predefined, rule-based object relationships applied during its post-processing threshold adjustment, avoiding architectural modifications *and focusing primarily on complexity-driven thresholding rather than learned contextual reasoning as the main driver*.

**Adaptive Confidence Thresholding:** The idea of adjusting the confidence threshold itself has been explored, particularly for specific applications. Lee et al. [8] proposed adjusting thresholds for 3D LiDAR object detection based on the object's distance from the sensor, addressing range-dependent performance variations. In the domain of multi-object tracking, Ma et al. [9] investigated adapting the threshold distinguishing high-confidence and low-confidence detections within the ByteTrack algorithm, using insights from overall detections to potentially improve track association. AdaptiVision differs by focusing on 2D object detection filtering and using a specific combination of object count, size variance, and density from initial detections to derive a scene complexity score, which then directly modulates the filtering threshold for each image, *offering a more general scene complexity approach compared to application-specific adjustments like distance or tracking-specific thresholds*.

**Contribution of AdaptiVision:** Compared to the related work, AdaptiVision's contribution lies in its specific heuristic approach to scene complexity estimation derived directly from initial, low-confidence detections and its use as a lightweight, post-processing module to dynamically adjust confidence thresholds. It combines complexity analysis, class-specific biases, and optional rule-based context in a unified framework that can be integrated with standard object detectors without retraining or architectural changes. This work evaluates the effectiveness of this specific heuristic approach using standard benchmarks.

# 3 Methodology

AdaptiVision operates as a post-processing refinement layer applied to the output of a base object detector. In this work, YOLOv8n [3] serves as the base detector. The core idea is to replace the standard fixed confidence threshold filtering with a dynamic process informed by scene characteristics. The workflow proceeds as follows (See Figure 1 for an architectural overview):
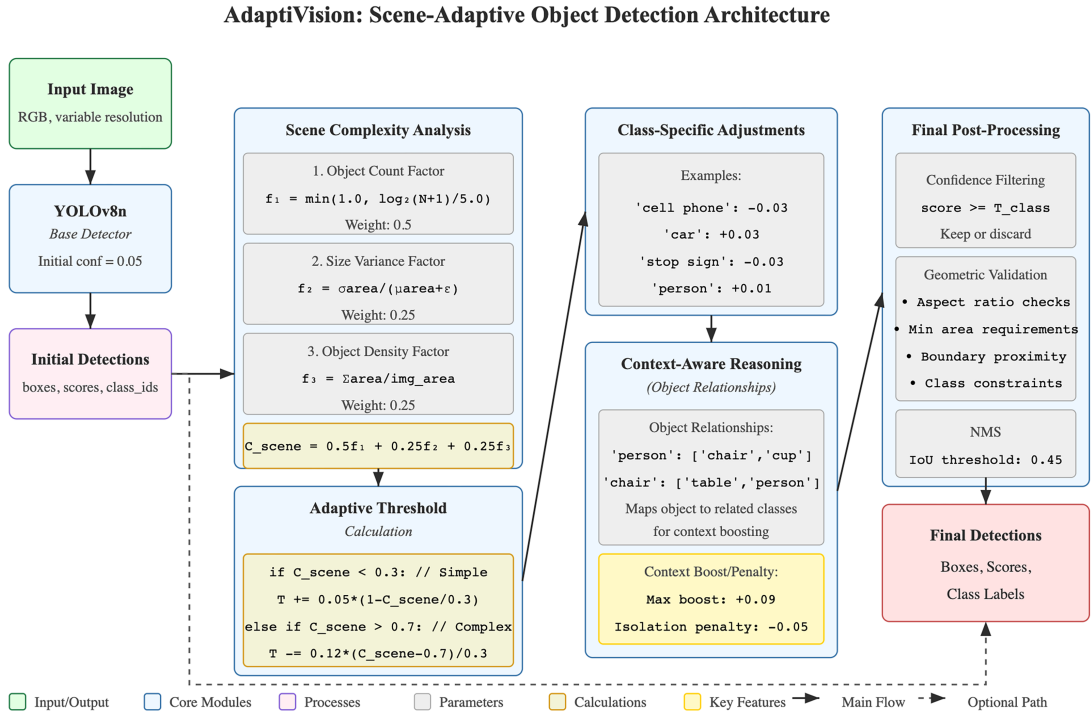


