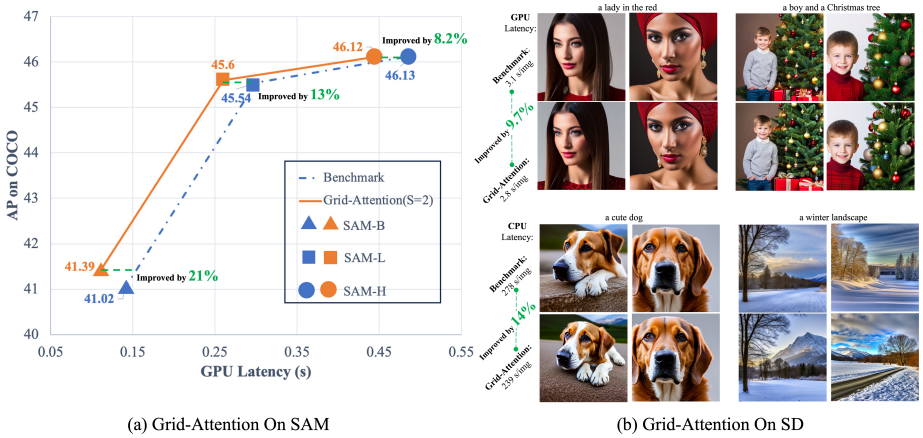
Grid-Attention: Enhancing Computational Efficiency of Large Vision Models without Fine-Tuning

Li Pengyu, Guo Tianchu, Wang Biao*, and Hua Xian-Sheng

Terminus Labs, Terminus Group, China.
lipengyu007@gmail.com,
antares_tcguo@163.com, wangbiao225@foxmail.com, huaxiansheng@gmail.com



(a) Grid-Attention On SAM (b) Grid-Attention On SD

Fig. 1: Without any fine-tuning or re-training, our Grid-Attention enhances the efficiency of large vision models while preserving their performance.

Abstract. Recently, transformer-based large vision models, *e.g.*, the Segment Anything Model (SAM) and Stable Diffusion (SD), have achieved remarkable success in the computer vision field. However, the quartic complexity within the transformer's Multi-Head Attention (MHA) leads to substantial computational costs in these models whose inputs and outputs are high-resolution. Although several prior works attempted to alleviate this challenge, none have successfully reduced the complexity and latency of large vision models while preserving their remarkable capabilities without requiring enormous efforts and GPU hours to re-train or fine-tune the models. To address the challenge, we propose a simple yet effective plug-and-play transformer block called Grid-Attention(GridAttn).

*Corresponding author

The GridAttn integrates the proposed Grid Clustering module, Grid Distributing strategies, and Grid Recovering module with common MHA to enhance the large vision models' computational efficiency and preserve their performance without the need for re-training or fine-tuning their parameters. We conduct extensive experiments on recent high-resolution tasks, including zero-shot instance segmentation (SAM, Expedit-SAM), text-to-image generation (Stable Diffusion V2.1), and semantic segmentation (SegFormer B0-B5). The experiments demonstrate that: Without any training or fine-tuning, GridAttn reduces GFlops by the range of [4.6%, 16.1%] and GPU inference latency by [8.2%, 21.4%], all while achieving equivalent performance (the performance bias ratio is less than 1%). Furthermore, the experiments present that GridAttn can also be trained from scratch or fine-tuned with very slight fine-tuning costs, resulting in a significantly improved performance-efficiency tradeoff. As a recommendation, we encourage the community to incorporate our GridAttn whenever deploying a well-trained transformer directly, fine-tuning a pre-trained one, or training a new one from scratch. The source code will be released in `https://github.com/pengyuLPY/GridAttn`.

1 Introduction

Since the transformer has been proposed [36], various vision transformers have been successfully applied to a broad range of vision applications, *e.g.* image classification [9, 10, 12, 20, 26, 27, 35, 41, 43], object detection [3, 7, 25, 46], segmentation [5,14,18,22,39,44,47], and image generation [16,31,32,34], surpassing CNN models and reaching state-of-the-art performance. In particular, the recent large vision transformer models, *e.g.* the Segment Anything Model (SAM)[1] [18] and Stable Diffusion (SD)[2] [31], present extraordinary performance on the segmentation and text-to-image generation tasks. However, it is costly to train and infer these transformer-based large vision models with high-resolution inputs and outputs, as the computational complexity of their stacked Multi-Head Attention (MHA) layers scales quadratically with the number of tokens, *i.e.*, scales quartic with the image size [29]. Consequently, efficiently running vision transformers is challenging amid such substantial computational requirements.

Several prior works attempted to alleviate this challenge. For instance, Yin *et al.* [41], Feichtenhofer *et al.* [10, 20], and Mehta *et al.* [28, 29] proposed efficient vision transformer architectures. He *et al.* [14] and Xie *et al.* [39] improved the efficiency of specific tasks, *e.g.* instance and semantic segmentation. However, all their proposals cannot inherit the parameters of well-trained models and necessitate re-training models almost from scratch. Re-training these large vision models like SAM[1] [18] and SD[2] [31] is challenging, costly, and time-consuming, although the new architecture may be more computationally efficient. Bolya *et al.* [2] and Liang *et al.* [23] merged the input tokens to reduce the computational complexity of MHA. Although their proposals do not require a re-training

[1] https://github.com/facebookresearch/segment-anything
[2] https://huggingface.co/stabilityai/stable-diffusion-2-1

phase, the lost valuable information in merged tokens leads to performance terrible drops, as Tab.2 presents. Overall, there is no previous work has improved the efficiency of large vision models while simultaneously maintaining their remarkable capabilities, without requiring re-training or fine-tuning the models.

