machine learning & knowledge extraction

MDPI

Article

Parameter-Efficient Fine-Tuning of Large Pretrained Models for Instance Segmentation Tasks

Nermeen Abou Baker *,†, David Rohrschneider *,† and Uwe Handmann

Computer Science Department, Ruhr West University of Applied Sciences, Lützowstraße 5, 46236 Bottrop, Germany; uwe.handmann@hs-ruhrwest.de

* Correspondence: nermeen.baker@hs-ruhrwest.de (N.A.B.); david.rohrschneider@hs-ruhrwest.de (D.R)

† These authors contributed equally to this work.

Abstract: Research and applications in artificial intelligence have recently shifted with the rise of large pretrained models, which deliver state-of-the-art results across numerous tasks. However, the substantial increase in parameters introduces a need for parameter-efficient training strategies. Despite significant advancements, limited research has explored parameter-efficient fine-tuning (PEFT) methods in the context of transformer-based models for instance segmentation. Addressing this gap, this study investigates the effectiveness of PEFT methods, specifically adapters and Low-Rank Adaptation (LoRA), applied to two models across four benchmark datasets. Integrating sequentially arranged adapter modules and applying LoRA to deformable attention—explored here for the first time—achieves competitive performance while fine-tuning only about 1–6% of model parameters, a marked improvement over the 40–55% required in traditional fine-tuning. Key findings indicate that using 2–3 adapters per transformer block offers an optimal balance of performance and efficiency. Furthermore, LoRA, exhibits strong parameter efficiency when applied to deformable attention, and in certain cases surpasses adapter configurations. These results show that the impact of PEFT techniques varies based on dataset complexity and model architecture, underscoring the importance of context-specific tuning. Overall, this work demonstrates the potential of PEFT to enable scalable, customizable, and computationally efficient transfer learning for instance segmentation tasks.

Keywords: finetuning; instance segmentation; adapters; LoRA; SEEM; MASK DINO; PEFT

Citation: Abou Baker, N.; Rohrschneider, D.; Handmann, U. Parameter-Efficient Fine-Tuning of Large Pretrained Models for Instance Segmentation Tasks. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 2783–2807. https://doi.org/10.3390/make6040133

Academic Editor: Vasile Palade

Received: 26 September 2024
Revised: 15 November 2024
Accepted: 29 November 2024
Published: 2 December 2024

1. Introduction

Large pretrained models have gained increasing attention due to their ability to generalize across various tasks. Models such as the Segment Anything Model (SAM) [1] demonstrate this versatility by handling new tasks with zero-shot predictions. While earlier models such as those based on convolutional networks have proven effective in specific domains, their ability to generalize to new tasks remains limited [2].

Traditional fine-tuning techniques involve adjusting the entire network or a subset of it in order to transfer knowledge to downstream tasks. However, this process often requires copying and updating the model's weights for each task, resulting in high computational and memory costs. In addition, such methods can lead to catastrophic forgetting, where the model loses previously acquired knowledge when fine-tuned for new tasks. Despite their strong starting point, large pretrained models struggle to maintain their ability to predict out-of-distribution tasks. As the size of pretrained models continues to grow, the computational cost of fine-tuning for task-specific purposes increases significantly. Moreover, the risk of overfitting to target datasets further complicates the process.

To address these challenges, the emerging field of PEFT has introduced techniques that significantly reduce the number of trainable parameters while maintaining or approaching the performance levels of full fine-tuning. By modifying a small subset of parameters,

PEFT not only reduces computational requirements but also mitigates the risks associated with overfitting.

Existing PEFT methods offer significant computational advantages, but face limitations that restrict their wider use in computer vision (CV) tasks. First, many techniques originally developed for natural language processing (NLP) are not optimized for unique challenges of vision tasks such as instance segmentation, which requires precise pixel-level predictions and the ability to differentiate between object instances. Second, while these methods have succeeded in simpler vision tasks, their effectiveness in more complex scenarios such as instance segmentation and optimal use for models based on multiscale deformable attention remains underexplored.

The motivation for this work stems from a notable gap in research regarding the application of PEFT techniques to contemporary DEtection TRansformer (DETR) models, particularly for instance segmentation tasks. Specifically, this study is motivated by two key areas of interest:

1. The lack of prior literature addressing the integration of adapters and LoRA within deformable attention modules.
2. Adapter modules usually yield limited scalability, as the bottleneck dimension is the primary adjustable parameter. Instead, a linearly scalable module can adapt more flexibly and efficiently to different model sizes and demands.

To address these problems, as well as the limited exploration of PEFT techniques within the context of instance segmentation, this paper makes the following contributions:

1. The first use of LoRA in deformable attention: We introduce a LoRA approach for multiscale deformable attention, achieving parameter efficiency by only updating low-rank matrices in the attention layers.
2. Sequential adapter integration for scalable finetuning: This study presents the first implementation of sequentially arranged adapter modules, particularly for two large pretrained instance segmentation models.

This study aims to open up new avenues of research by efficiently adapting large pretrained models to various vision applications. Additionally, the source code has been published to provide a valuable resource upon which the research community can build.

2. Fine-Tuning Techniques for Large Pretrained Models in Computer Vision

PEFT has proven highly effective in NLP, where models such as BERT and GPT are extensively fine-tuned for tasks such as sentiment analysis and machine translation [3]. As large pretrained Vision Transformer (ViT) models gain importance in CV, there is a growing need for computationally efficient fine-tuning methods in this domain as well. PEFT techniques such as adapters, LoRA, prefix tuning, and prompt tuning have shown great promise in reducing the number of trainable parameters, enabling a wide range of applications in NLP [4–8] without the high costs associated with traditional fine-tuning. The main difference between these methods lies in their approach to parameter reduction and its impact on the model structure.

2.1. Adapters

Adapters are one of the earliest approaches to PEFT. They consist of small trainable networks (adapter layers) between the layers of a pretrained model. Adapters typically work by adding a nonlinear transformation that helps the model to learn the task-specific features without changing the parameters of the main model. This reduces the required number of trainable parameters while mitigating the computational and memory costs associated with extensive model updates. Although this approach minimizes adjustments to the main model parameters, it can introduce additional complexity to the model architecture due to the inclusion of new layers. Moreover, these extra components added to the model structure may increase inference time. Further details on the proposed adapter location and architecture are described in Section 4.3.

2.2. Low-Rank Adaptation (LoRA)

LoRA is a PEFT technique designed to optimize large pretrained models by integrating trainable low-rank matrices into the attention layers. This method approximates weight updates using these low-rank matrices, thereby reducing the number of parameters required for fine-tuning. During this process, the core parameters of the model remain largely frozen, which minimizes memory usage and computational cost while speeding up adaptation [9]. LoRA's approach ensures that the architecture of the model remains efficient during inference, which is particularly effective for NLP and vision tasks. By focusing on a lower-dimensional space for parameter tuning, LoRA addresses the challenges of fine-tuning large pretrained models which may contain hundreds of millions of parameters. This technique aims to balance parameter efficiency and performance, making it a feasible solution for managing large-scale models in diverse applications.

2.2.1. Mathematical Framework and Efficiency of LoRA

Given a pretrained model with a weight matrix $W \in \mathbb{R}^{d\times k}$, the conventional fine-tuning approach updates W to $W + \Delta W$. Instead, LoRA represents the update ΔW as a product of two smaller matrices A and B:

$$\Delta W = B \times A \tag{1}$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and r is the rank of the decomposition, typically $r \ll \min(d, k)$. LoRA achieves a minimum amount of trainable parameters and computational complexity through this low-rank decomposition. The main benefits are:

- Parameter efficiency: Traditional fine-tuning requires updating $d \times k$ parameters in ΔW. LoRA reduces this to $r \times (d + k)$ parameters (the sum of elements in the A and B) matrices, significantly reducing memory requirements.
- Training efficiency: While the original model computes $W \times$ input with $d \times k$ operations, LoRA adds only $r \times (d + k)$ operations for $(B \times A) \times$ input. This means that during training, the original pretrained weights W are frozen and only the low-rank matrices A and B are trained. This approach preserves the pretrained knowledge while adapting to new tasks.
- Inference optimization: During inference, LoRA computes the effective weights as follows:

$$W_{\text{LoRA}} = W + B \times A. \tag{2}$$

  This allows the original weights W to remain unchanged, with only the smaller matrices A and B being updated and stored.
- Empirical effectiveness: Studies have shown that LoRA can achieve comparable performance to full fine-tuning on various tasks, as detailed in Section 3.

2.2.2. LoRA in Transformer Models and Hyperparameters

LoRA is typically applied to the attention layers in transformer-based models. Specifically, it modifies the query and value projection matrices in multihead attention mechanisms.

LoRA has two main hyperparameters: rank r, which controls the trade-off between parameter efficiency and model capacity by adjusting the size of the decomposed weight matrices, and a scaling factor α, which adjusts the magnitude of the LoRA update.

2.3. Prefix Tuning

Prefix tuning is another method that has been studied mainly in NLP. It involves freezing the parameters of a language model while optimizing a small and continuous task-specific vector, known as the prefix. Inspired by prompting, prefix tuning allows subsequent tokens to attend to this prefix as if it were a "virtual token" [10]. However, prefix tuning performs poorly in many NLP tasks due to its instability during training and its reliance on flawed evaluation protocols [11]. As discussed in the original LoRA

paper [9], increasing the number of special tokens in prefix tuning beyond a certain limit often leads to diminishing returns or performance drops. In contrast, LoRA scales better and consistently matches or exceeds baseline fine-tuning even for large models such as GPT-3, making it a more reliable approach for NLP tasks.

2.4. Prompt Tuning

Prompt tuning is a PEFT technique that introduces trainable *soft prompts* into the input sequence of a frozen pretrained language model [12]. This approach simplifies adaptation by learning these soft prompts end-to-end, which can efficiently encapsulate task-specific signals and carry them throughout the model.

While prompt tuning performs well in LLMs for NLP, it has limitations in CV applications. The main drawback is that it relies on text processing, which is not directly applicable to visual data. Additionally, the performance of prompt tuning is constrained by the input capacity of the model, which limits the amount of task-specific information it can encode. As a result, this method is less effective for tasks that require detailed visual feature extraction or fine-grained adaptation, where specialized architectures or other fine-tuning methods may be more appropriate.

Due to the above characteristics, this study explores the application of adapters and LoRA for instance segmentation tasks and focuses on their customization for large pretrained models. Our research evaluates the effectiveness of these parameter-efficient techniques in enhancing the performance of instance segmentation applications.

3. Related Work

3.1. Adapters in Vision Tasks

The idea of using adapters and transformers together in a compact and scalable architecture for NLP research was first introduced in [13]. Adapters were initially designed for multitasking and continual learning using a single BERT model shared among task-specific parameters. The *Vision Transformer Adapter* transferred this idea to CV, allowing a plain ViT [14] to be adapted to multiple dense prediction tasks [15]. The authors found that the plain variant reached its limits when transferred to tasks such as object detection or instance segmentation, and designed a parallel architecture to extract, augment, and inject task-specific information along the entire ViT.