Figure 1: Overview of the AdaptiVision processing pipeline. An input image is processed by a base detector (YOLOv8n) using a low initial threshold. Scene complexity is analyzed from these initial detections, which informs the calculation of a dynamic adaptive threshold. Optional class-specific and context adjustments further refine the threshold before final filtering and NMS produce the output detections.

## 3.1 Initial Low-Confidence Detection

First, the base YOLOv8n detector processes the input image using a very low confidence threshold (e.g., 0.05). This step aims to capture a comprehensive set of potential object candidates, including true positives that might otherwise be discarded by a higher, fixed threshold, especially in challenging scenes. This produces an initial list of candidate bounding boxes, scores, and class labels.

## 3.2  Scene Complexity Calculation

Using the results from the initial detection pass, AdaptiVision calculates a scene complexity score $C_{\text{scene}} \in [0, 1]$. This score heuristically estimates scene challenge using metrics chosen for their intuitive correlation with detection difficulty and computational simplicity. Specifically:

1. **Normalized Object Count ($f_{\textbf{count}}$):** The number of initial detections ($N_{\text{obj}}$) is normalized using a logarithmic scale to temper the influence of very high counts: $f_{\text{count}} = \min(1.0, \log_2(N_{\text{obj}} + 1)/5.0)$.

2. **Normalized Size Variance ($f_{\textbf{var}}$):** The areas ($A_i$) of the initial bounding boxes are calculated. The normalized standard deviation of these areas (coefficient of variation, $\sigma_{\text{area}}/(\mu_{\text{area}} + \epsilon)$) is computed and scaled: $f_{\text{var}} = \min(1.0, (\sigma_{\text{area}}/(\mu_{\text{area}} + \epsilon))/2.0)$. If $N_{\text{obj}} \leq 1$, $f_{\text{var}} = 0$.

3. **Normalized Object Density ($f_{\textbf{density}}$):** The ratio of the total area covered by initial bounding boxes ($\sum A_i$) to the total image area ($A_{\text{img}}$) is calculated and scaled: $f_{\text{density}} = \min(1.0, (\sum A_i/A_{\text{img}}) \times 3.0)$.

These factors are combined using weights ($w_{\text{count}} = 0.5, w_{\text{var}} = 0.25, w_{\text{density}} = 0.25$) determined empirically through preliminary experiments to balance the contribution of each factor:

$$C_{\text{scene}} = w_{\text{count}} f_{\text{count}} + w_{\text{var}} f_{\text{var}} + w_{\text{density}} f_{\text{density}}$$

If no initial objects are detected ($N_{\text{obj}} = 0$), a default moderate complexity (0.3) is assigned. While a learning-based approach could potentially derive optimal weights, this heuristic method provides interpretability and avoids the need for complexity-labeled training data.

## 3.3  Adaptive Threshold Calculation

The calculated $C_{\text{scene}}$ score is mapped to an adaptive confidence threshold ($T_{\text{adaptive}}$) using a non-linear function designed empirically to provide gradual adjustments in mid-ranges and more significant changes at complexity extremes, relative to a base threshold ($T_{\text{base}} = 0.25$). The adjustment range is defined by $\Delta_{\text{min}} = -0.12$ and $\Delta_{\text{max}} = 0.05$. The core mapping aims to decrease the threshold as complexity increases. A simplified representation of the adjustment logic can be expressed as:

$$T_{\text{adjustment}} \approx \Delta_{\text{min}} \times C_{\text{scene}}^{\gamma} + (\text{offset related to } \Delta_{\text{max}})$$

where $\gamma$ controls the non-linearity (empirically determined, $> 1$ to emphasize high complexity). The actual implementation uses conditional logic based on $C_{\text{scene}}$ ranges to achieve the desired smooth transition between $T_{\text{base}} + \Delta_{\text{min}}$ and $T_{\text{base}} + \Delta_{\text{max}}$.

