# HEATDETR: HARDWARE-EFFICIENT DETR WITH DEVICE-ADAPTIVE THINNING

**Anonymous authors**
Paper under double-blind review

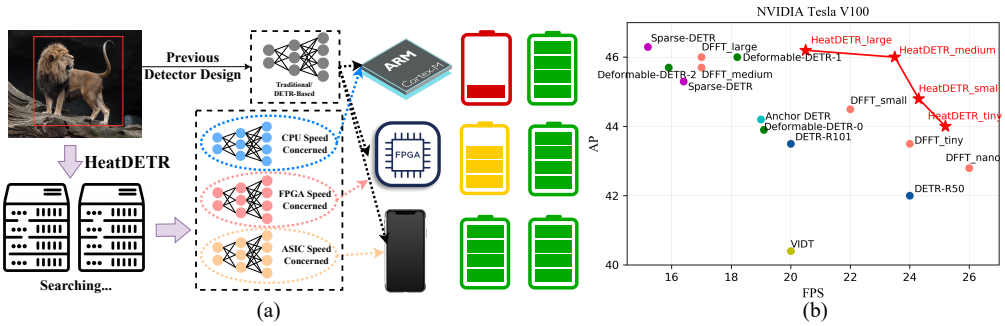
## ABSTRACT

Vision transformers (ViTs) have continuously achieved new milestones in computer vision. A natural usage of ViTs in detection is to replace the CNN-based backbone with a transformer-based backbone directly, but with the price of considerable computation burden for their deployment on resource-limited edge devices. More potential usage is the DETR family, which eliminates the need for many hand-designed components in object detection but still cannot reach real-time edge applications. In this paper, we propose a novel hardware-efficient adaptive-thinning DETR (**HeatDETR**), achieving high speed inference on multiple edge devices and even the realtime, for the first time. Specifically, it mainly includes three contributions: 1) For decent detection performance, we introduce a backbone design principle based on the visual modeling process that focuses on locality to globality. Meanwhile, we propose a semantic-augmented module (SAM) in the backbone with the global modeling capabilities of self-attention to enhance low-level semantics. We also introduce an attention-based task-couple module (TCM) to reduce contradictions between classification and regression tasks. 2) For on-device efficiency, we propose a scale-combined module (SCM), through which we transform the multi-level detection process into the single-level process, releasing the multi-branch inference for higher hardware speed while maintaining detection performance. Then we first revisit network architectures and operators used in ViT-based models, reparametered CNNs, identify hardware-efficient design and introduce basic HeatDETR structure. 3) Based on our device-adaptive model-thinning strategy, deployable end-to-end HeatDETR on target devices can be generated efficiently. Experiments on the MS COCO dataset show HeatDETR outperforms current DETR-based methods by 0.3%$\sim$6.2% AP with 5%$\sim$68% speedup on a single Tesla V100. Even real-time inference can be achieved on extremely memory-constrained devices, e.g., Dual-Core Intel Core i7 CPU.

## 1 INTRODUCTION

With the excellent long-range modeling capabilities, Transformers (Bahdanau et al., 2015; Parikh et al., 2016) have recently made an attractive resurgence in the form of Vision Transformers (ViTs) (Dosovitskiy et al., 2021), showing strong versatility in computer vision tasks (e.g., image classification (Dosovitskiy et al., 2021), object detection (Carion et al., 2020; Zhu et al., 2020), semantic segmentation (Zheng et al., 2021), depth estimation (Yang et al., 2021a), and video understanding (Zhou et al., 2018)). Currently, there are two main branches to applying the transformer structure in object detection. One is developing ViTs as effective backbones (Dosovitskiy et al., 2021; Wang et al., 2021b; Han et al., 2021; Cao et al., 2021) in traditional frameworks such as Faster RCNN (Ren et al., 2015), Mask RCNN (He et al., 2017), and RetinaNet (Lin et al., 2017b), which constantly sets the state-of-art for the object detection task. The other one is the DETR series which utilizes an encoder-decoder transformer design that reduces object detection to an end-to-end set prediction problem. ViTs are largely considered one of the future dominant object detection techniques.