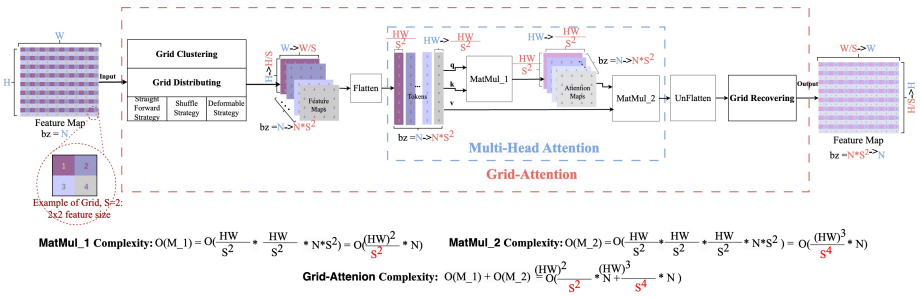


Fig. 2: Pipeline of Grid-Attention. bz is batch size, and S is a hyperparameter. For simplicity, we omit the operations that are irrelevant to our proposal in MHA.

We argue that a practical proposal for enhancing transformer efficiency should possess the following properties besides the efficiency itself. 1) Backward compatible with existing well-trained models. Given a well-trained model, the proposal can be employed directly with very few costs or is even cost-free. Thus, the community does not need to re-train the training-expensive large vision models. 2) Easy to train from scratch and easy to fine-tune from a pre-trained model. Therefore, the practitioners and researchers can employ the proposal in their consequent works whenever they aim to train a new transformer from scratch or fine-tune one from a pre-trained model. 3) Supporting the remarkable capabilities exhibited by large vision transformers. When applied to large models, it should reduce the computational complexity of models while preserving their excellent performance. 4) Friendly with mainstream transformer architectures and computer vision tasks.

In this paper, we are inspired by the claims that the images are natural signals with heavy spatial redundancy [4, 15] and propose a simple yet effective method possessing all four essential properties introduced above. Our proposal is called Grid-Attention (GridAttn), which integrates the proposed Grid Clustering (GC) module, Grid Distributing (GD) strategies, and Grid Recovering (GR) module with common MHA. GC clusters the spatially neighboring tokens within $S \times S$ into a grid and distributes intra-grid tokens into different sub-images with three GD strategies. The redundancy calculations between intra-grid tokens (*i.e.*, neighboring tokens) are reduced because of the distributions. MHA calculates the self-attention of inter-grid tokens and ignores the intra-grid ones. Then GR recovers distributed intra-tokens to their original locations. The pipeline of GridAttn is illustrated in Fig.2. By reducing redundant computations

within the intra-grid, GridAttn can reduce the computational complexity of the self-attention by S^2.

We conduct extensive experiments to prove the effectiveness of our GridAttn. Firstly, we quantitatively evaluate our GridAttn with the Segment Anything Model (SAM)[1] [18] and Expedit-SAM[3] [23]. The results in Tab.2 show that 1) without any fine-tuning, just employing our GridAttn into the image encoder of SAM or Expedit-SAM directly, we can achieve equivalent performance, 79% GPU latency, and 84% GFLOPs [30]. Moreover, we qualitatively evaluate GridAttn with Stable Diffusion V2.1 (SD)[2] [31] in Fig.1. The results show that GridAttn works well on SD besides the SAM and Expedit-SAM. It proves 2) our proposal supports the remarkable capabilities of mainstream large vision models and is friendly to these computer vision tasks. Finally, we conduct experiments with SegFormer [39], which provided abundant implementations ranging from real-time semantic segmentation to high-performance semantic segmentation. The experiments in Tab.3 and Tab.4 demonstrate that: 3) GridAttn can be fine-tuned with a slight fine-tuning cost, such as 10% or 5% of the original training iterations, and boosts the performance and efficiency simultaneously. 4) GridAttn can also be trained from scratch using the same implementations of the original transformer, achieving equivalent performance while only requiring 85% GPU latency and 73% GFLOPs.

GridAttn can be seamlessly plugged into workflows to significantly enhance the performance-efficiency tradeoff of transformers, whether practitioners and researchers aim to deploy a well-trained transformer directly, fine-tune a pre-trained one, or train a new one from scratch. It is worth noting that many existing techniques, *e.g.*, pruning [13, 40] and low-bit quantization [17, 21], are orthogonal to GridAttn and can be combined with GridAttn to improve their efficiency additionally, but these are not the scope of this paper.

2 Related Work

Benefiting from the dynamic modeling capability and long-range dependence of the attention mechanism, transformers [36] have achieved widespread success in various computer vision tasks, *e.g.*, image classification [9, 10, 12, 20, 26, 27, 35, 41, 43], object detection [3, 7, 25, 46], segmentation [5, 14, 18, 22, 39, 44, 47], and image generation [16,31,32,34]. In particular, some of the large vision transformers that trained with enormous efforts and GPU hours presented extraordinary performance. For example, Kirillov *et al.* [18] took 17,400 A100 GPU hours to train their Segment Anything Model (SAM)[1]. The SAM can segment any open-world target with the point, box, and text prompt. The Stable Diffusion (SD)[2] model is a text-to-image generation model inspired by the proposal of Rombach *et al.* [31], that can generate high-quality and high-resolution images based on the text prompt. The community took 200,000 A100 GPU hours to train it on the LAION-5B dataset [33]. Although these transformer-based models achieved

[3] https://github.com/Expedit-LargeScale-Vision-Transformer/Expedit-SAM

broad application, they took a mass of computational costs in their inference phases. Efficiently running vision transformers amid vast computational requirements is challenging but necessary.