The final adaptive threshold is bounded between 0.08 and 0.95: $T_{\text{adaptive}} = \max(0.08, \min(0.95, T_{\text{base}} + T_{\text{adjustment}}))$. This specific mapping function was selected based on preliminary experiments.

## 3.4  Class-Specific Adjustments (Optional)

Before final filtering, the calculated $T_{\text{adaptive}}$ can be further modified based on the specific class of each object candidate, using predefined, manually curated adjustments. These adjustments reflect general knowledge about relative detection difficulty (e.g., 'cell phone': -0.03, 'car': +0.03) and were tuned based on COCO object characteristics. This results in a class-specific threshold $T_{\text{class\_adaptive}}$.

## 3.5  Context-Aware Reasoning (Optional)

If enabled, a simple, rule-based context module provides an additional confidence boost or penalty based on predefined object co-occurrence relationships. Objects found with expected context receive a boost (max +0.09), while certain objects found without their typical context receive a penalty. For example, a 'tie' detection might receive a confidence penalty if no 'person' is detected nearby with high confidence, while a 'backpack' detection might receive a small boost if detected near a 'person'. These rules are manually defined and serve as a basic mechanism for incorporating context without complex learned models. This contextual adjustment effectively modifies the required confidence score for an object to pass the threshold.

### 3.6 Final Filtering

Each object candidate from the initial low-confidence detection pass (Section 3.1) is finally kept only if its original confidence score meets or exceeds its calculated dynamic threshold ($T_{\text{class\_adaptive}}$ potentially influenced by context). Standard NMS with an IoU threshold of 0.45 is then applied to the filtered detections.

## 4 Experiments & Results

We evaluated AdaptiVision against a baseline using standard COCO metrics on the val2017 dataset.

### 4.1 Experimental Setup

- **Dataset:** COCO 2017 Validation set (val2017, 5000 images) with standard annotations (`instances_val2017.json`)[4].

- **Baseline Method:** Standard YOLOv8n model inference results processed with a fixed confidence threshold of 0.25 and standard NMS (IoU 0.45). Results evaluated using `pycocotools`. (Note: For mAP/AR table, baseline uses low threshold 0.001 evaluated across all scores).

- **AdaptiVision Method:** YOLOv8n initial low-confidence (0.05) inference pass, followed by AdaptiVision post-processing (complexity calculation, adaptive thresholding with context and class adjustments enabled) and standard NMS (IoU 0.45). Final detections evaluated using `pycocotools`.

- **Hardware & Software:** Experiments run on Apple MacBook Pro (M-series chip with MPS acceleration via PyTorch [15]). Key software: Python 3.12, PyTorch 2.3, Ultralytics 8.3 [3], OpenCV 4.11 [14], NumPy [13], `pycocotools`, Matplotlib [16], Seaborn [17].

- **Evaluation Metrics:** Standard COCO mAP metrics (AP@[.50:.95], AP@.50, AP@.75, AP small/medium/large); Average Recall (AR@[.50:.95] maxDets=1/10/100, AR small/medium/large); average object counts; and average processing time per image.

### 4.2 Implementation Details

The evaluation involved running the `scripts/run_experiments.py` script to process all 5000 images in the COCO val2017 set for both the standard (fixed 0.25 threshold) and adaptive methods. The script generated per-image results, comparison images, visualizations, and summary analytics including COCO JSON prediction files for formal evaluation. `pycocotools` was used to calculate the official mAP and AR metrics. The code is available at https://github.com/FutureAtoms/AdaptiVision/.

