Enhancing Object Detection in YOLOS-Small through Advanced LoRA Methods

Orin Cohen
orincoh@post.bgu.ac.il

Tom Damari
damarit@post.bgu.ac.il

Abstract

This project explores fine-tuning strategies for YOLOS-Small (YOLOS-S), a Vision Transformer-based object detection model, using Low-Rank Adaptation (LoRA) and its advanced variants. While YOLOS-S performs well on COCO, its fine-tuning is computationally expensive. LoRA-based methods address this by introducing low-rank trainable matrices, significantly reducing computational costs. We compare multiple LoRA variants including standard LoRA, AdaLoRA, LoHa, and LoKr, evaluating their impact on detection performance.

Experiments on a COCO subset show slight improvements over the baseline, with AdaLoRA achieving the highest mAP@[0.5:0.95], though small-object detection remains a challenge. Future work should refine LoRA configurations, explore adaptive loss weighting, and integrate LoRA into additional Transformer layers and larger datasets to enhance generalization. The full implementation is available on GitHub: GitHub Repository.

1. Introduction

Object detection has been a cornerstone in computer vision, evolving from convolutional neural networks (CNNs) to more advanced architectures like Vision Transformers (ViTs). Traditional CNN-based methods, while effective, often rely on intricate designs and strong 2D inductive biases, making them computationally expensive and less flexible. Recently, the paper "You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection" introduced YOLOS, a series of Vision Transformer-based models, including YOLOS-Small (YOLOS-S). YOLOS-S explores whether Transformers, originally designed for natural language processing (NLP) and image classification, can effectively handle object detection tasks with minimal modifications while maintaining computational efficiency (Fang et al., 2021).

2. YOLOS

YOLOS (You Only Look at One Sequence) is a Transformer-based object detection model that eliminates the need for traditional region proposal mechanisms and anchor-based bounding boxes. Unlike CNN-based detectors that extract hierarchical features through convolutional layers, YOLOS operates on flattened image patches, processing them as a sequence of tokens. Inspired by DETR (Carion et al., 2020), YOLOS utilizes self-attention mechanisms to model spatial relationships and detect objects end-to-end. In the original study, the model was pre-trained on the ImageNet-1K dataset, leveraging its extensive image classification capabilities, and subsequently fine-tuned on the COCO 2017 dataset to optimize performance for object detection tasks (Fang et al., 2021).

2.1 YOLOS Architecture

YOLOS is built on the Vision Transformer (ViT) framework, adapting it for object detection by treating an image as a sequence of tokens rather than processing it through convolutional layers. The input image is first divided into non-overlapping patches, each of which is linearly projected into an embedding vector that preserves spatial information. Since Transformers do not inherently encode positional relationships, positional embeddings are added to the patch tokens to maintain spatial coherence. In addition to these patch embeddings, YOLOS introduces a set of learnable detection tokens ([DET] tokens), which function as object queries. These tokens interact with the patch embeddings through self-attention layers within a multi-layer Transformer encoder, gradually refining their representations to predict object locations and categories. Finally, each detection token is passed through a Multi-Layer Perceptron (MLP) head, which directly outputs bounding boxes and class labels. Unlike traditional object detection models that rely on region proposal networks (RPNs) or anchor boxes, YOLOS predicts objects in a single forward pass, eliminating the need for predefined bounding box anchors or multi-stage processing (Fang et al., 2021). This architecture is illustrated in Figure 1.

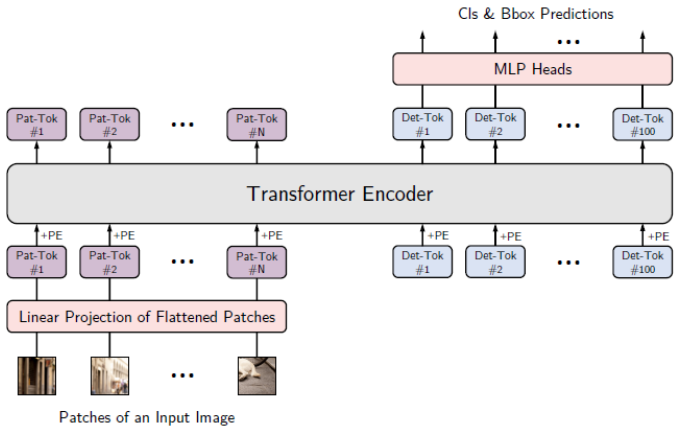


Figure 1: YOLOS architecture overview (Fang et al., 2021).