Recent advances in this area include SAM, which has recently received increased attention for providing accurate segmentation masks based on different types of user prompts [1]. Previous attempts to fine-tune SAM have been investigated, as shown by Abou Baker et al. [16]. To leverage the base knowledge of the model learned during pretraining, this approach involved fine-tuning only the decoder, resulting in significant improvements that outperformed the state-of-the-art (SOTA) on multiple waste datasets. Further, Chen et al. [17] additionally implemented adapter modules to fine-tune SAM for the detection of camouflaged objects, shadows, and polyps in underrepresented scenes. The authors demonstrated the effectiveness of these modules, which outperformed the segmentation capabilities of plain SAM on five different datasets.

In the medical image segmentation domain, Medical SAM Adapter (Med-SA) [18] adds bottleneck adapters to both the encoder and decoder components. This method achieves superior performance over SOTA medical image segmentation techniques while updating only 2% of the model parameters during fine-tuning.

In multimodal large pretrained models, CLIP, originally a combination of visual and text embeddings for visual classification tasks [19], was further enhanced with adapters in the work of Gao et al. [20]. They added a bottleneck adapter after both the language and vision backbone and combined the result with the initial embeddings through a skip connection, realizing performance improvements on eleven different datasets.

3.2. LoRA in Vision Tasks

Large pretrained models trained using self-distillation with no labels [21] and its successor [22] have shown exceptional performance in many vision tasks; however, certain limitations have constrained their application in medical and surgical contexts. Nonetheless, significant progress has been made in several areas towards addressing these challenges via LoRA techniques. For example, Zhang et al. [23] integrated LoRA layers to improve diagnostic accuracy in capsule endoscopy. By freezing the parameters of the core model and selectively applying LoRA to the query and value projection layers in the transformer blocks, this method addresses the challenges posed by limited medical datasets and improves the adaptability of image classification. Evaluations on the Kvasir-Capsule and Kvasir-v2 datasets produced in impressive results, with the adapted model achieving accuracy of 97.75% and 98.81%, respectively, outperforming several SOTA Convolutional Neural Network (CNN) and ViT models. Additionally, LoRA-based adaptation reduced training time and memory requirements compared to full fine-tuning while maintaining strong visual representation capabilities. These results emphasize the efficiency of LoRA in adapting large pretrained models for specialized medical tasks such as capsule endoscopy diagnosis.

Similarly, Cui et al. [24] focused on depth estimation in robotic surgery, which is critical for tasks such as 3D reconstruction, surgical navigation, and augmented reality visualization. Their study introduced LoRA to produce a model called Surgical-DINO specifically designed for depth estimation in endoscopic surgery. By integrating LoRA layers and keeping the image encoder frozen, this approach allows for effective adaptation to surgery-specific domain knowledge without extensive fine-tuning. The Surgical-DINO model was evaluated on the MICCAI SCARED dataset and demonstrated superior performance compared to SOTA models in endoscopic depth estimation. The study shows that zero-shot prediction is insufficient for this particular case, and that LoRA adaptation is crucial for achieving high performance. Additionally, the study refers to a gap in current research, which mostly focuses on segmentation and detection tasks rather than pixel-wise regression tasks such as depth estimation. This work demonstrates how adapting large pretrained models with LoRA can significantly improve their applicability to specialized medical tasks, providing a promising direction for future research.

Another study investigated the application of LoRA to unsupervised domain adaptation for semantic segmentation tasks, addressing the challenges posed by large transformer models and their high computational requirements [25]. The study used LoRA to transfer models from synthetic datasets (GTA5) to real-world datasets (Cityscapes) and focused on improving training stability and efficiency. The authors integrated LoRA into a Swin transformer and TransDA framework, showing that LoRA effectively stabilized the self-training process. This approach achieved performance comparable to the exponential moving average (EMA) mechanism while reducing training time and memory usage by 11%. The authors demonstrated the potential of LoRA as a computationally efficient alternative for domain adaptation and illustrated its effectiveness in improving semantic segmentation performance with reduced resource requirements.

Generalized LoRA (GLoRA) [26] is an advanced method for PEFT that significantly improves performance on the VTAB-1K benchmark. GLoRA extends LoRA by adding a generalized prompt module to optimize both model weights and intermediate activations, providing greater flexibility and efficiency for a variety of tasks. The study shows that GLoRA outperforms existing methods in the VTAB-1K benchmark by up to 2.9%, achieving superior accuracy with fewer parameters and reduced computational requirements. GLoRA maintained high generalization capabilities across 14 out of 19 datasets, proving its effectiveness in bridging the gap between parameter efficiency and model performance.

To address the challenge of efficiently fine-tuning large models for image generation, SuperLoRA is a novel framework that extends LoRA for fine-tuning large models [27]. SuperLoRA extends traditional LoRA by including techniques such as grouping, folding, shuffling, projecting, and tensor factoring, which significantly improves flexibility and

performance, especially in scenarios with extremely limited parameters. SuperLoRA was evaluated through transfer learning tasks, including image classification and image generation, and demonstrated superior parameter efficiency compared to existing LoRA variants. SuperLoRA achieved up to a ten-fold reduction in the number of parameters while maintaining or improving performance, proving highly effective in low-parameter scenarios.

Building on this, a recent study applied LoRA to fine-tune a stable diffusion model for generating synthetic images of defects (e.g., scratches, cracks, and pits) in the NEU-seg dataset [28]. To overcome data scarcity and class imbalance, synthetic images were used to augment the training data for segmentation models (DeepLabV3+ and FPN). This method led to significant improvements in segmentation performance, with the mean Intersection-over-Union (mIoU) increasing by about 6%. The key contribution was the use of LoRA to efficiently adapt stable diffusion for high-quality synthetic image generation, which improved the robustness of defect segmentation.

The related works discussed above show that adapters and LoRA techniques are effective in a variety of domains, including NLP and certain CV tasks such as image classification and semantic segmentation, as summarized in Table 1. However, their application to more specialized and complex CV problems, particularly instance segmentation, remains underexplored.

Table 1. Summary of related works and their classification into the topics of *Adapters in CV* and *LoRA in CV*.

Publication	***Adapters in CV***	***LoRA in CV***
Vision Transformer Adapter for Dense Predictions [15]	✓	
SAM-Adapter: Adapting Segment Anything in Underperformed Scenes [17]	✓	
Medical SAM adapter: Adapting segment anything model for medical image segmentation [18]	✓	
Clip-adapter: Better vision-language models with feature adapters [20]	✓	
Learning to Adapt Foundation Model DINOv2 for Capsule Endoscopy Diagnosis [23]		✓
Surgical-DINO: adapter learning of foundation models for depth estimation in endoscopic surgery [24]		✓
Low Rank Adaptation for Stable Domain Adaptation of Vision Transformers [25]		✓
Latent Diffusion Models to Enhance the Performance of Visual Defect Segmentation Networks in Steel Surface Inspection [28]		✓
This work	✓	✓

Instance segmentation presents a unique challenge because it requires not only the identification of semantic classes at the pixel level but also the grouping of these pixels into distinct object instances. This task involves separately predicting both the mask and the category for each object instance, adding a layer of complexity beyond what existing adaptation methods in CV have primarily supported.

To the best of our knowledge, this work pioneers the application of adapters and LoRA to instance segmentation using large pretrained models. By addressing this underexplored area, our research aims to extend the capabilities of adaptation methods to handle complex

visual tasks and contribute to the advancement of efficient and adaptable CV models for instance segmentation.

4. Models and Architecture

4.1. Large Pretrained Models

This study examines two specialized large pretrained models for instance/panoptic segmentation: the Segment Everything Everywhere Model (SEEM) [29] and DETR with Improved Denoising Anchor Boxes (Mask DINO) [30]. SEEM is a multimodal segmentation model that accepts prompts such as clicks, boxes, polygons, scribbles, text, and referring regions from another image to control the focus of the detection during inference. Inspired by the architecture of CLIP [19], it also incorporates a text encoder to enable open-set segmentation capabilities, and as such can perform so-called "zero-shot inference" on unseen datasets. With approximately 340 million parameters, SEEM provides a robust framework for various types of input data.

Interactive models are very helpful for image labeling, which remains a major challenge in CV. In the work of Huang et al. [31], a novel attention mechanism was proposed that focuses primarily on scribbles to enhance the contrast between background and foreground objects. This approach reduces the need for dense annotations, increases the efficiency of automatic labeling, and achieves high-quality segmentation with minimal user input. In a more recent work by Meta, SAM2 extends upon its predecessor by allowing the segmentation of specific objects throughout videos with one or more interactive prompts on the first frame. It is designed to provide a real-time experience while achieving similar or better results than SAM [32].

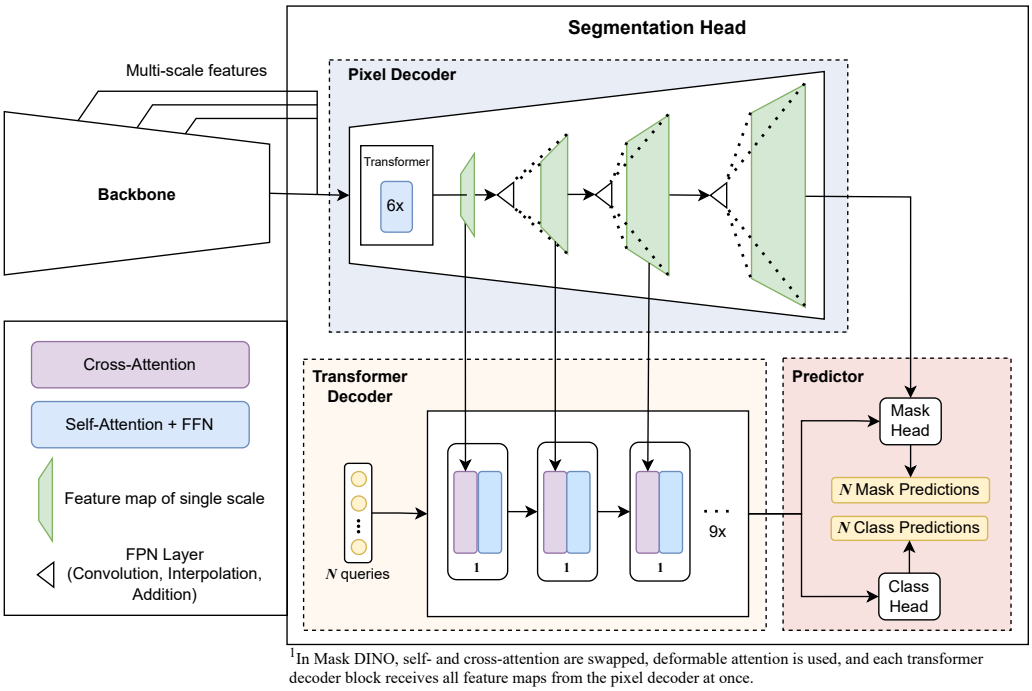
Mask DINO includes two versions provided by the authors: one based on the ResNet backbone (with 52.02 million parameters), and another based on the Swin-L backbone (with 222.76 million parameters). Mask DINO is a large pretrained model for object detection and segmentation. It adopts the same architecture as the plain DINO model, but modifies the transformer decoder slightly to enable output of instance-wise segmentation masks. This is achieved by adding a parallel branch dedicated to mask prediction alongside DINO's bounding box prediction branch. Mask DINO outperforms existing instance segmentation methods based on the ResNet50 [33] and Swin-L [34] backbones, achieving 54.5 average precision (AP) on the COCO dataset.