### 4.3 Quantitative Results (COCO val2017)

The standard COCO evaluation metrics comparing the AdaptiVision method against the baseline YOLOv8n (evaluated across all score thresholds for mAP/AR) on the val2017 dataset are presented in Table 1.

The results indicate that AdaptiVision, despite its adaptive mechanism, resulted in lower performance across all standard COCO metrics on the val2017 benchmark compared to the baseline YOLOv8n evaluated appropriately. Both Average Precision (AP) and Average Recall (AR) decreased, suggesting that the adaptive filtering, in its current form, removed more true positives relative to false positives than the standard approach evaluated across score thresholds. The largest drops were observed in Average Recall (e.g., AR maxDets=100 dropped from 0.464 to 0.428).

#### 4.3.1 Object Count and Processing Time Analysis

While accuracy decreased, AdaptiVision demonstrated effects on object counts and processing speed, summarized in Table 2 and Figure 2.

AdaptiVision detected significantly more objects on average (6.43 vs. 4.95) compared to using a fixed threshold of 0.25. This aligns with the goal of potentially increasing recall by lowering thresholds in complex scenes. Furthermore, the average processing time per image was substantially lower for AdaptiVision (0.0241s vs 0.0343s),

Table 1: COCO val2017 Object Detection Accuracy Results (mAP/AR)

| Metric | Baseline (YOLOv8n) | AdaptiVision |
|---|---|---|
| AP @[IoU=0.50:0.95] area=all | 0.355 | 0.340 |
| AP @[IoU=0.50] area=all | 0.496 | 0.470 |
| AP @[IoU=0.75] area=all | 0.387 | 0.372 |
| AP area=small | 0.162 | 0.150 |
| AP area=medium | 0.392 | 0.377 |
| AP area=large | 0.509 | 0.490 |
| AR @[IoU=0.50:0.95] area=all maxDets=1 | 0.290 | 0.277 |
| AR @[IoU=0.50:0.95] area=all maxDets=10 | 0.446 | 0.414 |
| AR @[IoU=0.50:0.95] area=all maxDets=100 | 0.464 | 0.428 |
| AR area=small maxDets=100 | 0.221 | 0.196 |
| AR area=medium maxDets=100 | 0.513 | 0.477 |
| AR area=large maxDets=100 | 0.647 | 0.600 |

Table 2: Average Object Counts and Processing Time (COCO val2017)

| Metric | Standard (Fixed 0.25) | AdaptiVision |
|---|---|---|
| Avg. Objects Detected | 4.95 | 6.43 |
| Avg. Processing Time (s) | 0.03 | 0.02 |
| Avg. Speed Improvement | - | 1.49x |

resulting in an average speed improvement factor of approximately 1.49x. This speedup likely occurs because the initial low-confidence pass (0.05) in AdaptiVision might be faster on average than the fixed 0.25 pass, and the subsequent complexity/filtering logic adds less overhead than the time saved during inference, especially on the MPS device.

The distribution plots in Figure 2 provide more detail. The object count difference plot shows AdaptiVision frequently detects more objects than the standard method (mean difference > 0). The processing time difference plot confirms AdaptiVision is generally faster (mean difference > 0). The scatter plots (Figure 3) visually confirm these trends across the dataset, showing most points above the y=x line for object counts and below the y=x line for processing time.

### 4.4 Qualitative Examples

Figure 4 presents selected qualitative comparisons from the COCO val2017 dataset, illustrating AdaptiVision's behavior in different scenarios. While AdaptiVision detects more objects on average, these examples show instances where it finds relevant objects missed by the standard threshold (e.g., Figure 4a) and cases where complexity leads to lowered thresholds and potentially more detections overall, some potentially false positives (e.g., Figure 4b). Figure 5 shows an example complexity visualization.