However, to fully unleash the advantages of transformer architectures, we need to address the following challenges before ViTs become an indispensable staple in future object detection systems. (i) Although the self-attention mechanism is a key defining feature of transformer architectures, a well-known concern is its computation and memory complexity (e.g., at least 200 GFLOPs in traditional frameworks; 100 GFLOPs in DETR series). This hinders scalability in many settings, let




(a) (b)

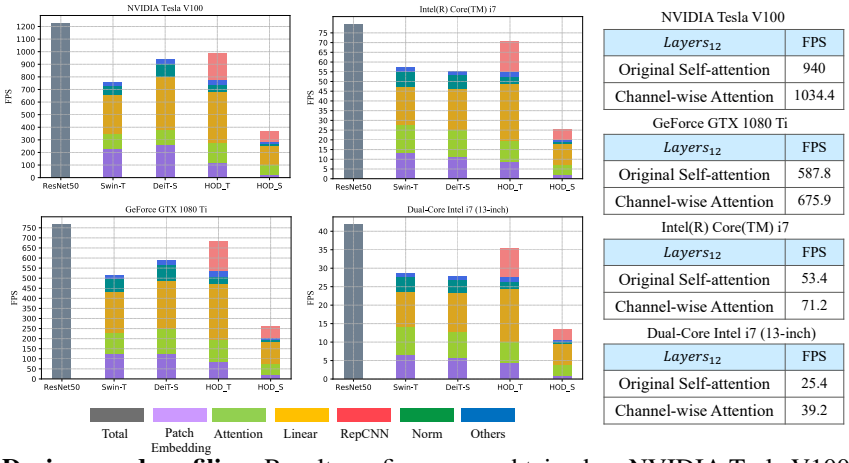
Figure 1: (a) Comparison with previous works in real-world object detection. (b) The trade-off between performance (AP) and hardware efficiency (FPS) for different detection methods. HeatDETR gets faster inference (25.2 FPS $>$ 24 FPS) with 2%$\sim$4.2% higher AP than DETR and outperforms state-of-the-art efficient transformer-based detectors on both precision and efficiency.

alone deployment on resource-constrained edge devices. (ii) Existing transformer-based detectors do not have optimizations for specific hardware deployment, such as memory access cost, degree of parallelism, and compiler characteristics, which can have a non-trivial effect on hardware throughput during inference. (iii) The original self-attention structure converges slowly due to the lack of inductive bias information, similar to the convolutional structure (Khan et al., 2021). Specifically, DETR comes at the expense of around 10$\times$ to 20$\times$ slower training convergence than traditional frameworks, which also restricts constant model improvements to serve evolving needs of real-world platforms. Therefore, we need to address this low hardware efficiency while enjoying the power of long-range modeling from ViTs.

Some pioneering works explore efficient DETR. For instance, Sparse-DETR Roh et al. (2021) added an independent scoring network to image-adaptively remove the redundant information inside the feature map, resulting in the AP improved by 2.8 with a 21% degradation of the inference speed compared with DETR; ViDT Song et al. (2021) redesigned the detection-suitable attention mechanism, leading to inference speed decreasing about 20%$\sim$45% under acceptable detection precision ($>$44% AP). Other works dedicated to improving the training efficiency of the DETR mainly focus on reducing the delayed convergence induced by the decoder. They enhance object queries of the decoder with explicit spatial priors such as anchor points (Sun et al., 2021), RPN proposals (Ren et al., 2015), and conditional spatial embeddings (Gao et al., 2021). However, the newly added spatial prior in the decoder stage impairs the inference efficiency of the detector and consumes more than 1.5$\times$ GFLOPs. Moreover, they lack design optimization for hardware deployment and are mainly tested on high-end GPUs (e.g., V100, A100) for inference speed rather than on resource-limited devices that serve in more real-world applications.