2.2 YOLOS Variants

The YOLOS family consists of multiple model variants, each designed to balance accuracy, computational cost, and efficiency. The smallest variant, YOLOS-Ti (Tiny), is optimized for lightweight applications, reducing the number of transformer layers and embedding dimensions to achieve faster inference at the cost of lower detection performance. YOLOS-S (Small) provides a trade-off between speed and accuracy, maintaining a reasonable model size while offering competitive results on object detection benchmarks. Finally, YOLOS-B (Base) is the largest and most powerful model, incorporating more transformer layers and a higher hidden dimension, leading to improved object recognition capabilities but at a higher computational cost (Fang et al., 2021). Each of these models follows the same fundamental architecture, differing mainly in depth, width, and parameter count, allowing flexibility for different application needs.

2.3 YOLOS-S

YOLOS-S was selected for this study due to its optimal balance between performance and computational efficiency. It consists of 12 transformer encoder layers, a hidden dimension of 384, and fewer overall parameters compared to YOLOS-Base, making it more suitable for real-world applications with limited resources. Despite its smaller size, YOLOS-S achieves an mAP of 36.1 on the COCO validation dataset, showcasing its ability to maintain competitive performance while being computationally efficient (Fang et al., 2021). This combination of accuracy and efficiency makes YOLOS-S an ideal choice for exploring further improvements and fine-tuning techniques tailored to specific tasks, such as those implemented in this study.

3. LoRA

3.1 LoRA Fundamentals

Large-scale pre-trained vision models (PVMs) demonstrate strong adaptability across vision tasks, yet their growing parameter scale makes full fine-tuning computationally impractical. Parameter-Efficient Fine-Tuning (PEFT) addresses this by modifying only a small subset of parameters, significantly reducing computational costs while maintaining performance (Xin et al., 2024; Han et al., 2024). Given these advantages, PEFT methods, particularly LoRA and its variants, have been applied to YOLOS-S, leveraging its ViT architecture to enhance fine-tuning efficiency for object detection.

Among PEFT techniques, Low-Rank Adaptation (LoRA) enables fine-tuning by introducing low-rank trainable matrices while keeping most model weights frozen, preserving efficiency without sacrificing accuracy (Hu et al., 2022; Lue, 2024; Wu et al., 2024). Instead of modifying all model parameters, **LoRA** decomposes weight updates into two smaller trainable matrices $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$, where $r \ll \min(m, n)$, significantly reducing trainable parameters. The pre-trained weight matrix W remains unchanged, and only A and B are optimized during fine-tuning. At inference, the LoRA updated weight $\Delta W = BA$ is merged back into the original model, allowing efficient adaptation to new tasks without additional computational overhead. The scaling factor regulates LoRA's contribution, balancing task-specific adaptation with pre-trained knowledge.

A visual representation can be seen in Figure 2.

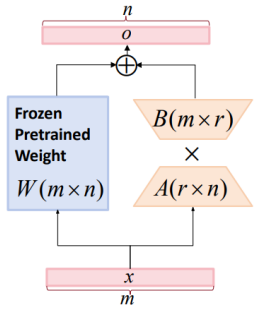


Figure 2: LoRA weight adaptation process (Lu, 2024).

3.2 LoRA Variants

The fixed-rank structure of LoRA limits adaptability in complex tasks, leading to the development of Low-Rank Hadamard Product (LoHa), Low-Rank Kronecker Product (LoKr), and Adaptive Low-Rank Adaptation (AdaLoRA), which enhance flexibility, structural preservation, and dynamic adaptation (Hu et al., 2022; Lue, 2024; Wu et al., 2024).

LoHa replaces standard matrix multiplication with the Hadamard product, an element-wise multiplication that enhances expressiveness without significantly increasing the number of parameters. Instead of two matrices, LoHa uses four smaller ones, improving feature representation- particularly useful for diffusion models and generative AI (Lue, 2024; Wu et al., 2024).

LoKr preserves weight matrix structure while reducing computational overhead by leveraging the Kronecker product (Hugging Face, 2023; Lue et al., 2024). This technique allows for efficient vectorization of trainable parameters, making it ideal for multimodal and vision-language models (Edalati et al., 2022; Lue, 2024; Wu et al., 2024).

AdaLoRA dynamically allocates trainable parameters, assigning more resources to critical layers while pruning less essential ones. Using Singular Value Decomposition (SVD)-based techniques, AdaLoRA optimizes computational efficiency, improving fine-tuning for large-scale AI models (Zhang et al., 2023; Lue, 2024; Wu et al., 2024).