## 5   Discussion

The evaluation using standard COCO mAP metrics on the val2017 dataset revealed a critical outcome: the current AdaptiVision implementation degrades overall object detection accuracy compared to the baseline YOLOv8n (Table 1). This occurred despite AdaptiVision detecting significantly more objects on average and processing images faster than the standard fixed-threshold approach (Table 2, Figures 2, 3). This highlights a fundamental trade-off: the heuristics used successfully lowered thresholds often enough to increase object counts and potentially enable faster inference passes, but this came at the cost of reduced precision or insufficient recall gains at appropriate score levels, ultimately lowering standard mAP/AR scores.
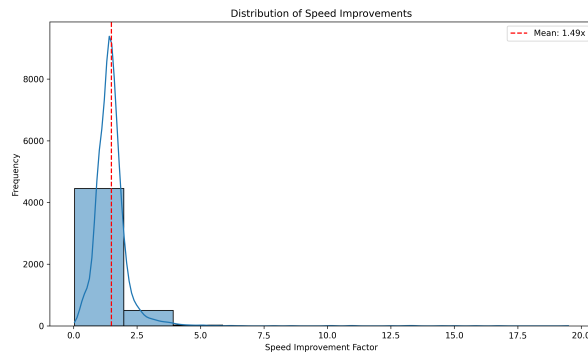
The discrepancy between increased object counts/speed and decreased mAP/AR underscores the limitations of simple metrics and the importance of standard evaluation protocols. The heuristic scene complexity calculation, while
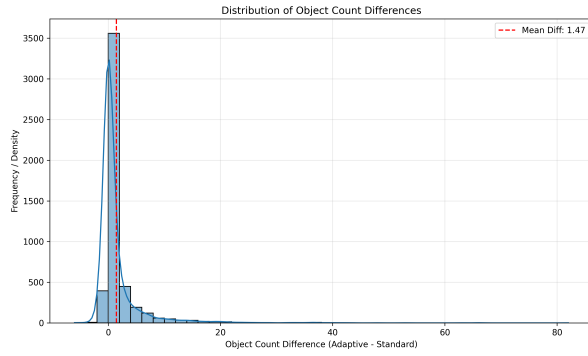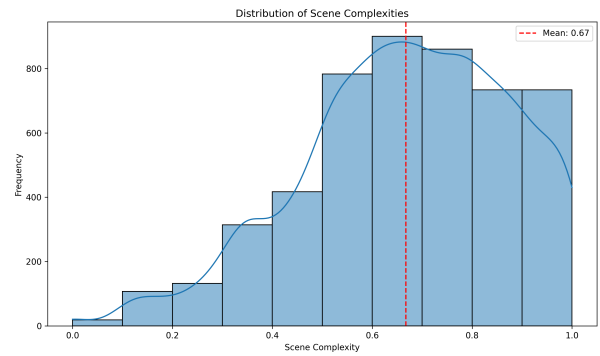
(a) Object Count Difference (Adaptive - Standard)
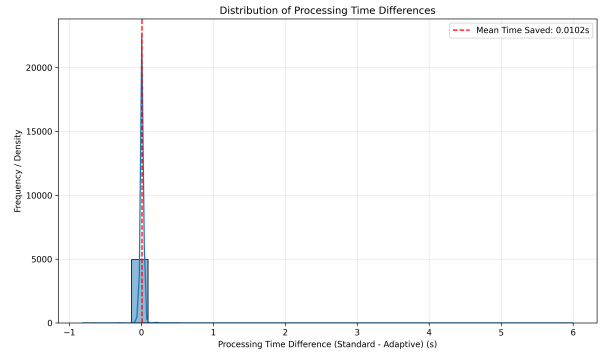


(b) Processing Time Difference (Standard - Adaptive)