Several prior paradigms attempted to alleviate this challenge. One paradigm proposed efficient vision transformer architectures [28, 29, 38, 41]. For example, Yin *et al*. [41] proposed A-ViT, which reformulated Adaptive Computation Time and extended halting to discard redundant spatial tokens. Mehta *et al*. [28, 29] proposed MobileViT and MobileViTv2. They presented a different perspective for the global processing of information with transformers and used element-wise operations for computing self-attention. However, most of the works in this paradigm focused on image classification and object detection tasks. Few of them have considered dense prediction tasks like segmentation or generation. Another paradigm focused on improving the efficiency of specific tasks [14, 39], *e.g*., instance and semantic segmentation. For instance, He *et al*. [14] presented FastInst, an effective query-based framework for real-time instance segmentation. Xie *et al*. [39] presented SegFormer, an efficient yet powerful semantic segmentation framework that unified a novel hierarchically structured transformer encoder and none-complex decoders. Although these works filled the gap in efficient dense prediction, they remain challenges when extending to other computer vision tasks and transformer architectures. Most importantly, all proposals in both above paradigms cannot inherit the parameters of well-trained models and necessitate re-training models almost from scratch. Even though their proposed architectures may be more efficient, re-training a large vision model, *e.g*., SAM [18] and SD [31], will take enormous efforts and hundreds of thousands of A100 GPU hours, which is very costly. We think these limitations are the reasons why ViT [9] and Swin transformers [27] with common MHA still dominate the community.

The third paradigm is the training-free token-merging paradigm [2,23], which is the one most related to ours. Bolya *et al*. [2] proposed ToMe, which combines similar tokens in a transformer. The number of tokens in ToMe is reduced irreversibly, which blocks their usage in dense prediction tasks like segmentation and image generation. Liang *et al*. [23] imported the token merging approaches into segmentation tasks. However, no evidence proves their proposal can work in large vision models like SAM and SD. In fact, we find importing their proposal to SAM (*i.e*., Expedit-SAM^3) leads to terrible performance drops, as Tab.2 shows. For example, their proposal drops the AP of COCO [24] from 46.1 to 42.3.

Overall, the previous works do not possess sufficient capacity to support the remarkable capabilities exhibited by large vision transformers and reduce their inference complexity simultaneously. The challenge is far from being solved.

3 Method: Grid-Attention

3.1 Observation and Inspiration

He *et al*. [15] and Chen *et al*. [4] claimed that the images are natural signals with heavy spatial redundancy. We conduct a lower-boundary experiment to analyze

how performance would drop when redundancy between all spatial neighboring pixels is eliminated via toughly disconnecting them. Specifically, we sequentially sample one pixel in every 2×2 neighbor to create four sub-images. Every sub-images can be respectively predicted segmentation map by a frozen SAM. Finally, we aggregate these outputs to obtain the segmentation result of the original whole image. The pipeline of lower-boundary is illustrated in the Supplementary Materials. Its results are shown in Tab.1. For easy comparison, we also early report the performance of our GridAttn in this table.

Frozen Model		AP	AP^S	AP^M	AP^L
SAM-B	Benchmark	41.02	28.33	45.59	53.41
	Lower-Boundary	36.50	21.73	42.03	51.85
	Ours: GridAttn	41.39	27.98	45.79	54.90
SAM-L	Benchmark	45.54	30.21	50.12	60.44
	Lower-Boundary	40.53	22.95	46.03	58.45
	Ours: GridAttn	45.59	29.92	50.07	60.73
SAM-H	Benchmark	46.13	30.70	50.58	61.15
	Lower-Boundary	41.51	23.39	47.02	59.77
	Ours: GridAttn	46.12	30.43	50.60	61.12

Table 1: Experiment about the lower boundary. It follows evaluation settings in SAM [18] on COCO instance segmentation [24]. Notable, we establish the Benchmark of SAM via the released codes and models. The SAM-H benchmark we achieved is 46.13, which differs from the 46.5 reported in their paper. To ensure fair comparisons, we utilize the benchmarks we obtained for SAM-H, SAM-L, and SAM-B.

Based on Tab.1, we observed that eliminating the spatial redundancy with the naive and rough lower-boundary method still achieves reasonable performance by a frozen large vision model. The observation inspires us to elaborately design a training-free method to improve the computational efficiency of large vision models while preserving their performance by eliminating the redundancy.

3.2 Details of Grid-Attention

We implement our Grid-Attention (GridAttn) with the proposed Grid Clustering (GC) module, Grid Distributing (GD) strategies, Grid Recovering (GR) module, and common Multi-Head Attention (MHA). The pipeline is illustrated in Fig.2.