In this paper, we build a novel detection method named HeatDETR: hardware-efficient DETR with device-adaptive thinning, which achieves both higher accuracy and better training-inference efficiency across a spectrum of low resource devices (see Fig. 1).

HeatDETR centers around three points: *Decent detection performance.* 1) Based on the visual modeling process from local to global, we introduce a backbone design principle with a two-phase search space. The first phase mainly includes convolutional and local-wise attention layers, and the second phase further adds global-wise attention layers. 2) Semantic-augmented module (SAM) is incorporated into several stages of the backbone to absorb rich low-level semantics, which benefits the detector identity distractors meticulously. 3) With the help of global attention to capturing abstract-level semantics (global attention capture contextual information inside the data (Khan et al., 2021)), the task-couple module (TCM) reduces conflicts between the classification and regression tasks, providing consistent predictions. *On-device efficiency.* 1) Considering memory access cost and degree of parallelism in resource-limited devices, a scale-combined module (SCM) is designed to transform the multi-level detection process into the single-level one for higher hardware throughput. 2) Furthermore, through the on-device speed analysis of architectures, we identify the basic HeatDETR structure that balances detection precision and hardware efficiency. *Fast model generation.* We perform the device-adaptive model-thinning strategy on the single-width superset, generating the deployable HeatDETR on target edge devices. In addition, these generated models can be trained from scratch directly with 14$\times$ fewer epochs than DETR.




NVIDIA Tesla V100

| $Layers_{12}$ | FPS |
|---|---|
| Original Self-attention | 940 |
| Channel-wise Attention | 1034.4 |

GeForce GTX 1080 Ti

| $Layers_{12}$ | FPS |
|---|---|
| Original Self-attention | 587.8 |
| Channel-wise Attention | 675.9 |

Intel(R) Core(TM) i7

| $Layers_{12}$ | FPS |
|---|---|
| Original Self-attention | 53.4 |
| Channel-wise Attention | 71.2 |

Dual-Core Intel i7 (13-inch)

| $Layers_{12}$ | FPS |
|---|---|
| Original Self-attention | 25.4 |
| Channel-wise Attention | 39.2 |

Figure 2: **Device speed profiling**. Results on figures are obtained on NVIDIA Tesla V100, GeForce GTX 1080 Ti, Intel(R) Core(TM) i7, and Dual-Core Intel i7 (13-inch). The on-device speed for frequently used backbone and various operators are reported. The throughput of models and operations are denoted with different colors. The model-level ($\times$12 layers) comparison of the tables shows that the efficiency of architecture is hardware dependent.

Compared with the classical DETR of similar GFLOPs, HeatDETR achieves over 1 FPS speedup with superior detection performance (2% AP higher); With current state-of-art efficient DETRs, HeatDETR can reach 5%$\sim$68% speedup with up to 6.2% AP higher detection precision; On resource-limited devices, HeatDETR can outperform other existing DETR-based methods in terms of both precision and on-device speed while reaching model scalability. Even real-time inference can be won by chopping off the most resource-constrained devices, Dual-Core Intel Core i7 CPU.