(c) Speed Improvement Factor Distribution



(d) Scene Complexity Score Distribution

Figure 2: Distributions of key metrics across the COCO val2017 dataset. (a) Difference in object counts per image. (b) Difference in processing time per image (positive means adaptive is faster). (c) Speed improvement factor (Standard Time / Adaptive Time). (d) Distribution of calculated scene complexity scores.
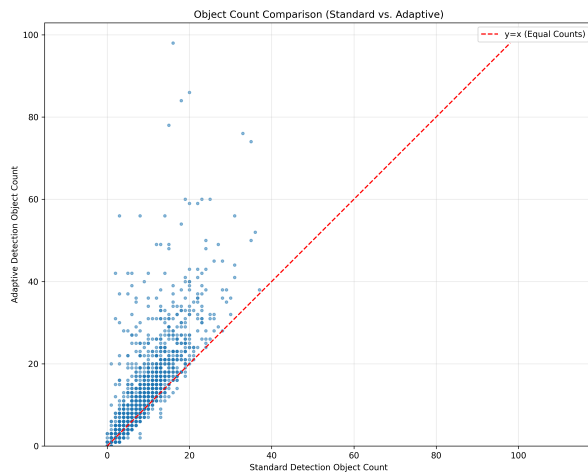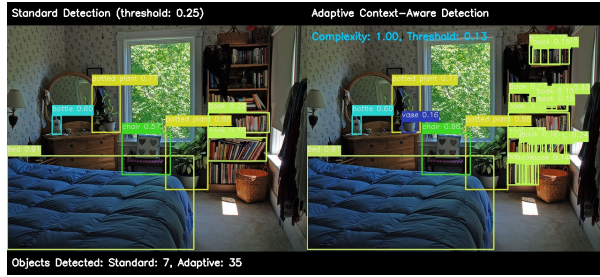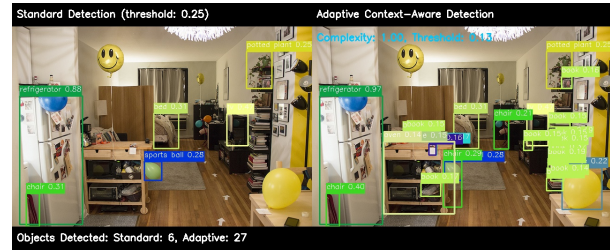


(a) Object Count Scatter Plot
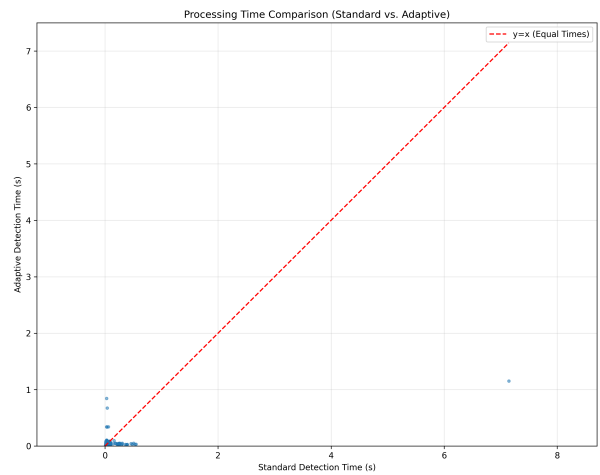


(b) Processing Time Scatter Plot

Figure 3: Scatter plots comparing per-image results on COCO val2017. (a) Number of objects detected (Standard vs. Adaptive). (b) Processing time in seconds (Standard vs. Adaptive). The red dashed line represents y=x.
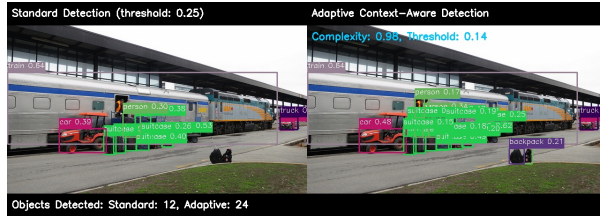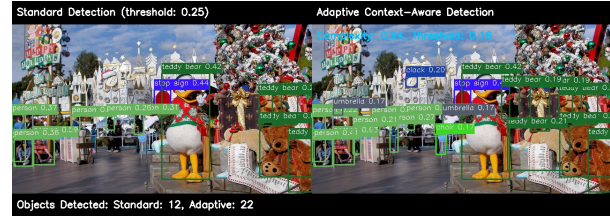
(a) Example 1 (Image ID: ...0632)