Grid Clustering (GC) module. GC clusters the spatially neighboring features within $S \times S$ into a grid, as indicated by the red circle in Fig.2. The intra-grid pixels (such as all four features in the red circle) are distributed to different sub-images with Grid Distributing strategies (Sec.3.3) while combining inter-grid pixels into the same ones. For example, all pixels marked $'1'$ in Fig.2 constitute a sub-image, while those marked $'2'$ constitute another. This process reduces the resolution by $S \times S$ times and increases the batch size by S^2. Thus, the input size is transformed from $[N, C, H, W]$ to $[N \cdot S^2, C, H/S, W/S]$, where

N is the original batch, H and W are the height and width of original feature maps, C is the number of channels, and S is the hyper-parameter manually set to balance the performance and efficiency.

Multi-Head Attention (MHA). MHA is calculated as common. Its parameters can be loaded directly from a well-trained transformer. Because of the prior Grid Clustering module, the input size here is $[N \cdot S^2, (HW)/S^2, C]$ instead of the original $[N, HW, C]$. We provide the complexity of the self-attention in common MHA and GridAttn in Eq.1 and Eq.2. The equations exclude the variable C for brevity, and their derivation procedure is detailed in Fig.2. The equations present that the GridAttn reduces the complexity by a factor of S^2.

$$\begin{aligned} O(MHA) &= O(MatMul_1) + O(MatMul_2) \\ &= O(N \cdot (HW)^2) + O(N \cdot (HW)^3) \end{aligned} \tag{1}$$

$$O(GridAttn) = O(N \cdot (HW)^2/S^2) + O(N \cdot (HW)^3/S^4) \tag{2}$$

Grid Recovering (GR) module. GR is the inverse of the Grid Clustering module. GR gathers the intra-grid outputs of the MHA across the S^2 sub-images and aggregates them to compose the final output of the GridAttn. The intra-grid features, which were distributed to different sub-images in the prior GC module, are recovered to their original location within the grid. Via GR, the output size of GridAttn is recovered to the same as its original input size, *i.e.*, $[N, C, H, W]$.

Algorithm.1 presents a pseudo code of GridAttn. It is worth noting that GC, GD, and GR only involve parameter-free operations, *i.e.*, reshaping and permuting. The parameters in MHA can be loaded from the well-trained transformer models and then frozen. Therefore, GridAttn can be directly employed to a well-trained transformer without the need for fine-tuning or re-training.

3.3 Grid Distributing Strategies

Line.3-Algorithm.1 presents a Straightforward Strategy for distributing intra-grid tokens. Tab.2 shows this strategy works well when GridAttn is directly employed in the inference/deploy phase. However, when trained from scratch, its performance is inferior to the original transformers, as Tab.4 presents. It may be because the strategy only calculates self-attention with checkerboard patterns (*i.e.*, the same color area in the feature maps shown in Fig.2). This causes too much neighboring information to be lost in the training phase. The issue has been observed and defined as the *gridding issue* in the dilated convolution layer [37,42].

To tackle the issue, we propose two additional Grid Distributing strategies, namely the Shuffle Strategy and the Deformable Strategy, alongside the existing Straightforward Strategy, making a total of three GD strategies in our GridAttn.

Straightforward Strategy. This strategy distributes the intra-grid pixels through a reshaping operation. It is easy to implement and performs well but suffers from the *gridding issue* when trained from scratch.

Shuffle Strategy. In the training phase, the strategy shuffles the intra-grid features within $S \times S$ before distributing them via reshaping operation. This strategy enables the neighboring information to be randomly exchanged across

Algorithm 1: Pseudo code of Grid-Attention

Input : A feature map I_{in} of size $[N, C_{in}, H, W]$
Output: A feature map I_{out} of size $[N, C_{out}, H, W]$

// Grid Clustering Module
$I_{in} \leftarrow$ `Reshape` $(I_{in}, [N, C_{in}, H/S, S, W/S, S])$;
`// Straightforward Grid Distributing Strategy`
$I_{in} \leftarrow$ `Permute` $(I_{in}, [N, S, S, H/S, W/S, C_{in}])$;
$I_g \leftarrow$ `Reshape` $(I_{in}, [NS^2, H/S, W/S, C_{in}])$;

// MHA Module
$N_g, L_g = NS^2, (HW)/S^2$;
$I_g \leftarrow$ `Flatten` $(I_g, [N_g, L_g, C_{in}])$;
$\mathsf{q} = \mathsf{k} = \mathsf{v} = I_g$;
`// size of` I_{out} `is` $[N_g, L_g, C_{out}]$
$I_{out} \leftarrow$ `Multi-Head-Attention` (q, k, v);
$I_{out} \leftarrow$ `UnFlatten` $(I_{out}, [NS^2, H/S, W/S, C_{out}])$;

// Grid Recovering Module Corporating Straightforward Strategy
$I_{out} \leftarrow$ `Reshape` $(I_{out}, [N, S, S, H/S, W/S, C_{out}])$;
$I_{out} \leftarrow$ `Permute` $(I_{out}, [N, C_{out}, H/S, S, W/S, S])$;
$I_{out} \leftarrow$ `Reshape` $(I_{out}, [N, C_{out}, H, W])$;

MHA layers in the training phase. In the inference phase, the strategy works in the same way as the Straightforward Strategy. It is easy to implement the GR in this strategy because the indexes of intra-grid features are cached in prior GC.

Deformable Strategy. Dai *et al*. [6] introduced a method for estimating the offsets from the preceding feature maps. Our Deformable Strategy follows their proposal, enabling form deformation of the sampling intra-grid features freely. For easy implementation, this strategy utilizes two auxiliary 3×3 convolutional layers. One convolutional layer estimates offsets in GC, and the other estimates recovery-offsets in GR. To ensure that predicted offsets are initialized as zero, we initialize the parameters of the convolutional layers accordingly. As a result, the Deformable Strategy is equivalent to the Straightforward Strategy in its initial iteration but adaptively optimized in the following training phase.