The methodology proposed in this work was applied to both SEEM and Mask DINO, as they are based on the segmentation *meta-architecture* proposed by the authors of MaskFormer [35] and deformable DETR [36], simplifying the process of code adjustments during implementation. Additionally, they offer a variety of backbone methods to choose from, including various sizes of FocalNet [37] or ViT [14] for SEEM and the choice of ResNet50 [33] or Swin-L [34] for Mask DINO. Unlike SEEM, Mask DINO is not a multimodal model, allowing the proposed techniques to be compared for both types of large pretrained models. As a replacement for the regular multihead self-attention layer, we implemented both models to support the use of multiscale deformable attention. However, only the authors of Mask DINO made use of this replacement during the creation of the publicly available weights, resulting in a minor architectural difference in the transformer blocks compared to SEEM.

4.2. Architecture

Figure 1 provides a high-level overview of the previously mentioned *meta-architecture*. The models consist of four components: The **backbone**, which produces multiple image embeddings of different sizes; **a pixel decoder** to upsample the embeddings into high-resolution feature maps gradually; a **transformer decoder** that generates N embeddings for possible instance candidates; and a **predictor** to convert and filter these candidates to actual class and mask predictions. The pixel decoder is built upon a three-level feature pyramid network (FPN) [38] enhanced by a transformer encoder with six layers that captures the global image features and efficiently extracts high-resolution per-pixel features.

Each of the nine transformer decoder blocks consists of three layers: a cross-attention module with layer normalization (LN), which receives the image features produced by the pixel decoder; a self-attention module with LN; and a feed-forward network (FFN) with LN, based on the method proposed by Cheng et al. [39]. Inside the predictor, the output of the transformer decoder is combined with a learnable *class embedding*, resulting in the class logits. In Mask DINO, a softmax function generates class probabilities. As an open-set segmentation model, SEEM calculates the cosine similarity between these class logits and the embeddings of the textual input to derive the probabilities. Mask predictions are obtained by feeding the same decoder output through a simple MLP, called *mask embedding*, and linearly combining these outputs with the high-resolution feature map of the *pixel decoder*. In Figure 1, these operations take place in the Mask and Class Head, respectively. Throughout this work, the composition of all the modules after the backbone is called the *segmentation head*.



[1]In Mask DINO, self- and cross-attention are swapped, deformable attention is used, and each transformer decoder block receives all feature maps from the pixel decoder at once.

Figure 1. High-level view of the model architecture of Mask DINO and SEEM.

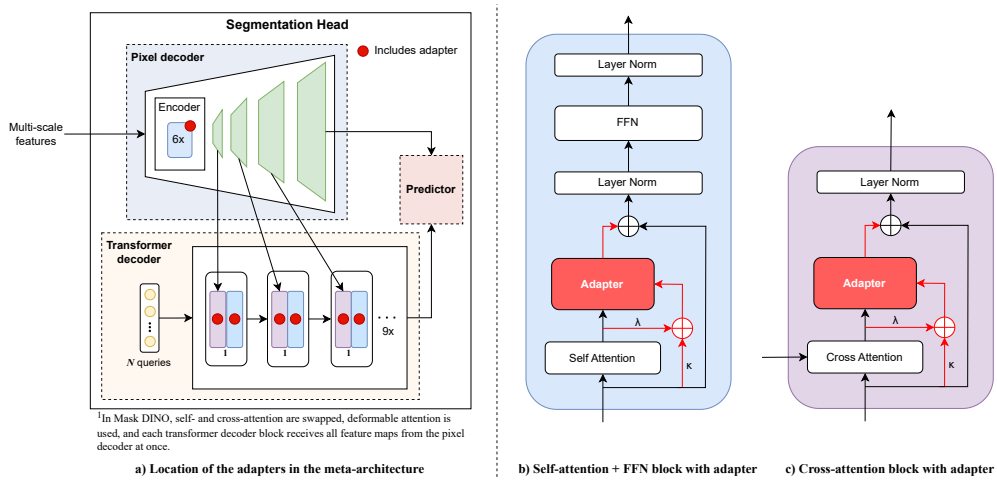
4.3. Adapters

The proposed adapter architecture is inspired by the NLP-based approach introduced by Houlsby et al. [13]. In their approach, a linear layer is used to project down the output of the intermediate transformer layer, an activation function is used to introduce nonlinearity at the bottleneck, a linear layer is used to project up to the original input size, and a residual connection is established to preserve important information about the input signal. In the original paper, the adapters were placed after each feed-forward network of the BERT transformer layers, resulting in two adapters per layer. Considering these aspects, the proposed method introduces three main architectural decisions to transfer and extend this idea to segmentation frameworks.

4.3.1. Adapter Location

Because the main purpose of the adapters is to "translate" the knowledge of the pretrained transformer to a downstream task, the adapter modules have been placed directly after all cross- and self-attention layers of each transformer block (highlighted in red-colored blocks in Figure 2b,c below). This location ensures that the adapters receive the original attention maps of the network as a prior and are trained to apply the nonlinear

transformation. Consequently, the outputs can be seen as modified attention maps that contain additional task-specific information for the downstream tasks.



[1]In Mask DINO, self- and cross-attention are swapped, deformable attention is used, and each transformer decoder block receives all feature maps from the pixel decoder at once.

a) Location of the adapters in the meta-architecture **b) Self-attention + FFN block with adapter** **c) Cross-attention block with adapter**

Figure 2. Location of adapters inside the meta-architecture (**a**) and detailed placement inside a self-attention block (**b**) or cross-attention block (**c**). Added components are colored red.

4.3.2. Adapter Architecture

The internal adapter architecture closely follows the original implementation mentioned earlier [13], consisting of a linear downprojection, a ReLU activation function, a linear upprojection, and a residual input $\oplus$ added to the adapter output, as shown in Figure 3 (the blue block). The bottleneck dimension is defined and calculated as follows:

$$dim_{bottleneck} = dim_{input} * \frac{1}{k} \tag{3}$$

where k is the downscale factor, which is a tunable hyperparameter shared by all adapter modules. Increasing k reduces the dimension of the bottleneck, effectively leading to a common tradeoff between compressing the data to capture high-level relations and losing relevant information due to massive dimension reduction. Furthermore, because the attention prior is highly informative and may require a deeper neural network than just a single MLP to capture all relevant information, another hyperparameter I defines the number of repetitions of the MLP within each adapter module. Fundamentally, all repetitions receive the same residual input, meaning that they refer to the original signal throughout the sequential alignment.

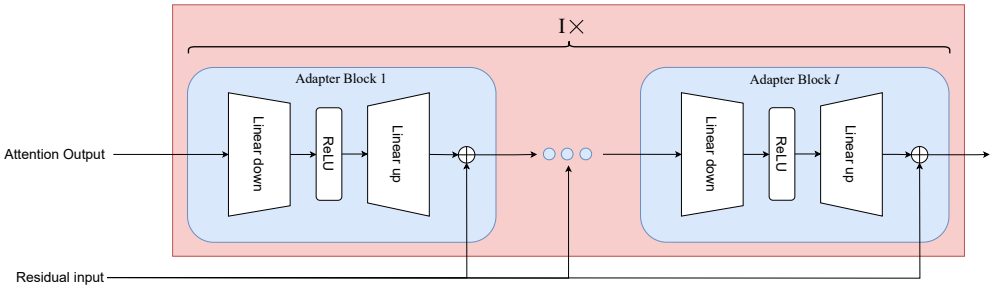


Figure 3. An adapter block consists of a simple feed-forward network and a residual connection (blue block). This block is sequentially repeated I times inside a single adapter (red frame). Each of the I blocks receives the same residual input.

4.3.3. Residual Connection

In contrast to the original implementation, where the residual input serves as a skip connection to add up the information before the adapter module, we observed that com-

bining the attention output with the key vector of the attention input yields slightly better overall performance. This is discussed further in Section 5.2. This improvement occurs even though the models are designed to include this signal inherently and add it to the output of the adapter in both cases (as shown in Figure 2b,c above the adapter blocks). The interpretation of this behavior is that emphasizing the attention's key vector **after** the adapter modules forces the adapters to focus more on learning how to fine-tune the pretrained attention layers, rather than on learning the relationships between the input features from scratch. Using the attention prior only as a residual input is referred to as the κ configuration. In contrast, residual use of the attention output is referred to as the λ configuration. The combination of both, shown in Figure 2b,c, is referred to as the $\kappa + \lambda$ configuration.

4.4. LoRA Layers

Similar to the adapter modules, LoRA was applied to each of the six transformer encoder layers within the pixel decoder and to all nine blocks of the transformer decoder, as illustrated in Figure 4.

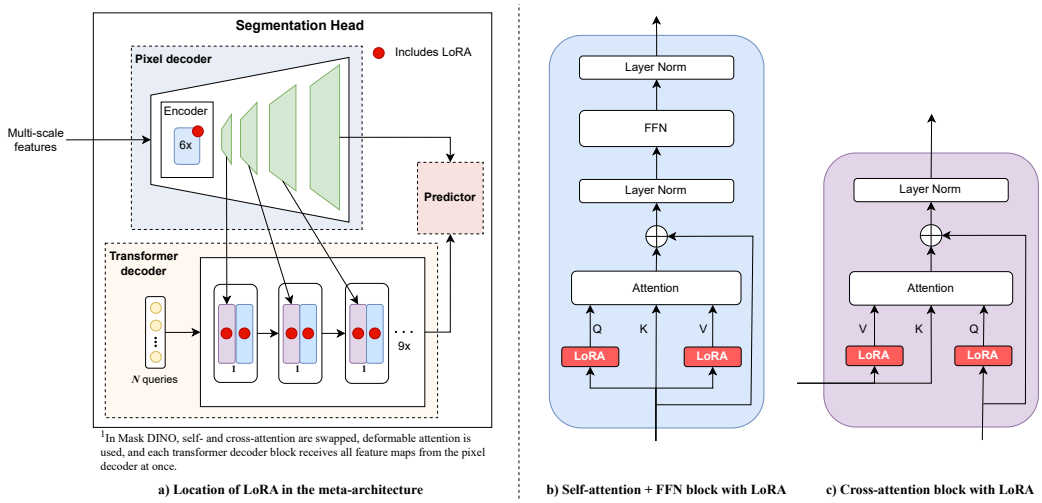


Figure 4. Location of LoRA inside the meta-architecture (**a**) and detailed placement inside a self-attention block (**b**) or cross-attention block (**c**). Added components are colored red.