(b) Example 2 (Image ID: ...14038)

(c) Example 3 (Image ID: ...137727)

(d) Example 4 (Image ID: ...145020)

Figure 4: Qualitative comparison examples on COCO val2017 images. Left panel: Standard detection (0.25 threshold). Right panel: AdaptiVision detection. These examples illustrate how AdaptiVision's behavior varies with scene content.

intuitively designed, likely fails to capture the true factors determining the optimal threshold for maximizing mAP. The empirically tuned mapping function, derived from preliminary experiments, did not generalize effectively to the diverse COCO val2017 set. The adaptive threshold may be set suboptimally—too high in some cases (harming recall of true positives) or too low in others (introducing false positives not adequately compensated for, harming precision).

**Ablation Study Insights** Further analysis through an ablation study reinforces the conclusion that the core adaptive strategy, as implemented, is ineffective for improving mAP. Evaluating variants with only adaptive thresholding enabled (disabling context and class adjustments) or only context-aware reasoning enabled yielded identical mAP results (AP@[.50:.95] = 0.340) to the full AdaptiVision method, and significantly lower than the baseline (AP@[.50:.95] = 0.355). This indicates that neither the heuristic complexity-based thresholding nor the simple contextual rules, even in isolation, provided a benefit over the baseline evaluation, suggesting the fundamental approach requires reconsideration (see results in `results/ablations/adaptivision_adaptive_only_eval.txt` and `results/ablations/adaptivision_context_only_eval.txt`).
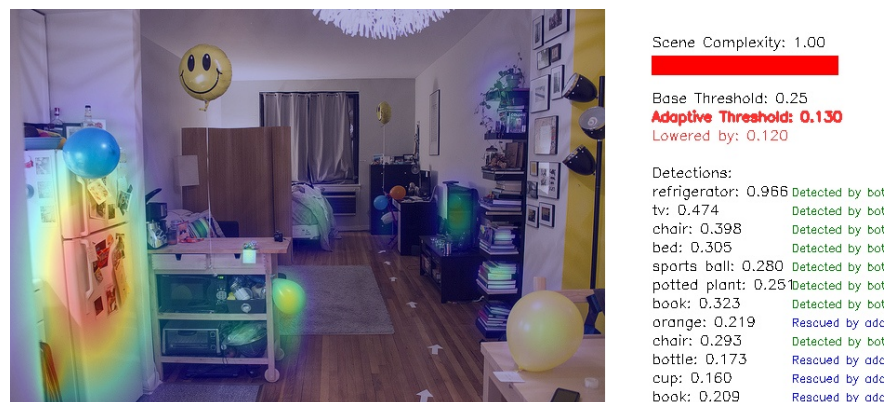


Figure 5: Example scene complexity visualization (Image ID: ...14038). Shows initial low-confidence detections used to compute the scene complexity score (0.94) and the resulting adaptive threshold (0.15) compared to the base (0.25).

Furthermore, an additional ablation confirmed that the geometric post-processing filter, designed to remove redundant overlapping boxes, had a negligible impact on performance; removing it yielded the same final mAP (AP@[.50:.95] = 0.340) as the full AdaptiVision method (`results/ablations/adaptivision_no_postfilter_eval.txt`). This rules out the filter as a source of the observed accuracy degradation.

Critically, this performance degradation was not confined to a specific type of scene; the evaluation stratified by the method's own complexity score showed lower mAP for AdaptiVision compared to the baseline in low, medium, and high complexity image subsets (`results/ablations/baseline_eval_with_complexity.txt`, `results/ablations/adaptivision_full_eval_with_complexity.txt`).