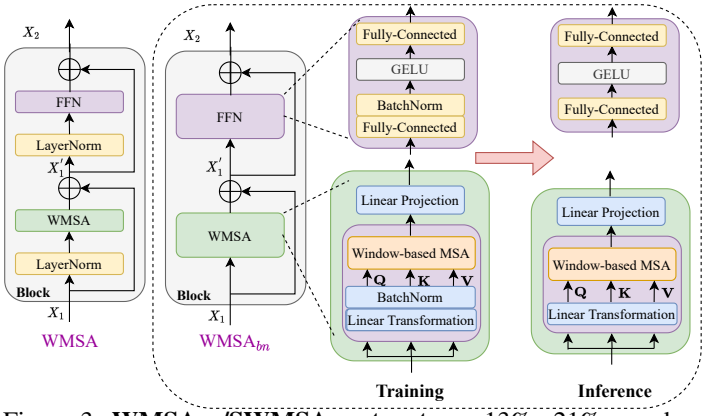
## 2 Related Work

**Traditional Detection Frameworks.** One main challenge in object detection is effectively representing objects at different scales. Both one-stage (Chen et al., 2021; Duan et al., 2019) and two-stage detectors (Cai & Vasconcelos, 2018; He et al., 2017) overcome it with multi-scale features and multi-level predictions. By building a feature pyramid to combine two adjacent layers in a feature hierarchy (with top-down and lateral connections), FPN (Lin et al., 2017a) is widely applied (Tian et al., 2019; Lin et al., 2017b). And through cross-scale connections, e.g., bottom-up paths (Liu et al., 2018b), and U-shape modules (Zhao et al., 2019), later CNN-based designs perform multi-level prediction. Nonetheless, all of these designs introduce extra memory access costs, which can weaken the speed of memory-constrained devices. In contrast, our HeatDETR introduces large receptive fields (of the SCM) to cover large objects and aggregate low-level semantics, thus reducing the multi-level prediction system to the single-level one. Such design ensures superior performance while reducing memory access and improving hardware throughput.

Transformer-based backbones (Chen et al., 2022b) have shown superior performance in traditional frameworks such as Mask RCNN and RetinaNet. However, these backbones are usually inserted directly into the framework without optimizing the efficiency issues after the replacement.

**DETR Family.** End-to-end detectors delete sophisticated post-processing like NMS and achieve matchings between targets and candidates by the Hungarian algorithm. DETR (Carion et al., 2020) utilizes an encoder-decoder transformer framework. The cross-attention module of the decoder attends to different locations in the image for different object queries, which requires high-quality content embeddings. In addition, self-attention naturally has a high degree of freedom due to the lack of inductive bias information similar to convolutional layers. All these factors can lead to larger training costs (500 epochs for DETR training). Deformable DETR (Zhu et al., 2020) accelerates the convergence via learnable sparse sampling and multi-scale deformable encoders. Anchor DETR (Wang et al., 2022) exploits anchor points to accelerate training. Although better performance and fast convergence are achieved through these works, the computation cost has yet to be optimized (e.g., over 170 GFLOPs) due to multi-scale feature encoding.

And these excellent detection performances are typically obtained on high-end GPUs or servers with enormous computation costs (e.g., 170$\sim$10,000 GFLOPs ). Compared with them, HeatDETR is the




$$f^* = W_{linear} \cdot (W_{bn} \cdot f + b_{bn}) + b_{linear}$$
$$W = W_{linear} \cdot W_{bn}$$
$$b = W_{linear} \cdot b_{bn} + b_{linear}$$

| Model | Norm | Top-1 Acc | FPS |
|---|---|---|---|
| Swin-T | LN | 79.8 | 717.4 |
| | **$\text{WMSA}_{bn}$** | **81.2** | **866** |
| Swin-S | LN | 80.9 | 415.2 |
| | **$\text{WMSA}_{bn}$** | **81.4** | **467.2** |

Model-level comparison of WMAS and $\text{WMSA}_{bn}$ with ImageNet-1K on Tesla V100.

Figure 3: **$\text{WMSA}_{bn}$/$\text{SWMSA}_{bn}$ structure**. 13%∼21% speedup can be achieved with <0.3 accuracy degradation on ImageNet-1k.

first work on transformer-based detectors to explore comparable performance and fast inference speed on the resource-limited hardware while maintaining efficiency training.

## 3 On-Device Speed Analysis of Architectures

Design and deployment of efficient network architectures for resource-limited devices have made great strides with consistently reducing parameter count and floating-point operations (FLOPs) while improving accuracy. However, these classical efficiency metrics like FLOPs need to take into account memory access cost and degree of parallelism, which can have a non-negligible impact on inference speed. Parameter count correlates poorly with inference speed as well. We are committed to improving their detection performance while decreasing the speed cost of efficient architectures by identifying congestions that prejudice running speed. To clarify these congestions, we implement neural networks on multiple devices and benchmark their speed costs, as shown in Fig. 2, whereby the following congestions are drawn.