4. Proposal

This project aims to examine the impact of integrating LoRA and its variants into YOLOS-S for object detection. The objective is to compare the performance of the standard YOLOS-S model with different adaptations and as-

sess whether these modifications enhance detection capability while maintaining computational efficiency. Specifically, we explore LoRA, AdaLoRA, LoHa, and LoKr to evaluate their influence on the model's ability to effectively represent and detect objects. By analyzing these adaptations, we aim to gain insights into their suitability for fine-tuning transformer-based object detection models.

5. Methodology

5.1 Dataset and Pre-processing

For this project, we used the COCO 2017 dataset, a widely recognized benchmark for object detection tasks that includes 80 categories. To prevent data leakage, we chose to use only images from the COCO Validation set for training and evaluation, as YOLOS-S was originally fine-tuned on the COCO Train dataset. Due to resource constraints, we did not utilize the entire COCO 2017 validation set but instead applied shuffling and then selected a subset, while ensuring that all 80 categories were adequately represented in the different splits of the train, validation and test sets.

The subset of the COCO 2017 validation set, which contains 3,000 images, was divided according to 70% - 30% split ratio between training (2,100 images) and testing (900 images). Within the training set, we further split the data into 70% for training (1,470 images) and 30% for validation (630 images). After performing these splits, we have applied another shuffle.

Before using the dataset in our YOLOS-S model, several preprocessing steps were required to ensure compatibility. These preprocessing steps were handled both manually and by the YOLOS Image Processor.

First, we applied manual preprocessing steps to ensure data consistency before passing it through the processor. Since some images in COCO are not stored in RGB format, we manually ensured that all images were **converted to RGB** before further processing, aligning with the model's requirements. Additionally, due to differences in label indexing between COCO and YOLOS-S, we applied **Category Mapping**. While COCO defines 80 object categories, YOLOS-S follows a 91-category scheme. These 11 additional categories in YOLOS-S, which do not exist in COCO, were marked as N/A in the original model. To ensure compatibility, we implemented a mapping function that aligned COCO's category indices with the expected label structure. Any annotations without a corresponding mapping were filtered out, ensuring consistency in the dataset used for training and evaluation.

After these manual adjustments, the dataset was processed using the **YOLOS Image Processor**, which performed the following automated transformations:

- **Image Resizing**: The processor resized each image to a maximum size while maintaining the original aspect ratio, ensuring that the shortest edge does not exceed 800 pixels and the longest edge does not exceed 1333 pixels.
- **Padding**: If needed, zero-padding was applied to standardize image sizes across a batch while preserving the aspect ratio. This ensured uniform input dimensions without distorting the image.
- **Pixel Value Rescaling**: Pixel values were rescaled from their original range of 0-255 to 0-1, a standard preprocessing step for deep learning models.
- **Normalization**: The processor applied normalization based on ImageNet statistics, where pixel values were standardized using:
  - **Mean**: [0.485, 0.456, 0.406]
  - **Standard deviation**: [0.229, 0.224, 0.225]
- **Annotation Conversion**: The YOLOS Image Processor converted COCO's bounding box format $(x_{\text{min}}, y_{\text{min}}, \text{width}, \text{height})$ into YOLOS's expected format $(x_{\text{center}}, y_{\text{center}}, \text{width}, \text{height})$, aligning with how YOLOS processes object locations. Additionally, the processor ensured that bounding box coordinates were properly scaled and normalized to fit the resulted images.

These preprocessing steps ensured that the input data was consistent with the model's expected format, improving both training stability and inference performance. The entire preprocessing pipeline was implemented within a custom collation function, which structured the dataset into properly formatted batches, containing both the processed **pixel values** of the images and their **labels** which contain the annotations for seamless integration into the YOLOS-S model.

5.2 Experimental Setup

Initially, we aimed to reproduce the performance of the YOLOS-S model in its original form to establish a baseline for comparison. Unlike the original paper, which evaluated the model on the entire COCO 2017 validation set, we used a significantly smaller test set. As a result, the metric scores we obtained were not directly comparable to those reported in the paper. However, this step was crucial for understanding the model's performance under our specific experimental conditions and providing a reference point for assessing improvements. After establishing the baseline, we proceeded to integrate LoRA variants into the YOLOS-S model to explore its impact on performance. Since LoRA variants does not directly support object detection models, the target modules had to be manually specified. We applied LoRA variants to the query and value projection matrices within the multi-head self-attention (MHSA) layers to enable efficient fine-tuning while keeping most of the model parameters frozen. These layers were chosen because the query influences attention distribution, while the value determines feature representations, both of which are crucial for object detection. In contrast, the key matrix remained unchanged as its primary role is in token comparison rather than representation learning. Table 1 presents the total and trainable parameters for each of the evaluated models.