According to the original LoRA implementation [9], the input sequence is processed in parallel by the base weights W_0 and the LoRA weights BA, scaled by a factor $\frac{\alpha}{r}$. **Then**, the results are added, leading to the following expression:

$$h = xW_0^T + \frac{\alpha}{r}x(BA)^T. \tag{4}$$

However, this calculation involves applying two matrix multiplications to the input sequence, one with the base weights and another with the LoRA weights. Thus, the LoRA weights must also be kept separate during inference, potentially increasing the inference speed compared to the plain model. To keep the latency as low as possible, we decided to use the "LoRA-Torch" Python library [40]. This efficiently bypasses the issue by **first** adding the scaled LoRA weights to the base weights, **then** applying a single matrix multiplication to the input sequence, leading to the following equation:

$$h = x(W_0 + \frac{\alpha}{r}BA)^T. \tag{5}$$

For each task adaptation, the weight matrix B is initialized as a zero matrix, allowing the model to begin with the pretrained weights. As training progresses, task-specific weights are added incrementally, ensuring that the model can learn new tasks while preserving

the knowledge encoded in the pretrained parameters. During training, recomposition of A and B is performed before each forward pass, then reversed by subtracting the scaled LoRA weights. This process isolates the contribution of LoRA-specific weights from the base weights, allowing the loss to be correctly back-propagated. For inference, the trained weights can be precomputed and then added to the model's weights to generate a fine-tuned version of the model, minimizing the computational latency during inference.

Because the authors of Mask DINO replaced the regular self-attention in the pixel decoder and the cross-attention layers in the transformer decoder with the deformable attention mechanism, in this work we propose a paradigm to apply LoRA to both variants while maintaining its functionality. This distinction and the previously mentioned weight addition are illustrated in Figure 5 and described below.

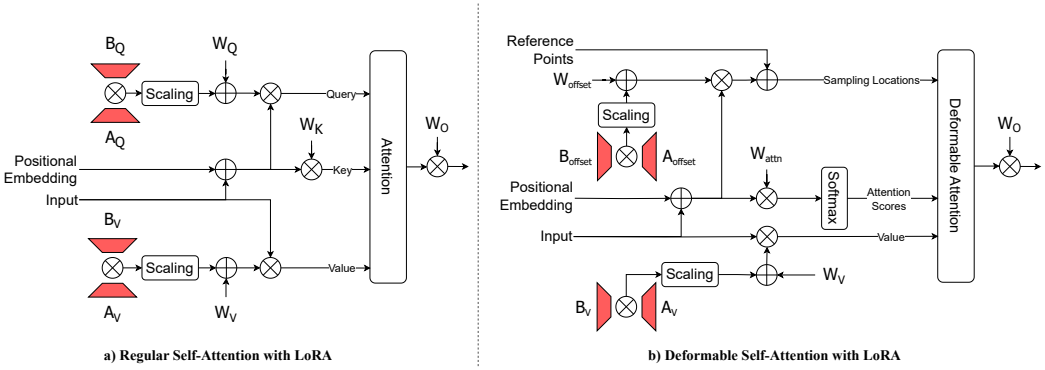


Figure 5. Application of LoRA to regular and deformable attention mechanism.

4.4.1. Regular Self- and Cross-Attention

In the regular self- or cross-attention mechanism, Q, K, and V are calculated by a matrix multiplication between the input sequence and corresponding weight matrices W_Q, W_K, and W_V. After the actual attention operation, the output is multiplied by a fourth set of weights W_O to apply a final linear transformation. This operation is expressed mathematically in Equation (6); for simplicity, the equation represents the self-attention operation with only a single head:

$$\begin{aligned} &Q = xW_Q^T, K = xW_K^T, V = xW_V^T, \\ &Output = Attention(Q, K, V)W_O^T. \end{aligned} \tag{6}$$

Based on findings from the original LoRA paper [9], this operation is applied only to the weight matrices of the query (W_Q) and value (W_V) projections, leaving the key (W_K) and output (W_O) projections untouched. To achieve this, Equation (6) is extended as follows:

$$\begin{aligned} Q &= x(W_Q + \frac{\alpha}{r} B_Q A_Q)^T \\ K &= xW_K^T \\ V &= x(W_V + \frac{\alpha}{r} B_V A_V)^T \\ Output &= Attention(Q, K, V)W_O^T, \end{aligned} \tag{7}$$

where the decomposed LoRA matrices (A_Q, B_Q, A_V, B_V) are trained during fine-tuning. In Figure 5a, the trainable matrices are visualized in red. After recomposition, they are added ($\oplus$) to the base weights, and the remainder follows the conventional attention mechanism. For cross-attention, LoRA was applied to the same weight matrices as in self-attention, with the only difference being that in this case the key and value matrices are derived from a different input sequence than the query matrix, allowing the model to attend to an external source of information.

4.4.2. Deformable Self- and Cross-Attention

The regular attention mechanism involves calculating attention scores between all elements of the query and key sequences, resulting in quadratic complexity with respect to the sequence length. This issue becomes particularly concerning for images, where computational complexity scales quadratically for **both** height and width.

To mitigate this issue, Zhu et al. [36] proposed the deformable attention mechanism, which restricts each element so that it only attends to a fixed number of points in the sequence. This approach makes it computationally feasible to execute self-attention operations on large sequences, for instance a a flattened concatenation of image patches obtained from multiple scales of the backbone architecture.

To accomplish this, K static reference points ($\hat{p}$) are predefined as an evenly distributed grid within the spatial dimensions of the input image. Based on the query sequence, the model predicts offsets Δp and attention weights $\hat{A}$ for each reference point. These offsets are added to the reference points to generate sampling locations p.

After the offsets are applied, a softmax function is used to normalize the attention weights A. The corresponding points are sampled from the value sequence and weighted according to these normalized attention scores. To produce the final output, a linear transformation using another weight matrix W_O is applied to the result.

In the original implementation [36], the offsets are calculated by multiplying the query sequence and a weight matrix W_{offset}. Similarly, the attention weights are obtained with a separate weight matrix W_{attn}, and the value projection remains the same as in the regular attention mechanism (W_V). Equation (8) mathematically expresses the deformable self-attention operation, simplified to include only a single head.

$$\begin{aligned} &\Delta p = xW^T_{offset}, \hat{A} = xW^T_{attn}, V = xW^T_V \\ &p = \hat{p} + \Delta p \\ &A = softmax(\hat{A}) \\ &Output = DeformableAttention(V, p, A)W^T_O \end{aligned} \tag{8}$$

Deformable attention uses a fixed number of static reference points with corresponding offsets. Instead of traditional query and key projections, the LoRA adaptation adjusts these offsets and attention weights through low-rank matrices during fine-tuning.

Theoretically, the shifted reference points, sometimes called sampling points, can be seen as queries, while the weights W_{attn} themselves can be seen as keys. Building upon this mental concept, LoRA is applied to the weights W_{offset} and W_V, resulting in the decomposed matrices A_{offset}, B_{offset}, A_V and B_V.
Mathematically, this LoRA adaptation of a deformable self-attention layer can be expressed as shown below.

$$\begin{aligned} &\Delta p = x(W_{offset} + \frac{\alpha}{r} B_{offset} A_{offset})^T \\ &\hat{A} = xW^T_{attn} \\ &V = x(W_V + \frac{\alpha}{r} B_V A_V)^T \\ &p = \hat{p} + \Delta p \\ &A = softmax(\hat{A}) \\ &Output = DeformableAttention(V, p, A)W^T_O \end{aligned} \tag{9}$$

The deformable self-attention mechanism is visualized in Figure 5b, where the trainable decomposed LoRA weights are again colored in red. In this variant, the input sequence is used to compute the offsets, attention weights, and value projection.

Complementing this, the cross-attention variant receives the value from a different sequence to facilitate the interaction between two sources. As in the self-attention variant, LoRA is applied to the offset and value projection weights. In the context of task adaptation,

this updates the model's understanding of which points to look at by adjusting the offsets of the static reference points and how to interpret the inputs by adjusting the value projection.

5. Datasets and Methodology

5.1. Datasets

For a comprehensive evaluation across datasets of varying complexity, 4 datasets were selected to serve as downstream tasks:

- The Northumberland Dolphin Dataset 2020 (NDD20) consists of 2201 images of each of two different dolphin species, including both above- and below-water images, for a total of 4402 images, along with 6102 annotations [41]. As the second species is only present in the underwater images, no distinction between species was made during preparation and the two types of images were concatenated and shuffled to create a training–testing split. The images and annotations are of high quality, making the resulting dataset relatively less complex compared to the others.
- The ZeroWaste dataset is a unique and challenging benchmark dataset that poses numerous challenges, including significant clutter, highly deformable and translucent objects, and a fine-grained difference between the object classes. It consists of real images taken from a conveyor belt in a materials recovery facility (MRF). The fully annotated partition, *ZeroWaste-f*, is unbalanced and consists of 4503 images with four possible classes: Cardboard, Soft Plastic, Rigid Plastic, and Metal. The SOTA for the AP is equal to 24.2 [42]; in the context of this work, the ZeroWaste dataset can be considered a medium-complexity task.
- The Waste Inspection X-ray (WIXray) dataset is a benchmark smart waste inspection dataset that introduces the real and novel problem of instance segmentation in X-ray images. It consists of 5038 X-ray images (total of 30,881 annotated waste items) with a constant resolution of 450×450, and includes four general types and twelve categories of small objects: Recyclable (PlasticBottle, Can, Carton, GlassBottle, Stick, and Tableware), Residual (HeatingPad, Desiccant, and MealBox), Foodwaste (FoodWaste), and Hazardous (Battery and Bulb). The SOTA for the AP is equal to 46.85 [43]; due to the combination of rather small-sized images, occlusion, and overlapping instances due to the nature of X-ray scans, the dataset can be considered a high-complexity task in this research.
- The Cityscapes dataset [44] focuses on semantic understanding of urban street scenes, and consists of 5000 finely annotated and 20,000 coarsely annotated images. It covers 30 classes such as Road, Sidewalk, Vehicle, and Building captured across 50 cities over several months during daytime and in good to medium weather conditions. This well-known benchmark dataset includes labels on the pixel, instance, and panoptic segmentation levels. In this study, we focus only on instance segmentation with fine-grained labels, for which this dataset consists of 5000 images (2975 for training, 500 for validation, and 1525 for testing) containing annotations for eight classes. The current SOTA for the AP is equal to 49.3, which was achieved by training the OpenSeeD model end-to-end [45].

The four levels of complexity also result from the contextual divergence from the datasets used for pretraining the large pretrained models. It can be hypothesized that downstream tasks that are fundamentally different from pretrained (common) knowledge require greater computational adjustments by the implemented adapter modules.