[**Congestion 1**] *Linear-BN is more speed-friendly than LN-Linear, having trivial accuracy drops.*

Considering MLP applies the global receptive field to model information, it is essential to determine the MLP implementation. There are two options: layer normalization with linear projection (LN-Linear); linear with batch normalization (Linear-BN). Linear-BN is more speed-friendly because BN can be fused into the preceding fully-connected layer for inference speedup, while dynamic normalization (LN) still gathers running statistics at the inference stage, thus causing the delay. From the analysis of Swin-Transormer (Liu et al., 2021b) (the main architectures utilized in our design) and DeiT (Touvron et al., 2021), our proposed backbone (HOD) and other standard backbones in Fig. 2, the latency introduced by LN constitutes around 10%∼20% of the whole network.

Inspired by LeViT Graham et al. (2021), we modify the basic structure of Swin-Transormer, WMSA/SWMSA, into $\text{WMSA}_{bn}$/$\text{SWMSA}_{bn}$ as shown in Fig. 3. Compared to the original design with channel-wise LN, a 13%∼20% speedup is harvested with negligible accuracy degradation (<0.3%) on small models. In this paper, we use $\text{WMSA}_{bn}$/$\text{SWMSA}_{bn}$ as our design candidates.

[**Congestion 2**] *Skip-connections or branching can impair speed, while reparametered CNNs applied at the initial layers can speed up with improving recognition performance.*

Two key factors that affect runtime performance are the memory access cost and the degree of parallelism. In a multi-branch structure, the memory access cost increases significantly because the activation of each branch will be stored to compute the following tensor in the graph. Also, the synchronization cost caused by multiple branches affects the overall runtime (Hu et al., 2018). This congestion issue can be sidestepped if the network has a small number of branches.

Thus, Identity $I$ becomes a candidate in our search space to minimize the number of branches while keeping detection precision. Also, we add RepCNN (Ding et al., 2021) to the search space of 1st phase, enabling more single-branch substructures during inference (see Sec. 4.3). We experimentally find that recognition performance is improved by 0.3%∼0.5% AP after replacing WMSA with RepCNN in Stage 1 of backbone. The fact may be that RepCNN utilizes the 3×3 convolutional kernel, which has a stronger ability to model locality than WMSA with the 7×7 attention window.

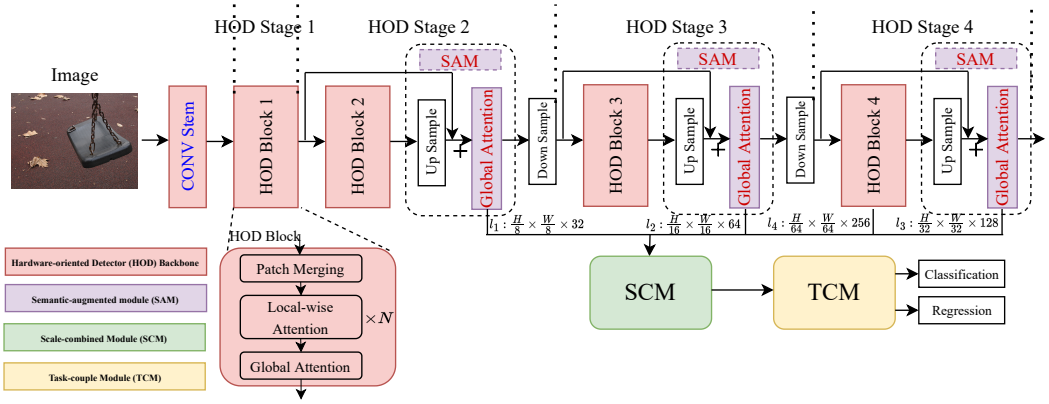



Figure 4: **The basic design of HeatDETR .** The TCM and SCM enable us to perform fast but accurate detection on a single-level feature map. Considering the visual modeling process and hardware specifications, we propose a HOD backbone to maintain detection precision while improving the on-device speed.