As introduced above, all three proposed Grid-Distributing strategies can be easily employed in GridAttn. Their benefits are carefully analyzed in Sec.4.4.

4 Experiments

Sec.4.1 details our implementation first.

In Sec.4.2, we employ our GridAttn to two mainstream large vision models that the community took enormous effort to establish, *i.e*., the Segment Anything Model (SAM)[1] [18] and the Stable Diffusion V2.1[2] (SD) [31]. Additionally, we conduct experiments with Expedit-SAM[3] [23], which is the state-of-the-art training-free efficiency-improved method. The experiments present that the proposed GridAttn reduces GFLOPs, accelerates inference speed, and preserves the

performance of mainstream large vision models, although there is NO fine-tuning or re-training phase. It is a state-of-the-art, cost-free enhancement approach.

SegFormer [39] is easy to train/fine-tune and provides abundant implementations from real-time to high-performance semantic segmentation. Therefore, we implement GridAttn on it and fine-tune the model with 10% of its original training iterations in Sec.4.3. It demonstrates that GridAttn can boost performance and efficiency simultaneously with a slight fine-tuning cost.

In Sec.4.4, we train the SegFormer [39] from scratch. The results show that GridAttn can be trained from scratch with the original implementations, achieving equivalent performance but with reduced computational costs.

The ablation studies are presented in Sec.4.5.

4.1 Implementation Details

Implementation on SAM and Expedit-SAM. We replace all global self-attention MHA layers in SAM and Expedit-SAM with GridAttn. The parameters are loaded from the original models and frozen. To analyze the effectiveness of GridAttn quantitatively, We follow SAM's evaluation settings on the **COCO** instance segmentation task [24]. Specifically, SAM and Expedit-SAM are prompted with ViTDet [19] boxes to do zero-shot instance segmentation and evaluate the performance with Average Precision (AP). The input resolution of 1024×1024 is obtained by rescaling the image and padding the shorter side.

The same experiments on the **LVIS** dataset [11] are provided in Supplementary Materials. The results on LVIS confirm the conclusions obtained on COCO.

Implementation on SD. We substitute all global self-attention MHA layers in DownBlocks and MidBlocks of SD V2.1 with our GridAttn. The configuration details are provided in Supplementary Materials. We visualize the generated images to assess the GridAttn performance on SD qualitatively. The output resolution is 768×768.

Implementation on SegFormer. We follow the original implementations of SegFormer [39] on the **ADE20K** dataset [45] and report semantic segmentation performance using mean Intersection over Union (Multi-Scale/Single-Scale) *i.e.*, mIoU (MS/SS). Notably, we double the training batch size as we encountered difficulty reproducing the performance with the original size. Unless specifically stated, all the experiments in this paper are with the doubled batch size. Additional experiments with the original batch size can be found in the Supplementary Material. When training from scratch, we do not pre-train models with the ImageNet dataset [8] but directly load the ImageNet pre-trained SegFormer parameters. For fine-tuning, we initialize the GridAttn models with the parameters of the original well-trained SegFormer models and fine-tune them with an initial learning rate of 1e-5.

Experiments are conducted with Pytorch [1] on eight NVIDIA A100 GPUs.

Frozen Model	Expedit [23]	ours: GridAttn	GFLOPs	Latency(s/img) GPU	Latency(s/img) CPU	Performance AP	Performance AP^S	Performance AP^M	Performance AP^L
SAM-B			969.6	0.14	2.01	41.02	28.33	45.59	53.41
		✓	**813.8**	**0.11**	**1.31**	41.39	27.98	45.79	54.90
	✓		888.8	0.12	1.69	39.73	25.91	44.65	52.96
	✓	✓	**750.4**	**0.09**	**1.11**	39.89	25.71	44.31	54.19
SAM-L			2980	0.30	3.92	45.54	30.21	50.12	60.44
		✓	**2772**	**0.26**	**3.16**	45.59	29.92	50.07	60.73
	✓		2588	0.25	3.17	42.94	25.59	46.72	60.85
	✓	✓	**2415**	**0.22**	**2.61**	43.04	25.95	46.86	60.46
SAM-H			5952	0.49	5.95	46.13	30.70	50.58	61.15
		✓	**5692**	**0.45**	**4.95**	46.12	30.43	50.60	61.12
	✓		5094	0.43	4.84	42.33	24.55	46.06	60.43
	✓	✓	**4893**	**0.39**	**4.22**	42.42	24.58	46.08	60.58

Table 2: Employing GridAttn to SAM and Expedit-SAM(*hourglass_num_cluster*=144) without any fine-tuning or re-training. GridAttn maintains its AP performance while improving its efficiency significantly. **For SAM**, GridAttn improves its AP by the ratio of $[-0.02\%, 0.9\%]$ (*i.e.*, [SAM-H, SAM-B]), reduces GFLOPs between $[4.6\%, 16.1\%]$, reduces GPU latency between $[8.2\%, 21.4\%]$. **For Expedit-SAM**, they are $[0.2\%, 0.4\%]$, $[3.9\%, 15.6\%]$, $[6.4\%, 18.1\%]$.