The observed speed improvement (avg. 1.49x) is notable and somewhat counter-intuitive, as adaptive logic adds overhead. This suggests the dominant factor is the base model's inference time, which might be faster on average when using a very low initial threshold (0.05 for AdaptiVision) compared to a higher fixed threshold (0.25 for standard count/time comparison), particularly with MPS acceleration where computation patterns can affect performance. While faster processing is desirable, it cannot justify the drop in accuracy metrics for most applications.

Limitations previously considered, such as reliance on non-learned heuristics and simplistic context rules, are likely central to the performance degradation. Future work must fundamentally revisit the adaptive mechanism. Data-driven approaches, potentially learning complexity metrics and threshold mappings directly from data optimized for mAP, appear necessary. Integrating such logic directly into the model architecture or using reinforcement learning could be viable paths. However, any adaptive method must be rigorously validated against standard benchmarks like COCO mAP/AR.

# 6    Conclusion

This paper evaluated AdaptiVision, a post-processing technique using heuristic scene complexity analysis to dynamically adjust confidence thresholds for YOLOv8n. Evaluation on the COCO 2017 validation set showed that while AdaptiVision increased the average number of detected objects and significantly improved processing speed compared to a fixed-threshold approach, it resulted in lower overall accuracy according to standard COCO mAP and AR metrics.

This outcome highlights the difficulty in designing heuristic-based adaptive methods that improve detection accuracy robustly across diverse datasets. The chosen heuristics and threshold mapping strategy, while conceptually simple, failed to effectively balance precision and recall for optimal mAP performance. The study emphasizes that improvements in intermediate metrics like object count or speed do not necessarily translate to better performance on standard evaluation benchmarks. Future efforts towards adaptive thresholding likely require learning-based approaches directly optimized for standard accuracy metrics.

## Broader Impact Statement

The goal of AdaptiVision was to improve object detection robustness. The finding that this implementation decreased standard performance metrics mitigates immediate concerns about misuse (e.g., enhanced surveillance) compared to if it had significantly improved capabilities. However, the general principle applies: any successful improvement in detection could be misused. Biases present in the base detector or datasets used for tuning remain a concern for any detection system and require careful analysis before deployment. Future development of adaptive methods should prioritize fairness evaluations alongside performance metrics.

## References

[1] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

[2] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*.

[3] Jocher, G., Chaurasia, A., Stoken, A., et al. (2023). Ultralytics YOLOv8. GitHub repository: https://github.com/ultralytics/ultralytics

[4] Lin, T. Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer.

[5] Wu, Y., Chen, Y., Yuan, L., et al. (2020). Rethinking classification and localization for object detection. In *CVPR* (pp. 10260-10269).

[6] Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-NMS—improving object detection with one line of code. In *ICCV* (pp. 5561-5569).

[7] Hu, H., Gu, J., Zhang, Z., et al. (2018). Relation networks for object detection. In *CVPR* (pp. 3588-3597).

[8] Lee, E., Jung, M., & Kim, A. (2024). Toward Robust LiDAR based 3D Object Detection via Density-Aware Adaptive Thresholding. *arXiv preprint arXiv:2404.13852*.

[9] Ma, L. V., Hussain, M. I., Park, J., et al. (2023). Adaptive Confidence Threshold for ByteTrack in Multi-Object Tracking. In *ICCAIS* (pp. 1-6). IEEE. (arXiv:2312.01650)

[10] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *ICML* (pp. 1321-1330). PMLR.

[11] Li, C., Li, L., Jiang, H., et al. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

[12] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *CVPR* (pp. 7464-7475).

[13] Harris, C. R., Millman, K. J., Van Der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362.

[14] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[15] Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *NeurIPS*, *32*.

[16] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90-95.

[17] Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.