**[Congestion 3]** *The speed of different architectures is hardware-dependent.*

The specifications of hardware memory affect the inference speed of the model in real-world applications. The current cache size of edge devices is about 2M∼8M with the bus speed of memory I/O transmission, 2 Gbps∼10 Gbps. And the part that exceeds the model and intermediate data size will be transferred to RAM. The speed of CPU reading data from RAM is about $10\times\sim20\times$ slower than reading directly from the cache Laguna et al. (2019). So the model with a large size and much intermediate data can attenuate the throughput of hardware with less cache and less bus speed of memory I/O. Previous work Ali et al. (2021) claims that channel-wise attention is more efficient than original self-attention, but the difference in speed is small on NVIDIA V100, large on Intel(R) Core(TM) i7 and Dual-Core Intel i7 (13-inch). With a linear memory complexity, channel-wise attention is probably a memory-friendly architecture.

## 4 Design of HeatDETR

In this section, we introduce HeatDETR, a hardware-efficient Transformer-based object detector, as shown in Fig. 4. The hardware-oriented detector (HOD) backbone extracts features at four scales and sends them to the following task-coupled single-level prediction system. The scale-combined module (SCM) first combines the multi-scale features into single-level feature maps. Then, the task-couple module (TCM) adjusts feature maps to reduce conflicts between classification and regression tasks simultaneously, and finalizes the detection task.

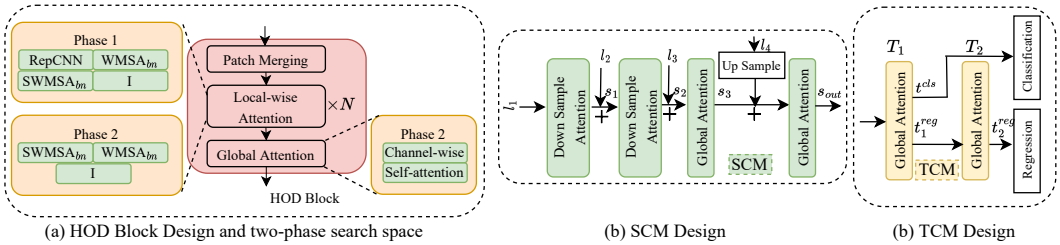
### 4.1 The Hardware-Efficient Backbone of Local-Global Modeling

**Backbone Design Principle.** HOD backbone aims to extract multi-scale features with strong semantics while guaranteeing the efficiency of the target hardware. Inspired by the visual modeling process inside ResNet (He et al., 2016), we divide the backbone into four HOD stages $S$, where feature scales have a trend from the local to the global visual receptive field. Then, to enhance the semantic information of low-level features in each HOD stage, one semantic-augmented module (SAM) is added after one HOD block (residual effect). In conjunction with the hardware speed analysis, we provide the two-phase search space (see Sec. 4.3). For each input image $x\in R^{H\times W\times 3}$, the HOD backbone extracts features at four different scales:

$$l_1, l_2, l_3, l_4 = L(x), \tag{1}$$

where $l_i\in R^{\frac{H}{s\cdot 2^{i-1}}\times\frac{W}{s\cdot 2^{i-1}}\times 3}$ is the $i$-th feature map with $D_i$ channels for $i\in\{1,2,3,4\}$, and $s$=8 to balance the detection precision and computational costs. In the following, we represent the HOD backbone as function $L$.

**Image Embedding Module.** Patch embedding is typically implemented with a non-overlapping convolutional layer that has a large kernel size and stride. However, our analysis in Fig. 2 shows that patch embedding with large kernels and stride is a speed bottleneck on multiple devices. Large-kernel convolutions are not well supported by most compilers nor accelerated by existing algorithms (e.g., Winograd). Thus, we utilize a convolution stem with fast downsampling, consisting of 3




(a) HOD Block Design and two-phase search space (b) SCM Design (b) TCM Design

Figure 5: The schematic of three major modules in HeatDETR.

hardware-efficient 3×3 convolutions with stride 2, as an image embedding module:

$$l_0^{\frac{H}{s}\times\frac{W}{s}\times D_1} = PatchEmbed(x^{H\times W\times 3}). \tag{2}$$