To systematically evaluate the effectiveness of LoRA variants, we conducted a hyperparameter tuning process, testing multiple configurations for each fine-tuned model. We experimented with different values for parameters such as

LoRA rank (r), scaling factor (α), dropout rates, and additional structural modifications for each LoRA variant. Throughout this process, we compared various configurations in terms of their impact on model performance and generalization ability. The final hyperparameter configurations used in our experiments are summarized in Table 2. To assess the impact of these configurations, we evaluated their performance using the mean Average Precision (mAP) metric on the validation set during training and test set. Further details on the evaluation process are provided in Section 5.3.

Model	Trainable	Total	Trainable (%)
YOLOS-S with LoRA	294,912	30,979,680	0.9520
YOLOS-S with AdaLoRA	295,296	30,980,088	0.9532
YOLOS-S with LoHA	442,368	31,127,136	1.4212
YOLOS-S with LoKr	74,112	30,758,880	0.2409

Table 1: Total and Trainable Parameters for each Model.

LoRA Variant	Rank (r)	Scaling (α)	Dropout	Special Features
LoRA	16	8	0.5	Standard
AdaLoRA	$24 \rightarrow 16$	Adaptive	None	Dynamic rank allocation
LoHa	8	12	0.1	Rank + module dropout
LoKr	8	16	0.3	Kronecker decomposition

Table 2: Hyperparameter configurations of LoRA variants.

In addition to the common hyperparameters, each LoRA variant introduces unique configurations:

- **AdaLoRA**: Dynamically adjusts rank allocation during training, starting at $r = 24$ and adapting to $r = 16$. Uses scheduling parameters: $t_{init} = 20\%$, $t_{final} = 40\%$, $\beta_1 = 0.9$, and $\beta_2 = 0.88$. To prevent rank collapse and maintain diversity in adapted weights, an orthogonal regularization term $\lambda_{\text{orth}} = 0.01$ is applied.
- **LoHa**: Applies *module dropout* (0.1), selectively deactivating adaptation in certain modules to enhance regularization and improve robustness.
- **LoKr**: Uses Kronecker decomposition with *module dropout* (0.1) for enhanced sparsity. Additionally, it enables Kronecker-based decomposition in Conv2D layers (*effective Conv2D support*) and applies compression to both Kronecker factors (*decomposing both*- left and right matrices) with *decompose factor*=8.

In YOLOS-S, the built-in loss function is Bipartite Loss, which comprises Cross-Entropy (CE) Loss for classification, Generalized Intersection over Union (GIoU) Loss for geometric alignment, and Bounding Box (BBox) Regression Loss for precise localization. However, during training, reductions in Bipartite Loss did not always correspond to improvements in mAP, revealing a misalignment between the optimization objective and the evaluation metric. This loss function optimizes per-instance predictions rather than overall detection quality, whereas mAP evaluates the model's ability to balance precision and recall across varying confidence thresholds. To address this issue, we experimented with different loss weight configurations and found that setting GIoU Loss to 2, replacing CE Loss with Focal Loss (weighted at 1.5), and keeping BBox Regression Loss at 1 yielded the best performance. These changes reinforced spatial alignment, improved localization accuracy, and helped mitigate class imbalance in our dataset, where certain categories (e.g., "person") dominate while others (e.g., "toaster," "hair drier") are significantly underrepresented. Unlike CE Loss, which treats all samples equally, Focal Loss dynamically down-weights easy examples and assigns greater importance to more challenging samples with higher classification uncertainty. Since Focal Loss is not natively included in the model's loss decomposition, we computed it manually, which required us to use a batch size of 4 instead of 8 due to resource constraints. These adjustments improved the alignment between the optimization process and mAP, leading to better generalization.

For optimization, we used AdamW, a widely adopted optimizer for Transformer-based models, particularly in LoRA fine-tuning. AdamW helps mitigate overfitting by decoupling weight decay from the optimization process, making it well-suited for parameter-efficient tuning where only a subset of parameters is updated. We experimented with different learning rates to balance stability and convergence. LoRA performed best with a $5e^{-5}$ learning rate, ensuring smooth optimization without instability. AdaLoRA, LoHa, and LoKr required a more conservative $3e^{-5}$ learning rate due to their unique rank adaptation and structural modifications. Additionally, for LoKr, we adjusted the optimizer's hyperparameters, setting $\beta_2 = 0.98$ and eps $= 1e^{-8}$ to improve gradient estimation, while reducing weight decay to $5e^{-3}$ to allow more flexibility in parameter updates. Due to memory constraints, we used gradient accumulation with *accumulation step=4*, effectively increasing the batch size from 4 to 16. This allowed stable training without exceeding hardware limitations, ensuring learning dynamics remained consistent with larger batch settings.