4.2 Without Fine-Tuning or Re-Training

In this section, we conduct experiments on SAM [18], Expedit-SAM [23], and SD [31] without any fine-tuning or re-training phase to demonstrate our GridAttenion is a cost-free method to improve the inference efficiency while preserving the performance of existing large vision models. Since Expedit-SAM represents state-of-the-art training-free efficiency improvements for segmentation tasks, the experiments in this section further demonstrate the superiority of our proposed GridAttn. The hyper-parameter S is set to 2. The quantitative experiments on SAM and Expedit-SAM are presented in Tab.2.

Tab.2 demonstrates that:

1) GridAttn enhances SAMs' efficiency while keeping their performance almost constant. For example, the AP performance with *vs.* without GridAttn is 41.39 *vs.* 41.02 on SAM-B, meaning GridAttn occasionally boosts the AP performance. The GPU latencies are reduced by 21.4%. The GFLOPs are reduced by 16.1%

2) Although Expedit-SAM [23] improves SAM efficiency, the AP performance presents terrible drops (from 46.1 dropped to 42.3 in SAM-H). The performance drops block it from achieving good efficiency-performance tradeoffs without fine-tuning. Furthermore, we seamlessly integrate our GridAttn with Expedit-SAM and observe that GridAttn can further improve the efficiency of Expedit-SAM while preserving its performance. These experiments not only present the superiority of our GridAttn over the previous state-of-the-art, *i.e.*, Expedit-SAM, but also demonstrate that GridAttn can be integrated into previous works

to enhance their efficiency further. Additional experiments with various *hourglass_num_cluster* settings in the Supplementary Material also support these conclusions.

3) We noticed that GridAttn improves SAM-B more than SAM-H. We hypothesize part of the reason is that the *embed_dim* of SAM-H is much larger than SAM-B (1280 vs. 768). A large *embed_dim* consumes a high proportion of computational resources, which may limit the improvement of GridAttn.

Furthermore, the generated images of SD with and without GridAttn are visualized in Fig.1. More visualizations are presented in Supplementary Material because of paper length limitations. This experiment further supports that GridAttn can significantly enhance the efficiency of SD, besides SAM, cost-freely. It has enough capacity to support the remarkable performance of mainstream large vision models like SAM and SD while enhancing their efficiency.

4.3 Fine-Tuning

In this section, we employ our GridAttn to SegFormer B0-B5 [39] to present that GridAttn can improve efficiency and performance simultaneously by taking a slight cost, *e.g.*, 10% of the original training iterations, to fine-tune it. We use the Shuffle Strategy and set the hyper-parameter S to be 2 in this section. The results are shown in Tab.3. The additional experiments implemented with other GD strategies are detailed in the Supplementary Materials.

mIoU (MS/SS) Performance						
	B5	B4	B3	B2	B1	B0
Benchmark*	51.8/51.0	51.1/50.3	50.0/49.4	47.5/46.5	43.1/42.2	38.0/37.4
Benchmark	51.4/**50.9**	50.6/49.9	**50.4**/**49.5**	**48.2**/**47.0**	43.1/42.2	**39.1**/**38.5**
Naive_FT	51.4/50.7	51.0/50.2	50.4/49.3	47.8/46.9	43.9/43.0	38.9/38.3
GridAttn	**51.5**/**50.9**	**51.3**/**50.4**	**50.4**/**49.5**	48.0/**47.0**	**44.1**/**43.1**	**39.1**/**38.5**

Table 3: Fine-tuning SegFormer with GridAttn. It takes 10% of the original training iterations, *i.e.*, 16000 iterations. Benchmark* reports the performance in paper [39] with $bz = 2/GPU$. Naive_FT refers to fine-tuning the original SegFormer without incorporating GridAttn.

The results present that the improvements, both in performance (Tab.3) and efficiency (Tab.4), hold across various transformers, including real-time transformers like SegFormer-B0 and B1, balanced transformers like SegFormer-B2 and B3, and high-performance transformers like SegFormer-B4 and B5.

4.4 Training from scratch

We use three proposed Grid Distributing Strategies to train SegFormer B0-B5 [39]. We directly import the original SegFormer configures but double the batch size as mentioned in Sec.4.1. The results are reported in Tab.4 and Fig.3.

GFLOPs							
		B5	B4	B3	B2	B1	B0
Benchmark		423.7	339.8	241.7	150.5	89.1	46.5
GridAttn	Straightforward	**301.2**	**237.1**	**170.3**	**103.4**	**63.9**	**33.8**
	Shuffle						
	Deformable	305.4	240.7	172.8	105.1	64.8	34.3
GPU/CPU Latency (ms per img)							
		B5	B4	B3	B2	B1	B0
Benchmark		56.2/1476	43.5/651	32.2/409	20.1/361	11.1/195	10.1/115
GridAttn	Straightforward	**47.9/923**	**37.0/456**	**26.8/318**	**16.9/298**	**10.6/170**	**10.0/98.0**
	Shuffle						
	Deformable	240/1191	184.2/730	133.1/585	78.4/430	41.4/317	40.3/270
mIoU (MS/SS) Performance							
		B5	B4	B3	B2	B1	B0
Benchmark		51.4/50.9	50.6/**49.9**	50.4/49.5	48.2/47.0	43.1/42.2	39.1/38.5
GridAttn	Straightforward	51.1/50.5	49.1/48.6	49.4/48.5	47.3/46.4	42.5/41.7	38.9/38.0
	Shuffle	51.8/51.4	**50.7**/**49.9**	50.2/48.9	48.0/46.8	42.9/42.2	39.0/38.6
	Deformable	**52.0**/**51.6**	50.6/49.8	**50.5**/**49.6**	**48.5**/**47.1**	**43.5**/**42.5**	**40.0**/**39.1**

Table 4: Training SegFormer with GridAttn from scratch, using the proposed three Grid Distributing Strategies. The Straightforward strategy and the Shuffle strategy share the same inference phase, thus sharing the same GFLOPs and latency. The efficiency is tested with the input resolution of 2048×512.