5.2. Experimental Setup

SEEM and Mask DINO were used as base models. The training procedure used an NVIDIA Quadro RTX 8000 with a batch size of 4 in all cases. The experiments were performed with different epochs for each dataset. Specifically, NDD20 was trained with 15 epochs, Zerowaste with 10 epochs, and both WIXray and Cityscapes with 20 epochs. The performance of the models was evaluated based on the weights that produced the best results. The original optimizer (AdamW for both models) and loss function were

kept throughout the study; after the sweep of different learning rate settings, the final base learning rates were 0.001 for adapter runs and 0.0001 for traditional runs. Notably, the Cityscapes dataset was trained with a more complex backbone (Swin-L) in the case of Mask DINO to allow for a fair comparison with the latest SOTA results.

To highlight the parameter efficiency, this approach was compared to several traditional fine-tuning settings without the use of LoRA or adapter modules. Because full fine-tuning requires updating 100% of the parameters, which is computationally expensive, we performed ablation studies on traditional training of the following specific parts of the models: fine-tuning the full segmentation head (encoder + decoder), fine-tuning only the decoder, and fine-tuning only the class + mask embeddings.

The different adapter settings included {1, 2, 3, 4} adapters per block ($I = 1, 2, 3, 4$)). The residual connection setting mentioned in Section 4.3.3 requires further discussion. Two additional experiments were performed on the NDD20 dataset (see Table 2a below). The setting $\boldsymbol{\kappa}$ refers to using only the attention key as the residual adapter input; in contrast, the setting $\boldsymbol{\lambda}$ refers to using the attention mechanism's output, while $\boldsymbol{\kappa} + \boldsymbol{\lambda}$ refers to adding both before feeding them into the adapter as a residual connection. The resulting AP values are documented in Table 2a, where bold numbers represent the highest AP scores for each dataset and model.

Table 2. Ablation study of the four datasets with different fine-tuning settings and numbers of parameters for each setting. The adapter and LoRA configurations also include the trainable class and mask embeddings. (a) Ablation study for all four datasets. (b) Ablation study of trained parameters. Italic rows represent table headings.

(**a**) Ablation study for all four datasets; CE and ME refer to class and mask embedding, respectively.
$*$ uses the Swin-L backbone, bold numbers represent the highest score within a column

AP scores	*NDD20*		*ZeroWaste*		*WIXray*		*Cityscapes*		*Average*
	SEEM	*Mask DINO*	*SEEM*	*Mask DINO*	*SEEM*	*Mask DINO*	*SEEM*	*Mask DINO*$*$	
Full Head	**79.2**	**78.98**	**25.97**	24.94	**39.88**	**44.17**	**32.19**	39.11	**45.55**
Only Decoder	71.49	66.28	14.65	14.26	30.89	28.72	31.27	36.83	36.8
Only CE & ME	67.79	56.06	5.17	6.99	16.89	6.62	29.59	34.16	27.91
1 Adapter (κ)	76.26	62.53	-	-	-	-	-	-	-
1 Adapter (λ)	76.79	75.55	-	-	-	-	-	-	-
1 Adapter ($\kappa + \lambda$)	77.19	76.54	22.6	22.66	34	36.96	31.58	37.23	42.34
2 Adapters ($\kappa + \lambda$)	77.72	77.58	24.2	24.7	34.45	39.6	31.7	37.71	43.461
3 Adapters ($\kappa + \lambda$)	77.67	77.06	22.9	24.64	35.28	40.5	32.08	38.27	43.55
4 Adapters ($\kappa + \lambda$)	78.15	76.71	24.6	**25.02**	35.27	41.32	32.04	37.77	43.86
LoRA (r=2)	77.29	74.72	22.6	20.35	32.97	29.4	31.22	38.67	40.9
LoRA (r=4)	77.06	75.47	23.1	21.7	34.95	33.48	31.26	39.85	42.11
LoRA (r=8)	76.89	76.32	23.9	22.59	35.06	36.5	31.77	40.24	42.91
LoRA (r=16)	76.52	76.56	24.5	22.34	34.01	38	31.3	**40.39**	40.95

(**b**) Ablation study for the average number of trained parameters and their corresponding fraction of the resulting model.

# Parameters	*SEEM* (Focal-L)	*Mask DINO* (Resnet-50)	*Mask DINO (Swin-L)*
Full Head	134.61*M* (39.55%)	28.56*M* (54.91%)	27.56*M* (12.37%)
Only Decoder	102.42*M* (30.09%)	14.51*M* (27.89%)	14.73*M* (6.61%)
Only CE & ME	1.05*M* (0.31%)	0.33*M* (0.63%)	0.33*M* (0.15%)
1 Adapter ($\kappa + \lambda$)	4.21*M* (1.23%)	1.12*M* (2.13%)	1.12*M* (0.50%)
2 Adapters ($\kappa + \lambda$)	7.37*M* (2.13%)	1.92*M* (3.58%)	1.92*M* (0.86%)
3 Adapters ($\kappa + \lambda$)	10.53*M* (3.01%)	2.71*M* (4.99%)	2.71*M* (1.21%)
4 Adapters ($\kappa + \lambda$)	13.7*M* (3.88%)	3.51*M* (6.35%)	3.51*M* (1.55%)
LoRA (r=2)	1.15*M* (0.34%)	0.38*M* (0.73%)	0.38*M* (0.17%)
LoRA (r=4)	1.25*M* (0.37%)	0.43*M* (0.83%)	0.43*M* (0.19%)
LoRA (r=8)	1.44*M* (0.42%)	0.53*M* (1.01%)	0.54*M* (0.24%)
LoRA (r=16)	1.84*M* (0.54%)	0.73*M* (1.39%)	0.74*M* (0.33%)

A parameter sweep was conducted to identify the optimal scaling factor for the LoRA configurations. The configurations included $r = \{2, 4, 8, 16\}$, while always using $\alpha = r$ to attain a constant scaling factor of 1. This scaling was found to perform the best out of different α values during a separate ablation study on the WIXray dataset, which can be found in Appendix A.

SEEM integrates a dedicated language model that processes class names by injecting them into predefined prompts to generate the corresponding embeddings. The adapters and LoRA modules could potentially be introduced at this stage; however, as the primary function of the pretrained language model is to generate embeddings from textual input regardless of the dataset, we decided not to incorporate adapters or LoRA modules into the model at this stage, leaving this extension open for future work.

In Table 2, CE represents the class embedding and ME represents the mask embedding. Unless otherwise noted, the residual $\boldsymbol{\kappa} + \boldsymbol{\lambda}$ setting is used for the rest of the paper.

6. Discussion

This section presents a comparative analysis of adapter-based methods and LoRA, evaluating their performance and efficiency across different configurations and tasks. Quantitative and qualitative results are provided along with a comparison of inference speed to assess their respective advantages and limitations.

6.1. Quantitative Results

6.1.1. Analysis of Adapter Performance

This section provides an analysis of the performance of different adapter configurations compared to full-head tuning on different datasets and with different settings.

For the AP, it was observed that full-head tuning generally provides slightly better results than adding adapters for both SEEM and Mask DINO in most configurations. However, there was an exception in the case of Mask DINO for the Zerowaste dataset, where the setting with four adapters outperformed both the full-head setting and even the SOTA. A more comprehensive comparison is visualized and discussed later using Figures 6 and 7.

In addition, we found that tuning only the decoder or only the class and mask embeddings led to a significant decrease in AP, emphasizing that removing tunable parameters in the final layers is inferior to adding adapters with similar or fewer parameters.

Furthermore, while the performance of SEEM and Mask DINO varied across the datasets, on average it followed the trend of increasing performance as more adapters are added. Although the Mask DINO architecture with ResNet-50 backbone is more than six times smaller than SEEM, it achieved comparable or better performance across our experiments. This suggests that the deformable attention-based detection branch, the contrastive denoising in the underlying DINO method, and the absence of a language encoder help the model to localize objects more efficiently before creating the segmentation masks. Mask DINO was also found to perform better in predicting small objects. Thus, higher AP of Mask DINO regarding the WIXray dataset was observed for the approach with full-head fine-tuning, as shown in Table 2a.

Table 2b shows a significant reduction in parameters when using adapter configurations compared to the full-head and decoder-only settings. This reduction demonstrates the benefit of using adapters to achieve parameter efficiency rather than modifying a subset of the underlying model. The percentages were calculated as follows:

$$percentage_{trained} = \frac{100 \times num_{trainable}}{num_{original} + num_{adapters}}. \tag{10}$$

For the Cityscapes benchmark, the best AP score after 20 epochs was achieved by Mask DINO, with a score of 38.27 under the three-adapter setting and 39.11 when fine-tuning the full segmentation head. Compared to the current SOTA for this benchmark dataset (49.3 AP for the"OpenSeeD" multimodal model [45]), the results indicate the potential of

adapter fine-tuning, as the training was performed with only a small number of additional parameters (1.21%) rather than training the model end-to-end. The previous SOTA AP was equal to 46.7, achieved by training a $372M$-parameter model ("OneFormer" [46]) end-to-end for $90K$ optimization steps and a batch size of 16, making up a total of 484 epochs. To better compare the effectiveness of using adapters, the training of Mask DINO was extended with four adapters for an additional 30 epochs while keeping all the hyperparameters constant. The best AP score we obtained was 41.44, while the model was still not observed to converge or overfit. Although the method still leaves a performance gap with SOTA for Cityscapes, it can be acknowledged that adapters could close this gap with further fine-tuning and optimization, providing an efficient framework for transferring pretrained knowledge to downstream tasks.

In our results, the adapter settings achieved competitive results with only 1.23–3.88% (SEEM) and 0.5–6.35% (Mask DINO) of the total model parameters. In contrast, fine-tuning the entire segmentation head resulted in a high percentage of trainable parameters, 39.55% for SEEM and 54.91% for Mask DINO.

To directly compare the methods with the full-head configuration, the average AP was calculated for each method across all four datasets, then the relative difference between each of the resulting mean AP scores and the mean full-head AP was calculated as follows:

$$\delta_m = \frac{\overline{AP_m} - \overline{AP_{fullhead}}}{\overline{AP_{fullhead}}} \tag{11}$$

where $\overline{AP_m}$ is the mean AP for method m and $\overline{AP_{fullhead}}$ is the mean AP for the full-head method; in other words, the delta value δ measures how far a method is from the best-performing setting (full-head tuning) on a uniform scale. The results are shown on the y-axis in Figure 6 for the SEEM model and Figure 7 for the Mask DINO model. Additionally, the figures show the number of trainable parameters on the x-axis.