To maintain stable training, we tested different learning rate schedulers. Initially, we applied ReduceLROnPlateau, but it led to abrupt learning rate reductions, destabilizing training. Instead, we adopted CosineAnnealingLR for LoRA and AdaLoRA, with $T_{\text{max}} = 5$ for LoRA and $T_{\text{max}} = 10$ for AdaLoRA, allowing gradual decay and preventing performance drops. LoHa benefited from CosineAnnealingWarmRestarts with $T_0 = 2$ and $T_{\text{mult}} = 2$, enabling periodic resets that helped avoid stagnation. For LoKr, we found that ReduceLROnPlateau was the most effective, as its adaptive nature allowed precise learning rate adjustments based on validation performance. We set the reduction factor to 0.5 and patience to 2 epochs, ensuring learning rate decreases were controlled and only triggered when necessary.

To prevent overfitting and optimize training efficiency, we implemented Early Stopping with a patience value of 3 epochs, terminating training when no improvement in mAP was detected. Specifically, early stopping was applied when no improvement in validation mAP@$[0.5 : 0.95]$ exceeding min_delta $= 0.001$ was observed for three consecutive epochs. Although the initial training duration was set to 10 epochs, we observed that training converged earlier except for LoHa, making early stopping an effective safeguard. In-

stead of selecting the final model checkpoint, we identified the model from the epoch with the highest mAP score, ensuring the best-performing version was retained. This strategy helped prevent unnecessary parameter updates in later epochs, improving generalization while maintaining computational efficiency.

As part of post-processing, we evaluated different prediction refinement techniques to determine their effect on model performance. Since object detection models often generate multiple overlapping bounding boxes for the same object, we experimented with Non-Maximum Suppression (NMS) and Weighted Box Fusion (WBF) to refine the predictions. NMS filters out overlapping bounding boxes by selecting the one with the highest confidence score, while WBF averages the coordinates of overlapping boxes to produce a single, more accurate prediction. Our motivation for testing these methods stemmed from the observation that YOLOS-S sometimes produced duplicate boxes for a single object. However, neither NMS nor WBF led to a meaningful improvement in mAP, and in some cases, they even reduced performance by eliminating valid detections. This indicated that YOLOS-S's built-in decoding process already handled duplicate boxes effectively, making additional post-processing unnecessary.

Throughout training, we closely monitored the relationship between training loss, validation loss, and mAP. While these loss components generally decreased, the validation mAP often plateaued or even declined, reinforcing our decision to adopt early stopping and rely on validation mAP as the primary criterion for selecting the best-performing model. These refinements, including loss function tuning, optimizer selection, learning rate scheduling, early stopping, and post-processing evaluations, allowed us to develop a more stable and effective fine-tuning strategy for YOLOS-S when combined with LoRA-based adaptation.

5.3 Evaluation

In object detection tasks, model predictions are not binary (correct/incorrect) but rather evaluated based on the degree of overlap between the predicted bounding box and the ground truth box. To assess this overlap, the **Intersection over Union (IoU)** metric is used. IoU is defined as the ratio between the intersection area (where the two boxes overlap) and the union area (the total area covered by both boxes). IoU values range from 0 to 1, where 0 indicates no overlap and 1 represents a perfect match between the predicted and actual object location. Typically, a value of 0.5 is considered the minimum threshold for a prediction to be deemed valid.

To compare the models in our experiment, we chose to use the **mAP** metric, a widely adopted measure for evaluating object detection models. mAP combines precision and recall to assess how well a model detects objects and assigns correct categories. It is calculated by averaging the Average Precision (AP) across all observed categories, where AP is the area under the precision-recall curve.

Specifically, we utilize **mAP@[0.5:0.95]**, the standard benchmark used in datasets such as COCO. This metric is computed as the averaging AP scores over 10 different IoU thresholds, ranging from 0.5 (the minimum required overlap for a valid detection) to 0.95 (indicating highly precise localization), in steps of 0.05.

This approach provides a comprehensive and rigorous evaluation of model performance, as it ensures that the model is not only capable of classifying objects correctly but also of localizing them with high precision. Using mAP@[0.5:0.95] ensures that models are evaluated across a wide range of IoU thresholds, leading to a more reliable performance assessment.