It can be observed that:

1) Although the performance of GridAttn using the Straightforward Strategy is inferior to the Benchmark when it is trained from scratch, Fig.3 presents it achieves a superior performance-efficiency tradeoff to the Benchmark. We attribute the inferiority of the Straightforward Strategy to the gridding issue introduced in Sec.3.3. To alleviate the issue, we propose the Shuffle Strategy and Deformable Strategy. It is worth noting that the Straightforward Strategy works well when GridAttn is directly plugged into the frozen SAM, as Tab.2 shows.

2) The inference phase of the Shuffle Strategy is the same as the one of the Straightforward Strategy, as Sec.3.3 introduced. Therefore, these two strategies share the same efficiency results in Tab.4. The table concludes that the Shuffle Strategy achieves the equivalent mIoU performance to the *Benchmark* while significantly improving its efficiency.

3) The Deformable Strategy presents significant superiority in mIoU performance and GLOPs efficiency. For instance, the SegFormer-B5 with Deformable Strategy has 305.4 GFLOPs and 52.0 mIoU (MS). It significantly boosts the Benchmark that has 423.7 GLOPs and 51.4 mIoU (MS). However, the Deformable Strategy incurs high latency due to slow operations like *index-select*. For example, SegFormer-B5 with Deformable Strategy requires 240 GPU milliseconds (details in the Supplementary Materials) to process a single 2048×512

image, whereas Benchmark takes 56.2 milliseconds. Therefore, the Deformable Strategy is efficient for GFLOPs but not for latency.

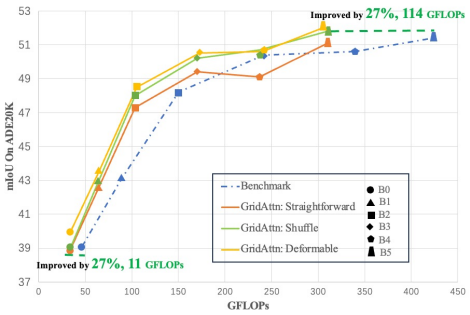


Fig. 3: Performance-Efficiency curve of Segformer with GridAttn trained from scratch.

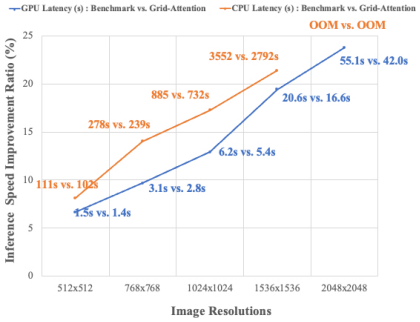


Fig. 4: Ablation studies about the impact of image resolution via SD. $S = 2$.

Based on the aforementioned observations, it can be concluded that GridAttn effectively improves the Benchmark, even though trained from scratch. Considering factors such as GFLOPs, latency, and performance, we recommend the Shuffle Strategy.

4.5 Ablation Studies

In this section, we design experiments to analyze the impact of image resolutions on efficiency improvement. Besides, we analyze how hyper-parameters S and fine-tuning iterations affect the performance-efficiency tradeoff of our GridAttn.

Changing the output image resolution of SD is easy because there is no learnable position embedding. Therefore, we analyze the impact of image resolution via SD. In Fig.4, it is evident that our enhancement becomes more pronounced as the image size increases.

Fig.5 illustrates the effect of hyper-parameter S. The performance-efficiency curves show that with the S increasing, the efficiency of SAM improves, but the performance drops. When $S \leq 4$, GridAttn enhances the Benchmark performance-efficiency tradeoff. The optimal tradeoff is achieved when $S = 2$ (orange line in Fig.5). When $S = 8$ (yellow line in the figure), GridAttn performs poorly if it is not fine-tuned or re-trained. We think it is because employing GridAttn with a large S (*e.g.* 8) to a frozen SAM may make its feature maps too sparse to capture valuable nearby information.

Tab.5 presents the effect of fine-tuning iterations on the performance. The results show that our GridAttn exceeds the Benchmark, and its performance is not sensitive to the number of fine-tuning iterations when the number is higher than 8000 iterations, which is only 5% of the number required by training from scratch. The performance slightly drops when the number decreases to 1600 (*i.e.*,

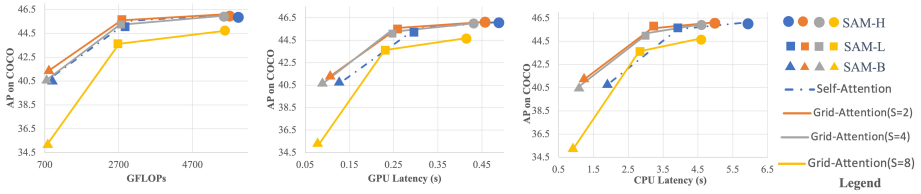


Fig. 5: Ablation Studies about effect of grid stride (S) via SAM.