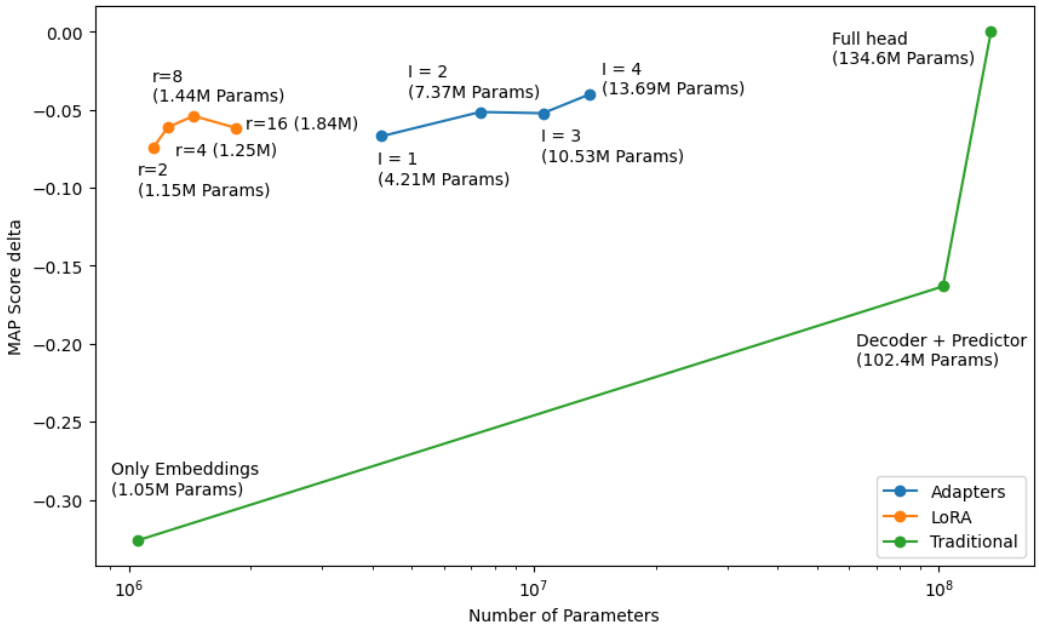


Figure 6. Delta δ comparison of SEEM results

Intuitively, a higher number of adapters leads to better the performance, and this proves common for all four datasets; therefore, finding the tradeoff between model precision and number of parameters remains a challenge. The AP performance when using adapters is close to that achieved by fine-tuning the full segmentation head. Although the traditional method yields slightly better results than the setting with two adapters, it requires significantly more parameters, 18 times more for SEEM and 15 times more for

Mask DINO (ResNet-50). Furthermore, adding three or four adapters has little impact on performance while doubling the number of additional parameters. However, fine-tuning the decoder only shows a significant decrease in the average delta δ, and is accompanied by a considerable increase in the number of parameters for both models. Fine-tuning only the embedding weights does not capture enough information about the downstream task, and results in an even larger reduction in the average delta (δ. This is exactly the situation where adapters and LoRA excel, as they are also placed in earlier layers, and as such can reduce error accumulation throughout the network.

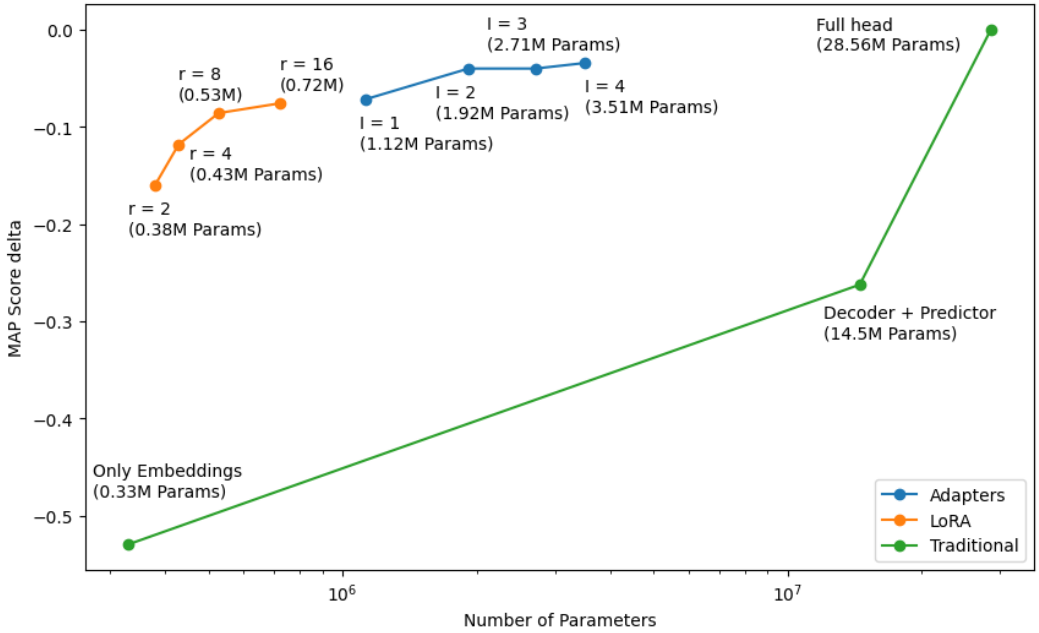


Figure 7. Delta δ comparison of Mask DINO results (does not include the Cityscapes results, as they were obtained with the larger Swin-L backbone).

To summarize, the previous results indicate that two to three adapters per transformer block provides fair and efficient transfer to instance segmentation tasks, offering the best tradeoff between performance and the number of trained parameters among the tested settings. In addition, it is worth mentioning that none of our experiments showed a converging trend, leaving potential for further improvement.

6.1.2. Analysis of LoRA Performance

This analysis aims to compare the performance of LoRA to that of full-head tuning and multiple adapter configurations based on the data provided in Table 2 and Figures 6 and 7. The analysis focuses on both performance metrics and parameter efficiency.

On the NDD20 dataset, adapters show superior performance compared to LoRA due to their higher number of parameters, which allows for more complex representation of water-specific visual characteristics. Moreover, the results in Table 2a show that increasing the rank of LoRA for SEEM results in a decline in performance, suggesting that these particular weight updates have a low intrinsic rank and that higher-rank adaptations may negatively impact the model's ability to generalize. Although LoRA outperforms the single-adapter configuration on this dataset, it does not achieve the results observed for both models with multiple-adapter configurations. For example, in SEEM, LoRA uses only $1.44M$ parameters, compared to $4.21M$ for the single-adapter setup, allowing it to nearly match the performance of configurations that use almost three times as many parameters. A similar analysis for Mask DINO shows that LoRA uses $0.53M$ parameters, while the single-adapter configuration uses $1.12M$. These results suggest that while LoRA is highly efficient, the additional parameters in adapters may capture more precise settings matched to the NDD20 task.

On the ZeroWaste dataset, the SEEM model with multiple-adapter configurations consistently outperforms the model with LoRA, demonstrating superior ability to handle the specific challenges of this dataset such as deformable objects with high variability, fine-grained distinctions between overlapping waste categories, and complex backgrounds with frequent occlusion. The increased parameter capacity of the adapters proves crucial for learning robust representations of these highly variable waste objects. Even the single-adapter configuration shows competitive performance, outperforming LoRA in most cases despite LoRA's parameter efficiency. This suggests that the ZeroWaste task benefits significantly from the ability of adapters to introduce task-specific layers. Similarly, for the Mask DINO model, all adapter configurations consistently outperform LoRA, suggesting that this architecture is particularly well suited to the adapter approach for the tasks presented in the ZeroWaste dataset. The superior performance of adapters in both models demonstrates their effectiveness in capturing the complex features and relationships unique to waste classification tasks.

The WIXray dataset produces mixed results, emphasizing the varying effectiveness of fine-tuning methods based on model architecture. In the SEEM model, LoRA outperforms the single-adapter configuration and competes with multiple-adapter setups despite using fewer parameters, indicating that LoRA's global updates can be effective for certain aspects of the WIXray task. However, in the Mask DINO model LoRA consistently underperforms compared to all adapter configurations, similar to its performance on the ZeroWaste dataset. This suggests that Mask DINO may be less compatible with LoRA's update mechanism. The challenges of the WIXray dataset, such as shifts from natural images to X-ray scans, complicate the segmentation task. While LoRA shows some effectiveness in SEEM, adapters generally show better performance, especially in Mask DINO. This is due to their ability to introduce specialized processing essential for extracting material-specific features from small overlapping objects in X-ray images. This context demonstrates the advantages of adapters, especially in architectures such as Mask DINO, while also illustrating the importance of the interaction between fine-tuning methods and model architectures in determining overall effectiveness for X-ray tasks.

The Cityscapes dataset illustrates remarkable results, especially for the Mask DINO model, where LoRA excels. In SEEM, LoRA's performance is similar to the other datasets, slightly outperforming the single-adapter setups but underperforming the multiple-adapter setups. However, in Mask DINO LoRA achieves the highest AP, outperforming all other configurations including full-head tuning, despite using only $0.54M$ parameters (0.24%) compared to $27.56M$ (12.37%) in full-head tuning. This success can be attributed to the fact that the Cityscapes dataset more closely matches common pretraining datasets that contain natural visible-light images of everyday scenes. Consequently, LoRA's efficiency in updating pretrained weights is sufficient for optimal performance, as minor adjustments to existing features are sufficient in this context. On the other hand, adapters may introduce additional complexity, making them less effective.

The parameter efficiency of LoRA makes it competitive in highly resource-constrained scenarios, as clearly shown in Figures 6 and 7, while adapters tend to scale better with additional layers, showing different scaling characteristics. However, performance gains are not always linear with increasing parameters. For SEEM, increasing the trainable parameters for LoRA faces limitations at a rank of 16, indicating that the affected layers have a lower intrinsic rank compared to Mask DINO. As a result, increasing the rank beyond 8 offers no significant benefit and even leads to a performance drop. From Figures 6 and 7, it can be concluded that the performance of LoRA quickly converges and that further increasing the rank may not lead to better results, which matches the observations from the original LoRA publication in the context of NLP [9]. In addition to these common limitations, our findings show that the effectiveness of fine-tuning methods varies depending on the task and model architecture.

6.2. Qualitative Results

To provide a more detailed visual representation of our results, one image from each dataset was tested three times with the Mask DINO inference script, using the best fine-tuned weights from each of full-head training, two-adapter training, and LoRA. The results provide a reasonable parameter–performance tradeoff across LoRA and the different adapter sizes. Figure 8 visualizes the ground truth (GT) along with the corresponding predictions of the two inference runs. A confidence threshold of 0.5 was implemented for the inference, effectively removing lower confidence predictions before visualization.

For the NDD20 dataset (first row), the predictions for all methods are highly accurate, almost achieving full intersection with the GT masks. The adapter method predicts the two masks with 1% lower confidence than the traditionally fine-tuned model. For the LoRA setting, the dolphin shapes are well-defined and slightly less accurate, which results in a small difference in the exact contours compared to the GT and the adapter method.

In the WIXray dataset (second row), the regions of interest mostly consist of bright contours on a white background. Although the adapters and LoRA successfully segment two out of three objects and correctly assign them to their respective classes, they have different levels of confidence, with LoRA showing the lowest confidence.