6. Results

Since YOLOS-S is pretrained on a large-scale dataset, it exhibits stable convergence and generalization performance, reflected in the training behavior of the LoRA-based fine-tuning methods. While all fine-tuned models outperformed the baseline, improvements were marginal, likely due to computational constraints and limited hyperparameter tuning. Figures 3a–6a in the Appendices show that validation loss remains lower than training loss, as pretrained models generalize well, while fine-tuning increases training loss. Figures 3b–6b illustrate a sharp decrease in Focal Loss, improving object category distinction, while BBox and GIoU losses remain stable, reflecting YOLOS-S's strong pretrained localization. LoRA shows the steepest decline, whereas LoHa and AdaLoRA decrease more gradually, suggesting a more balanced fine-tuning process.

Figure 3 highlights key differences in training loss progression. The LoRA model converged rapidly but plateaued early, leading to early stopping by epoch 4. In contrast, LoKr, LoHa, and AdaLoRA showed a more gradual loss reduction, with LoHa improving steadily over 10 epochs. AdaLoRA exhibited the smoothest and most sustained optimization, refining performance until epoch 9. This aligns with mAP performance trends, where AdaLoRA and LoHa showed the most stable improvements, while LoRA and LoKr plateaued earlier. These findings suggest that adaptive fine-tuning techniques, particularly AdaLoRA's rank adjustments, enhance learning efficiency and generalization.

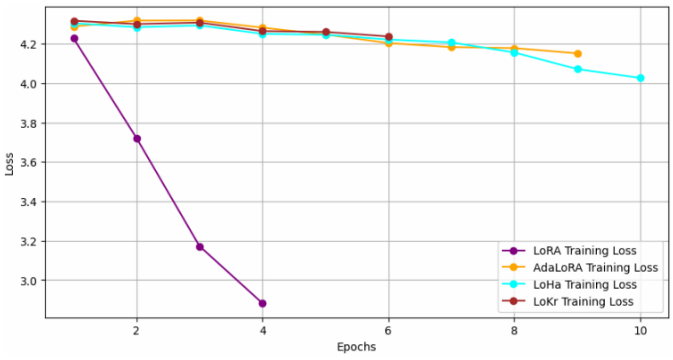


Figure 3: Training Loss Across Epochs for All Models.

The test set results indicate that all fine-tuned models surpassed the baseline YOLOS-S, though with marginal improvements. Table 3 compares mAP@[0.5:0.95] and detection performance at different confidence thresholds (mAP@50 and mAP@75). While all fine-tuned models slightly outperformed the baseline in mAP@[0.5:0.95], AdaLoRA achieved the highest score (33.3447), demonstrating its effectiveness in optimizing object detection. LoHa recorded the highest mAP@50 (51.6466), suggesting stronger performance at lower confidence thresholds, while LoKr achieved the best mAP@75 (34.5513), indicating improved precision under stricter confidence levels. These results highlight the role of LoRA variants in improving detection confidence and accuracy.

Model	mAP@50	mAP@75	mAP@[0.5:0.95]
Baseline YOLOS-S	51.2714	34.2762	33.2379
YOLOS-S with LoRA	51.6249	34.3871	33.3389
YOLOS-S with AdaLoRA	51.5363	34.4044	**33.3447**
YOLOS-S with LoHa	**51.6466**	34.2496	33.2727
YOLOS-S with LoKR	51.4323	**34.5513**	33.3272

Table 3: Comparison of mAP metrics at different confidence thresholds on the Test Set. The highest scores are highlighted in **bold**.

Further analysis in Table 4 evaluates detection performance across object sizes. LoKr was the only model to improve small-object detection (9.8507), slightly surpassing the baseline (9.7738), while LoRA and LoHa underperformed, reinforcing the challenge of detecting small objects. Improvements for medium-sized objects were minimal, with LoRA (31.9208) and AdaLoRA (31.9472) showing slight gains over the baseline (31.8657). LoHa (51.2017) and AdaLoRA (51.1807) achieved the highest scores for large-object detection, enhancing their ability to capture and classify larger objects. These findings suggest that LoRA variants are particularly effective for large-object detection but require further refinements for small-object localization.

Model	mAP_small	mAP_medium	mAP_large
Baseline YOLOS-S	9.7738	31.8657	50.9172
YOLOS-S with LoRA	9.2381	31.9208	51.0270
YOLOS-S with AdaLoRA	9.6612	**31.9472**	51.1807
YOLOS-S with LoHa	9.2201	31.8329	**51.2017**
YOLOS-S with LoKR	**9.8507**	31.8417	51.1198

Table 4: Comparison of mAP performance across different Object Size Categories on the Test Set. The highest mAP for each category is highlighted in **bold**.