1% of training from scratch), which may be because a short training phase with an initial lr=1e-5 leads the transformer doesn't fully converge.

mIoU (MS/SS) Performance							
	FT Iterations	B5	B4	B3	B2	B1	B0
Benchmark		51.4/50.9	50.6/49.9	50.4/49.5	48.2/47.0	43.1/42.2	39.1/38.5
GridAttn	32000 (20%)	51.7/50.9	51.1/50.2	50.5/49.5	48.3/47.0	43.7/42.9	39.1/38.5
	16000 (10%)	51.5/50.9	51.3/50.4	50.4/49.5	48.0/47.0	44.1/43.1	39.1/38.5
	8000 (5%)	51.5/50.5	51.3/50.2	50.4/49.3	47.4/46.6	43.9/42.9	39.0/38.2
	1600 (1%)	51.3/50.4	50.9/49.9	49.5/48.7	47.3/46.1	43.5/42.4	38.9/37.9

Table 5: Ablation studies about fine-tuning iteration number via SegFormer. $S = 2$.

5 Conclusion

This paper proposes a simple yet effective plug-and-play transformer block named Grid-Attention. Without any re-training or fine-tuning, Grid-Attention can enhance the efficiency of transformers while preserving their performance. Besides, Grid-Attention can also be trained from scratch or take a slight fine-tuning cost to boost performance and efficiency simultaneously. The experiments on SAM, Expedit-SAM, SD, and SegFormer consistently demonstrate that Grid-Attention outperforms the original transformers and state-of-the-art training-free efficiency improvement approach. As a recommendation, we encourage the community to incorporate our Grid-Attention whenever deploying a well-trained transformer directly, fine-tuning a pre-trained one, or training a new one from scratch.

Acknowledgments

This work is supported by the 2024 Innovative Talent Funding Program for Science and Technology in Tongzhou District, Beijing (Grant No. JCQN2024004).

References

1. Pytorch (2016), `https://pytorch.org/` Last accessed on 2023-09-15
2. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461 (2022)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
4. Chen, Y., Fan, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., Yan, S., Feng, J.: Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3435–3444 (2019)
5. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. Advances in Neural Information Processing Systems **34**, 17864–17875 (2021)
6. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
7. Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., Zhang, L.: Dynamic detr: End-to-end object detection with dynamic attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2988–2997 (2021)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. 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. arXiv preprint arXiv:2010.11929 (2020)
10. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6824–6835 (2021)
11. Gupta, A., Dollar, P., Girshick, R.: Lvis: A dataset for large vocabulary instance segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5356–5364 (2019)
12. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. Advances in Neural Information Processing Systems **34**, 15908–15919 (2021)
13. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
14. He, J., Li, P., Geng, Y., Xie, X.: Fastinst: A simple query-based model for real-time instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 23663–23672 (2023)
15. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
16. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
17. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2704–2713 (2018)

18. 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 preprint arXiv:2304.02643 (2023)
19. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. In: European Conference on Computer Vision. pp. 280–296. Springer (2022)
20. Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: Mvitv2: Improved multiscale vision transformers for classification and detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4804–4814 (2022)
21. Li, Y., Xu, S., Zhang, B., Cao, X., Gao, P., Guo, G.: Q-vit: Accurate and fully quantized low-bit vision transformer. Advances in Neural Information Processing Systems **35**, 34451–34463 (2022)
22. Li, Z., Wang, W., Xie, E., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P., Lu, T.: Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1280–1289 (2022)
23. Liang, W., Yuan, Y., Ding, H., Luo, X., Lin, W., Jia, D., Zhang, Z., Zhang, C., Hu, H.: Expediting large-scale vision transformer for dense prediction without fine-tuning. Advances in Neural Information Processing Systems **35**, 35462–35477 (2022)
24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
25. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023)
26. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12009–12019 (2022)
27. 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 IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
28. Mehta, S., Rastegari, M.: Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178 (2021)
29. Mehta, S., Rastegari, M.: Separable self-attention for mobile vision transformers. arXiv preprint arXiv:2206.02680 (2022)
30. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016)
31. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
32. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22500–22510 (2023)

33. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. Advances in Neural Information Processing Systems **35**, 25278–25294 (2022)
34. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
35. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
37. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. In: 2018 IEEE winter conference on applications of computer vision (WACV). pp. 1451–1460. Ieee (2018)
38. Wang, Y., Huang, R., Song, S., Huang, Z., Huang, G.: Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. Advances in Neural Information Processing Systems **34**, 11960–11973 (2021)
39. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. Advances in Neural Information Processing Systems **34**, 12077–12090 (2021)
40. Yang, H., Yin, H., Shen, M., Molchanov, P., Li, H., Kautz, J.: Global vision transformer pruning with hessian-aware saliency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18547–18557 (2023)
41. Yin, H., Vahdat, A., Alvarez, J.M., Mallya, A., Kautz, J., Molchanov, P.: A-vit: Adaptive tokens for efficient vision transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10809–10818 (2022)
42. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
43. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 558–567 (2021)
44. Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., Wang, J.: Fast segment anything. arXiv preprint arXiv:2306.12156 (2023)
45. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 633–641 (2017)
46. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
47. Zou, X., Yang, J., Zhang, H., Li, F., Li, L., Gao, J., Lee, Y.J.: Segment everything everywhere all at once. arXiv preprint arXiv:2304.06718 (2023)