For the ZeroWaste dataset (third row), the selected image contains four GT object annotations. The full-head method detects one additional object compared to GT with higher confidence, while the adapters and LoRA each miss one object and have lower precision.

In the Cityscapes dataset (fourth row), the inference example follows the trend of almost exact object segmentation regardless of visual distance, occlusion, or the size of the instances. Again, the adapters and LoRA show lower confidence scores compared to the full-head fine-tuning, and both fail to detect the small car on the right.

These examples emphasize that employing the suggested adapter and LoRA fine-tuning method can indeed compete with the traditional fine-tuning approach in terms of pixel-wise object segmentation; however, further research is needed to fully compensate for the loss of trainable parameters and increase the confidence of the model for individual detections.

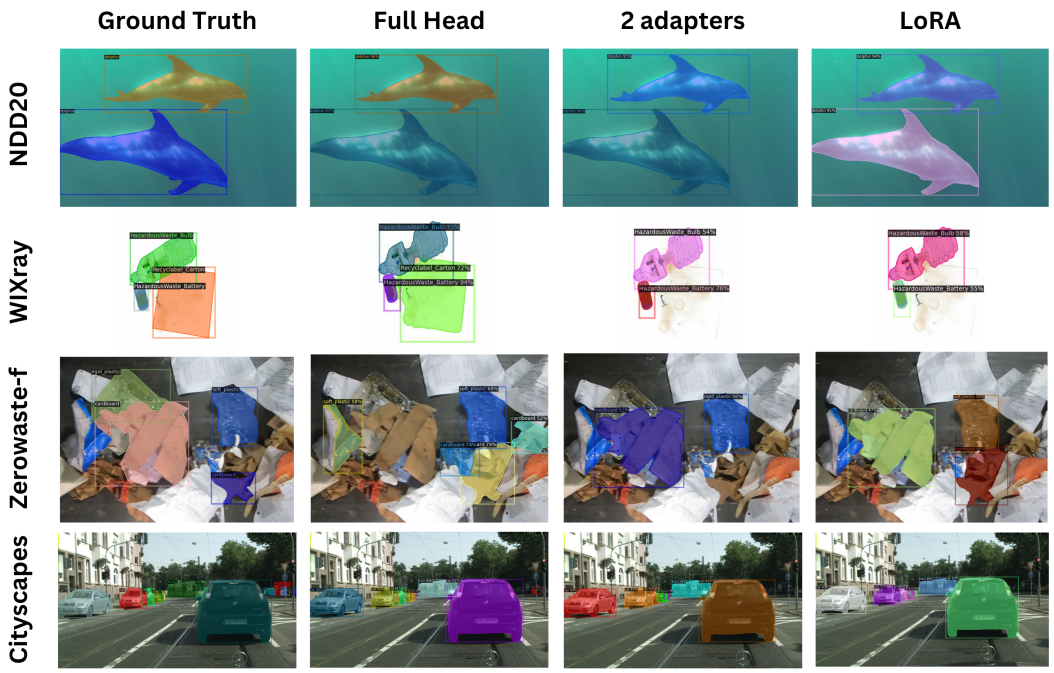


Figure 8. Four example visualizations (cropped and scaled) for Mask DINO, one from each dataset, showing the ground truth and the results of full-head, two-adapter, and LoRA fine-tuning.

6.3. Inference Speed

The inference times for the tested datasets and configurations are shown in Table 3 along with the image dimensions and numbers of classes. The results illustrate the impact of the PEFT methods on computational performance. The metric used for evaluation is *ms/iteration*, where each iteration used a batch size of 1. The increase in inference times when using adapters can be attributed to the computations introduced by the additional layers. As the number of adapters increases, more parameters are added to the model, which slightly increases the computational requirements and inference time. In contrast, LoRA generally imposes no computational overhead during inference, even with increasing rank, as the weights are recomposed and added to the base weights during initialization. This provides a clear advantage in terms of maintaining efficiency when fine-tuning large models.

For both models, the inference time varies across datasets; specifically, three factors influence this variation: variations in image resolution, the different numbers of classes in the datasets, and the different backbones used by the models. Datasets with higher-resolution images require more processing time, as do datasets with a larger number of classes. Although the open-set nature of SEEM ensures adaptability to different datasets, it does not reflect consistent processing times due to the variability of these characteristics.

In particular, the Cityscapes dataset shows a significant increase in inference time, especially when using Mask DINO. This is largely due to the Swin-L backbone used for Cityscapes, which is more computationally intensive than the ResNet50 backbone used for the other datasets, as well as the high-resolution image size of 1024×512. The opposite behavior can be observed for the WIXray dataset, where the original images are only 450×450.

On average, each increase of adapter repetition I was measured to introduce a delay of 1–2 milliseconds per iteration. Again, this result indicates the tradeoff between performance and computational complexity, achieving comparable results between LoRA and training the full segmentation head. It is important to note that these inference speeds are within the millisecond range, which may be negligible and unnoticeable during actual use.

Table 3. Image size, number of classes, and inference time (ms/iteration) of the baseline, adapter, and LoRA models across datasets.

Dataset	***NDD20***		***ZeroWaste***		WIXray		***Cityscapes***	
Image Size	512×512		512×512		450×450		1024×512	
# Classes	1		4		12		8	
Model	*SEEM*	*Mask DINO*	*SEEM*	*Mask DINO*	*SEEM*	*Mask DINO*	*SEEM*	*Mask DINO*
Model Baseline	65.43	52.19	85.06	64.59	54.5	40.2	135.69	267.36
1 Adapters ($\kappa + \lambda$)	66.58	54.92	87.02	66.7	55.63	42.89	136.73	271.25
2 Adapters ($\kappa + \lambda$)	67.52	56.38	88.16	67.57	56.91	44.03	137.18	275.31
3 Adapters ($\kappa + \lambda$)	68.32	58.29	88.28	69.07	57.52	46.08	137.94	283.92
4 Adapters ($\kappa + \lambda$)	70.36	61.12	88.79	71.11	58.52	48.22	138.43	298.11
LoRA (r=8)	65.76	52.21	85.28	64.85	54.11	40.61	135.86	266.34

7. Summary and Conclusions

This study investigated the effectiveness of sequentially repeated adapters and LoRA for instance segmentation tasks in CV. By applying these techniques to the SEEM and Mask DINO architectures and evaluating them on four different datasets (NDD20, ZeroWaste, WIXray, and Cityscapes), we have demonstrated their flexibility and scalability in transfer learning for large pretrained models.

Key findings and considerations include:

1. *Tradeoff between efficiency and performance*: While this study demonstrates significant parameter efficiency, it also reveals a consistent performance gap in certain scenarios.

The adapter configurations used only $1.23 - 3.88\%$ (SEEM) and $0.5 - 6.35\%$ (Mask DINO) of the total model parameters, compared to 39.55% (SEEM) and 54.91% (Mask DINO) for full-head tuning. LoRA demonstrated even greater efficiency, using just 0.42% as many parameters for SEEM and about 1% as many for Mask DINO with a ResNet-50 backbone. LoRA generally requires less computation and memory during training and inference due to its parameter-efficient design. Sequential alignment of adapters, despite being more parameter-intensive, can provide additional capacity, which is beneficial for complex tasks or significant domain shifts. This tradeoff between efficiency and performance raises important questions about the practical applicability of these methods in high-demand domains where even small drops in performance can be critical. Our results show that PEFT methods can achieve comparable performance to traditional full-head tuning while significantly reducing the number of trainable parameters.

2. *Optimal configurations*: Empirical evidence shows that between two and three adapter repetitions provides the best tradeoff between performance and parameter efficiency. Adding four adapters tends to result in diminishing returns in terms of performance improvement while significantly increasing the number of additional parameters. In contrast, LoRA demonstrates high parameter efficiency, often outperforming single-adapter configurations despite using fewer parameters.
3. *Scalability*: Both PEFT methods showed performance improvements with an increase in trainable parameters. More steady performance enhancement was observed for the adapters compared to LoRA, which is potentially attributable to the linear nature of sequentially aligned adapters. Scaling the number of layers instead of the bottleneck dimension or the rank in LoRA matrices may better capture the underlying representations and dependencies of certain downstream tasks.
4. *Dependence on dataset and architecture*: We found that the effectiveness of PEFT methods varied across datasets and model architectures, indicating that dataset complexity and architectural features play an important role in PEFT performance. For example, despite being six times smaller, Mask DINO achieved comparable or superior performance to SEEM in several experiments, likely due to its deformable attention-based detection branch and contrastive denoising approach. For the WIXray dataset, LoRA exhibited worse performance than adapters, suggesting that LoRA may be less effective on datasets where the images deviate significantly from the data used during pretraining.
5. *Extended applicability*: This study successfully applied PEFT techniques to the multiscale deformable attention module, thereby extending their applicability beyond standard transformer architectures. This opens up possibilities for application to other SOTA methods based on DETR.

These findings emphasize the importance of selecting PEFT methods based on model architecture and dataset characteristics. Furthermore, our empirical validation demonstrates the need to carefully consider the specific task requirements and computational constraints when choosing between LoRA and adapters.

There are several limitations to the study. First, there is some performance variability between tasks, with LoRA not performing as well on more complex datasets such as WIXray. This suggests that PEFT methods may not work equally well for every instance segmentation task. Another limitation is the increased inference time caused by the adapters due to the extra layers, which makes them less suitable for real-time applications. While LoRA shows promise in vision tasks, its potential has not yet been fully explored beyond the specific use cases discussed here, leaving many opportunities for broader application within vision models. In addition, the present study focuses on only two models, and does not fully explore how well these PEFT methods would generalize to other architectures or tasks such as object recognition or panoptic segmentation. Finally, the lack of detailed hyperparameter tuning in this paper means that several aspects which could be key to

further performance improvements, such as the learning rate and adapter placement, were not thoroughly investigated.

Future research could explore several directions to further optimize PEFT methods. A key area is to identify which tasks and model architectures are best suited to LoRA, potentially leading to guidelines on when it should be used versus adapters. In addition, hybrid fine-tuning strategies combining adapters and LoRA, such as by using adapters in higher layers and LoRA in attention layers, could be explored to further optimize efficiency. Although we successfully applied LoRA to the deformable attention mechanism in this work, several hyperparameters and configurations should be further investigated. Researchers could also develop new variants of adapters, such as using convolutional layers or testing different injection points in the network, which could improve performance and reduce early-stage errors. In addition, broader benchmarking on diverse sources such as medical imaging, multimodal learning, and different vision tasks such as object detection or panoptic segmentation could help to further test the adaptability and scalability of these methods. Overall, such efforts could pave the way for more effective and computationally efficient fine-tuning of large pretrained models.

In conclusion, this study provides empirical evidence supporting the feasibility of PEFT methods for the efficient adaptation of large pretrained models to instance segmentation tasks. By demonstrating competitive performance with significantly reduced parameter counts, these techniques offer promising solutions for resource-constrained environments and fast adaptation scenarios. As the field progresses, these results can contribute to ongoing efforts towards balancing computational efficiency and model performance in transfer learning for CV applications.