**Hardware-Oriented Detector Block.** Each HOD block (see Fig. 5(a)) is designed to effectively capture the local (texture-level semantics) and global information (abstract-level semantics) at each scale. So we extract texture-level semantics through several consecutive local-wise blocks following the local-to-global principle. Then reinforce abstract-level semantics in the feature map through a global attention block. The search space of local-wise blocks mainly includes RepCNN, WMSA$_{bn}$, and SWMSA$_{bn}$, while original self-attention (Touvron et al., 2021) and channel-wise attention (Ali et al., 2021) for global attention block. Note that SWMSA$_{bn}$ can compensate for the missing globality in RepCNN and WMSA$_{bn}$ productively. However, the 7×7 window-attention operation still limits its receptive field to cover the whole image, so it is necessary to add one global attention block after local-wise blocks.

**Semantic-Augmented Module.** Similar to the residual effect, we propose SAM to further enhance low-level semantics in the current feature map for the detection, as shown in Fig. 4. SAM consists of an upsampling layer and a global attention block. By inserting the SAM into every two consecutive HOD blocks, low-level semantics can complement high-level semantic information. SAM first aligns the scale level of the current HOD block with that of the previous one and then generates enhanced semantic features. Those features flow to the next HOD stage and the SCM, respectively.

**HOD Stage.** The final HOD backbone contains four HOD stages, each consisting of a HOD block and a SAM (except for stage 1).The output of the HOD backbone can be defined as

$$l_i^* = \begin{cases} L_{hod}(l_{i-1}^*), & \text{if } i=1,2 \\ L_{hod}(down(l_{i-1}^{\Delta}))), & \text{if } i=3,4 \end{cases}, \quad l_i^{\Delta} = \begin{cases} L_{sam}(up(l_i^*)+l_{i-1}^*), & \text{if } i=2 \\ L_{sam}(up(l_i^*)+l_{i-1}^{\Delta}), & \text{if } i=3,4 \end{cases}, \quad l_i = \begin{cases} l_{i+1}^{\Delta}, & \text{if } i=1,2,3 \\ l_i^*, & \text{if } i=4 \end{cases} \tag{3}$$

where $down$ denotes the downsampling function while $up$ for upsampling; $l^*$ is the output of $i$-th HOD block while $l^{\Delta}$ being the one of the $i$-th SAM; $l_i$ is final multi-scale features of the HOD backbone.

### 4.2 Task-Couple Single-Level Detection System

**Scale-Combined Module.** To reduce the large memory I/o overhead and the high parallelism caused by multi-branch, we transform multi-level into single-level prediction by SCM (see Fig. 5(b)) with global attention to extract abstract-level semantics (long-range modeling). The process is

$$\begin{aligned} s_1 = downsample\_attention(l_1)+l_2, \quad s_2 = downsample\_attention(s_1)+l_3, \\ s_3 = global\_attention(s_2), \quad s_{out} = global\_attention(s_3+up(l_4)). \end{aligned} \tag{4}$$

Inspired by Metaformer (Yu et al., 2022), we apply average_pooling (stride=2) as the token mixer to enhance abstract-level semantics in features while downsampling data. $downsample\_attention$ is

$$I_i^{B,C,\frac{H}{2^{j+1}},\frac{W}{2^{j+1}}} = Pool(x_i^{B,C,\frac{H}{2^j},\frac{W}{2^j}}), \quad x_{i+1}^{B,C,\frac{H}{2^{j+1}},\frac{W}{2^{j+1}}} = CONV_B(CONV_{B,G}(I_i))+I_i, \tag{5}$$

where $CONV_{B,G}$ refers to whether the convolution is followed by BN and GeLU. $global\_attention$ refers to the transformer block with channel-wise attention or original self-attention block with the adaptive token sparsity (Feng et al., 2022) (we use $<18\%$ sparsity). Due to the sequential computing of transformer block, we can always concatenate the classified sparse tokens into dense ones to improve the hardware efficiency, which is challenging for CNN-based architecture. Please note

that in some high-end devices such as V100, we deploy the transformer block with the original self-attention for the higher detection performance while channel-wise attention block in memory-constant devices.