As shown in Figure 4, AdaLoRA's overall mAP on the test set highlights its strong generalization across diverse object categories. A random sampling of 15 categories reveals high detection accuracy for large objects like airplanes and horses, aligning with Table 4, where AdaLoRA excelled in large-object detection. However, lower performance for categories like birds reflects ongoing challenges in small-object localization. Notably, for a category that had minimal representation across all dataset splits, and only 2 images in the test set, the model failed to detect it correctly, resulting in an mAP score of 0 for that category. Despite some difficulties in small-object detection, AdaLoRA emerged as the best-performing model, reinforcing the effectiveness of adaptive LoRA variants in object detection.

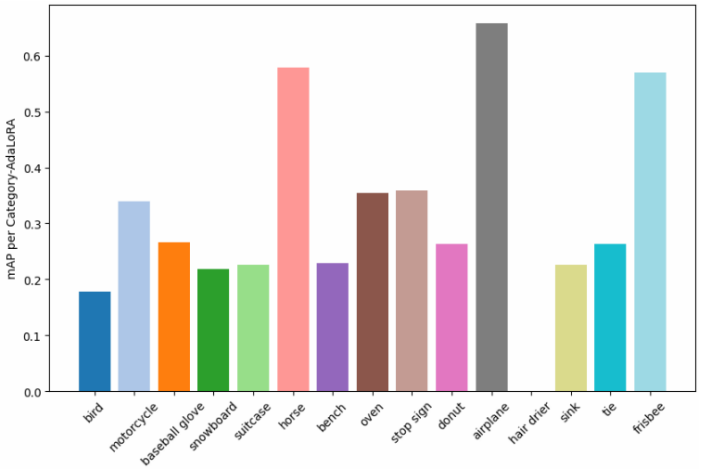


Figure 4: mAP@[0.5:0.95] of 15 Categories on the Test Set for the Best-Performing Model, AdaLoRA.

7. Conclusions and Future Work

The results of this study demonstrate that while LoRA-based fine-tuning provides slight improvements over the baseline YOLOS-S model, the overall gains in mAP were marginal. AdaLoRA achieved the highest mAP@[0.5:0.95], indicating that its adaptive rank allocation enhances detection performance. However, LoKr was the only model to improve small-object detection, while other variants primarily benefited large-object localization. The rapid convergence of LoRA and LoKr suggests potential limitations in long-term fine-tuning, whereas AdaLoRA and LoHa exhibited more sustained learning. These findings highlight that while LoRA variants can optimize computational efficiency, further refinements are needed to maximize their impact on object detection, particularly for small and complex objects.

Future research should focus on optimizing LoRA variant configurations to improve adaptation for object detection. Adaptive loss weighting could dynamically balance classification and localization losses, adjusting based on training progress to enhance detection accuracy. Exploring optimizers like Lion could improve convergence and generalization, especially with additional GPU resources. Expanding LoRA to more Transformer layers, such as feed-forward networks (FFN), may enhance feature extraction. Increasing dataset size (e.g., OpenImages, LVIS) could further improve model robustness and small-object detection accuracy. While our research suggests that LoRA-based fine-tuning is a promising approach for adapting YOLOS-S for object detection, fully unlocking its potential requires further optimizations, such as configuration tuning, loss balancing, and model integration, to enhance adaptation and performance.

8. References

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, August). End-to-end object detection with transformers. In *European Conference on Computer Vision* (pp. 213-229). Cham: Springer International Publishing.

Edalati, A., Tahaei, M., Kobyzev, I., Nia, V. P., Clark, J. J., & Rezagholizadeh, M. (2022). Krona: Parameter efficient tuning with Kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., ... & Liu, W. (2021). You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34, 26183-26197.

Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 1(2), 3.

Lu, J. (2024). Low-Rank Approximation, Adaptation, and Other Tales. *arXiv preprint arXiv:2408.05883*.

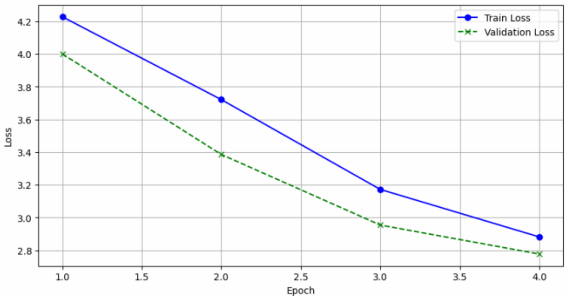
Solovyev, R., Wang, W., & Gabruseva, T. (2021). Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107, 104117.

Wu, T., Wang, J., Zhao, Z., & Wong, N. (2024). Mixture-of-subspaces in low-rank adaptation. *arXiv preprint arXiv:2406.11909*.