Author Contributions: Conceptualization, N.A.B. and D.R.; methodology, N.A.B. and D.R.; software, D.R.; validation, N.A.B., D.R., and U.H.; formal analysis, N.A.B. and D.R.; investigation, N.A.B. and D.R.; resources, N.A.B. and D.R.; data curation, D.R.; writing—original draft preparation, D.R.; writing—review and editing, D.R. and N.A.B.; visualization, D.R.; supervision, U.H.; project administration, U.H.; funding acquisition, U.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by the Ministry of Economy, Innovation, Digitization, and Energy of the State of North Rhine–Westphalia within the Digital.Zirkulär.Ruhr and Circular Performer Emscher Lippe projects.

Informed Consent Statement: Not applicable.

Data Availability Statement: The full implementation of this work has been published on Github: on access date: 3 November 2024 https://github.com/david-rohrschneider/SEEM-Adapter, https://github.com/david-rohrschneider/MaskDINO-Adapter. The final metrics and results are stored under Weights and Biases at: https://api.wandb.ai/links/david-rohrschneider/ly1kdwzr, https://api.wandb.ai/links/david-rohrschneider/rr18xj85.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
CE	Class Embedding
CNN	Convolutional Neural Network
CV	Computer Vision
DETR	DEtection TRansformer
DINO	DETR with Improved Denoising Anchor Boxes
EMA	Exponentially Moving Average
FPN	Feature Pyramid Network

GT	Ground Truth
mIoU	mean Intersection-over-Union
LN	Layer Normalization
LoRA	Low-Rank Adaptation
ME	Mask Embedding
MLP	Multi-Layer Perceptron
MRF	Materials Recovery Facilities
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
SAM	Segment Anything Model
SEEM	Segment Everything Everywhere Model
SOTA	State-Of-The-Art
ViT	Vision Transformer

Appendix A

The appendix presents Table A1, which evaluates four α values with constant $r = 8$. This comparison provides insights into the models' performance on the WIXray dataset under different scaling conditions.

Table A1. Evaluation of different scaling configurations α for LoRA.

WIXray Dataset	$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
AP	32.2	33.95	34.86	36.5

References

1. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment Anything. *arXiv* **2023**, arXiv:2304.02643.
2. Rohrschneider, D.; Abou Baker, N.; Handmann, U. Double Transfer Learning to Detect Lithium-Ion Batteries on X-Ray Images. In Proceedings of the 17th International Work-Conference on Artificial Neural Networks (IWANN), Ponta Delgada, Portugal, 19–21 June 2023; pp. 175–188.
3. Qiu, Y.; Jin, Y. ChatGPT and Finetuned BERT: A comparative Study for Developing Intelligent Design Dupport Systems. *Intell. Syst. Appl.* **2024**, *21*, 200308. [CrossRef]
4. Ebrahim, F.; Joy, M. Few-Shot Issue Report Classification with Adapters. In Proceedings of the International Workshop on NL-Based Software Engineering, Lisbon, Portugal, 20 April 2024; pp. 41–44.
5. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10–15 December 2024; pp. 10088–10115.
6. Zhang, R.; Han, J.; Liu, C.; Zhou, A.; Lu, P.; Qiao, Y.; Li, H.; Gao, P. LLaMA-Adapter: Efficient Fine-tuning of Large Language Models with Zero-initialized Attention. In Proceedings of the 12th International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
7. Stickland, A.C.; Berard, A.; Nikoulina, V. Multilingual Domain Adaptation for NMT: Decoupling Language and Domain Information with Adapters. In Proceedings of the 6th Conference on Machine Translation, Punta Cana, Dominican Republic, 10–11 November 2021; pp. 578–598.
8. Bapna, A.; Firat, O. Simple, Scalable Adaptation for Neural Machine Translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 1538–1548.
9. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the 10th International Conference on Learning Representations (ICLR), Online, 25–29 April 2022.
10. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Bangkok, Thailand, 1–6 August 2021; pp. 4582–4597.
11. Chen, G.; Liu, F.; Meng, Z.; Liang, S. Revisiting Parameter-Efficient Tuning: Are We Really There Yet? In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 2612—-2626.
12. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 3045–3059.

13. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. In Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 2790–2799.
14. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Online, 3–7 May 2021.
15. Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; Qiao, Y. Vision Transformer Adapter for Dense Predictions. In Proceedings of the 11th International Conference on Learning Representations (ICLR), Online, 1–5 May 2023.
16. Abou Baker, N.; Handmann, U. Don't Waste SAM. In Proceedings of the 31st European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 4–6 October 2023; pp. 429–434.
17. Chen, T.; Zhu, L.; Ding, C.; Cao, R.; Wang, Y.; Zhang, S.; Li, Z.; Sun, L.; Zang, Y.; Mao, P. SAM-Adapter: Adapting Segment Anything in Underperformed Scenes. In Proceedings of the International Conference on Computer Vision Workshops (ICCVW), Paris, France, 2–3 October 2023; pp. 3359–3367.
18. Wu, J.; Ji, W.; Liu, Y.; Fu, H.; Xu, M.; Xu, Y.; Jin, Y. Medical SAM Adapter: Adapting Segment Anything Model for Medical Image Segmentation. *arXiv* **2023**, arXiv:2304.12620.
19. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. In Proceedings of the 38th International Conference on Machine Learning (ICML), Online, 18–24 July 2021; pp. 8748–8763.
20. Gao, P.; Geng, S.; Zhang, R.; Ma, T.; Fang, R.; Zhang, Y.; Li, H.; Qiao, Y. CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *Int. J. Comput. Vis.* **2024**, *132*, 581–595. [CrossRef]
21. Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging Properties in Self-Supervised Vision Transformers. In Proceedings of the International Conference on Computer Vision (ICCV), Online, 11–17 October 2021; pp. 9650–9660.
22. Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.V.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. DINOv2: Learning Robust Visual Features without Supervision. *arXiv* **2024**, arXiv:2304.07193v2.
23. Zhang, B.; Chen, Y.; Bai, L.; Zhao, Y.; Sun, Y.; Yuan, Y.; Zhang, J.; Ren, H. Learning to Adapt Foundation Model DINOv2 for Capsule Endoscopy Diagnosis. *arXiv* **2024**, arXiv:2406.10508. [CrossRef]
24. Cui, B.; Islam, M.; Bai, L.; Ren, H. Surgical-DINO: Adapter Learning of Foundation Models for Depth Estimation in Endoscopic Surgery. *Int. J. Comput. Assist. Radiol. Surg.* **2024**, *19*, 1013–1020. [CrossRef] [PubMed]
25. Filatov, N.; Kindulov, M. Low Rank Adaptation for Stable Domain Adaptation of Vision Transformers. *Opt. Mem. Neural Netw.* **2023**, *32*, 277–283. [CrossRef]
26. Chavan, A.; Liu, Z.; Gupta, D.; Xing, E.; Shen, Z. One-for-All: Generalized LoRA for Parameter-Efficient Fine-tuning. *arXiv* **2023**, arXiv:2306.07967.
27. Chen, X.; Liu, J.; Wang, Y.; Wang, P.P.; Brand, M.; Wang, G.; Koike-Akino, T. SuperLoRA: Parameter-Efficient Unified Adaptation of Multi-Layer Attention Modules. *arXiv* **2024**, arXiv:2403.11887.
28. Leiñena, J.; Saiz, F.A.; Barandiaran, I. Latent Diffusion Models to Enhance the Performance of Visual Defect Segmentation Networks in Steel Surface Inspection. *Sensors* **2024**, *24*, 6016. [CrossRef] [PubMed]
29. Zou, X.; Yang, J.; Zhang, H.; Li, F.; Li, L.; Wang, J.; Wang, L.; Gao, J.; Lee, Y.J. Segment Everything Everywhere All at Once. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 10–15 December 2024; p. 868.
30. Li, F.; Zhang, H.; Xu, H.; Liu, S.; Zhang, L.; Ni, L.M.; Shum, H.Y. Mask DINO: Towards a Unified Transformer-Based Framework for Object Detection and Segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 3041–3050.
31. Huang, P.; Han, J.; Liu, N.; Ren, J.; Zhang, D. Scribble-Supervised Video Object Segmentation. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 339–353. [CrossRef]
32. Ravi, N.; Gabeur, V.; Hu, Y.T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; et al. SAM 2: Segment Anything in Images and Videos. *arXiv* **2024**, arXiv:2408.00714.
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June –1 July 2016; pp. 770–778.
34. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In Proceedings of the International Conference on Computer Vision (ICCV), Online, 11–17 October 2021; pp. 9992–10002.
35. Cheng, B.; Schwing, A.; Kirillov, A. Per-Pixel Classification is Not All You Need for Semantic Segmentation. In Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS), Online, 6–14 December 2021; pp. 17864–17875.
36. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Online, 3–7 May 2021.
37. Yang, J.; Li, C.; Dai, X.; Gao, J. Focal Modulation Networks. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 Novemeber–9 December 2022; pp. 4203–4217.

38. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
39. Cheng, B.; Misra, I.; Schwing, A.G.; Kirillov, A.; Girdhar, R. Masked-attention Mask Transformer for Universal Image Segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 1280–1289.
40. Lin, B. LoRA-Torch: PyTorch Reimplementation of LoRA. Available online: https://github.com/Baijiong-Lin/LoRA-Torch (accessed on 14 November 2024).
41. Trotter, C.; Atkinson, G.; Sharpe, M.; Richardson, K.; McGough, A.S.; Wright, N.; Burville, B.; Berggren, P. NDD20: A large-Scale Few-shot Dolphin Dataset for Coarse and Fine-grained Categorisation. *arXiv* **2020**, arXiv:2005.13359.
42. Bashkirova, D.; Abdelfattah, M.; Zhu, Z.; Akl, J.; Alladkani, F.; Hu, P.; Ablavsky, V.; Calli, B.; Bargal, S.A.; Saenko, K. ZeroWaste Dataset: Towards Deformable Object Segmentation in Cluttered Scenes. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 22–24 August 2022; pp. 21147–21157.
43. Qiu, L.; Xiong, Z.; Wang, X.; Liu, K.; Li, Y.; Chen, G.; Han, X.; Cui, S. ETHSeg: An Amodel Instance Segmentation Network and a Real-world Dataset for X-Ray Waste Inspection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 22–24 August 2022; pp. 2273–2282.
44. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 3213–3223.
45. Zhang, H.; Li, F.; Zou, X.; Liu, S.; Li, C.; Yang, J.; Zhang, L. A Simple Framework for Open-Vocabulary Segmentation and Detection. In Proceedings of the International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 1020–1031.
46. Jain, J.; Li, J.; Chiu, M.T.; Hassani, A.; Orlov, N.; Shi, H. OneFormer: One Transformer to Rule Universal Image Segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 2989–2998.