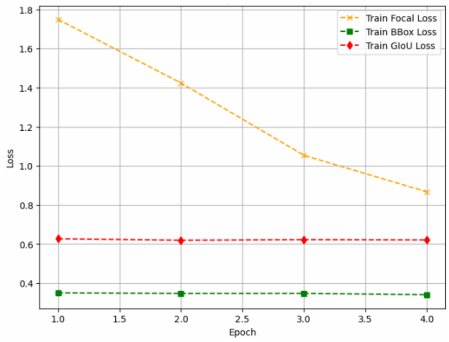
Xin, Y., Luo, S., Zhou, H., Du, J., Liu, X., Fan, Y., ... & Du, Y. (2024). Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv preprint arXiv:2402.02242*.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., ... & Zhao, T. (2023). AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

9. Appendices

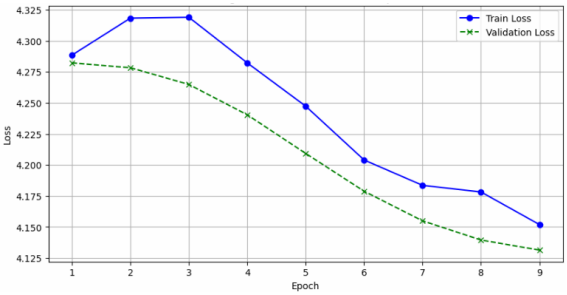


(a) Train and Validation Losses over Epochs

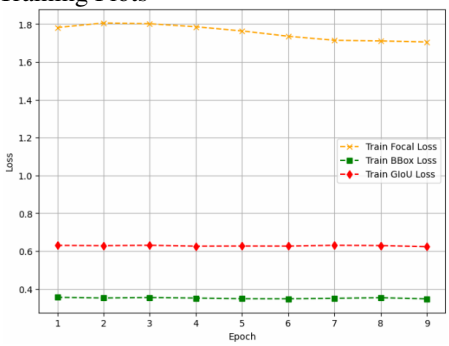


(b) Train Components over Epochs

Figure 5: YOLOS-S with LoRA Training Plots

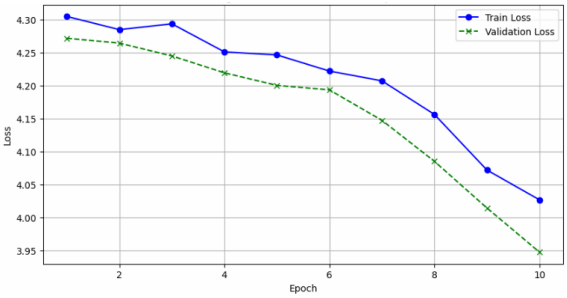


(a) Train and Validation Losses over Epochs

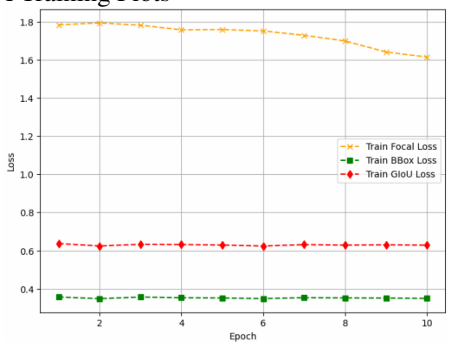


(b) Train Components over Epochs

Figure 6: YOLOS-S with AdaLoRA Training Plots

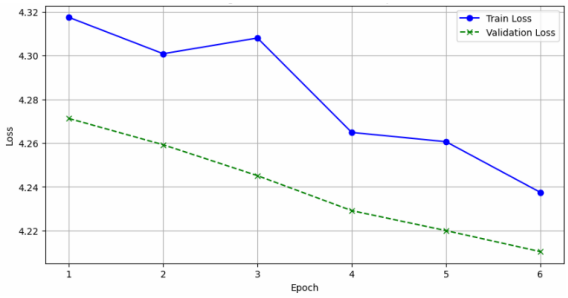


(a) Train and Validation Losses over Epochs

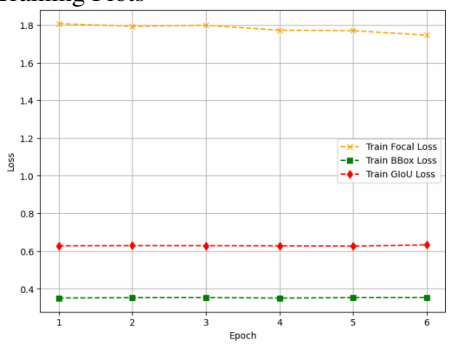


(b) Train Components over Epochs

Figure 7: YOLOS-S with LoHa Training Plots



(a) Train and Validation Losses over Epochs



(b) Train Components over Epochs

Figure 8: YOLOS-S with LoKr Training Plots