**Task-Couple Module.** One-stage detectors perform object classification and localization independently with two separate branches (e.g., decoupled heads). This two-branch design neglects interactions and conflicts between the two tasks, leading to inconsistent prediction results (Dai et al., 2021a; Song et al., 2020). We propose TCM in Fig. 5(c), which guides learning task interactions and eliminates conflicts between task-specific features by stacking global attention blocks $T$.

The global attention block $T_1$ couples and splits the single-level feature $s_{out}$ into two parts. After that, $T_2$ encodes one of the parts for the subsequent regression task. As follows,

$$t^{cls}, t_1^{reg} = T_1(s_{out}), \quad t_2^{reg} = T_2(t_1^{reg}). \tag{6}$$

We use the similar strategy as SCM for the model implementation.

### 4.3 Device-Adaptive Model Thinning Strategy

**Design of Supernet.** We introduce a two-phase supernet for a hardware-efficient backbone, as shown in Fig. 5. We only search the structures and dimensions of the backbone, while SCM and TCM use the fixed structures with the dimension adapted to the backbone. We also define the HeatPath (HP) as the collection of possible blocks:

$$\begin{aligned} HP^1_{i,j=1,2,3} &\in \{RepCNN, WMSA_{bn}, SWMSA_{bn}, I\}, \\ HP^2_{i,j=4} &\in \{WMSA_{bn}, SWMSA_{bn}, I, Channel-wise, Original-atten\}. \end{aligned} \tag{7}$$

To make the visual modeling a process from locality to globality, the 1st phase covers $S_1$, $S_2$, $S_3$ of the backbone, and the 2nd phase for $S_4$. Reducing the number of model branches, we add Identity $I$ as a search candidate; considering the excessive feature scale in $S_1$, $S_2$, $S_3$, the global attention used in HOD and SAM is fixed to channel-wise attention for hardware efficiency; for the global attention part in $S_4$, channel-wise and original self-attention are added as candidates for network search.

**Searching Space.** Our searching space includes $D_i$ (the dimension of each $S$), $N_j$ (the number of blocks in each $S$).

**Device-Adaptive Model Thinning Algorithm.** We propose a simple, fast, and effective gradient-based search algorithm to obtain a candidate network that requires only once training of the supernet. The algorithm has three steps.

1) We train the supernet with the Gumble Softmax sampling Liu et al. (2018a) to get the importance score for the blocks within each $HP$, which can be expressed as

$$\kappa_{i+1} = \sum_n \left( \frac{e^{(r_i^n + \varepsilon_i^n)/\tau}}{\sum_n e^{(r_i^n + \varepsilon_i^n)/\tau}} \cdot HP_{i,j}(\kappa_i) \right), \tag{8}$$

where $r$ represents the importance of each block in $HP$ by calculating the probability to select a block (candidate), e.g., $WMSA_{bn}$ or $RepCNN$ for the $i$th block; $\varepsilon \sim U(0,1)$ enhances exploration; $\tau$ is the temperature; $n$ represents the type of blocks in $HP$. By Eq. 8, the derivatives related with the architecture search can be easily computed. The training follows the recipe in Appendix 9.

2) We build a FPS lookup table by collecting the on-device speed of each search candidate (except for $I$ ) with different widths (multiples of 32) for multiple devices.

3) We perform device-adaptive model thinning on the supernet obtained from step 1) by FPS evaluation with the device lookup table. Instead of building up a multiple-width supernet that is memory-inefficiency, we design a gradual thinning on the single-width supernet. Specifically, we define the importance score for $HP_i$ as $\frac{r_i^{RepCNN} + r_i^{WMSA} + r_i^{SWMSA}}{r_i^I}$ and $\frac{r_i^{WMSA} + r_i^{SWMSA}}{r_i^I}$ for $S_1$, $S_2$, and $S_3$ and $S_4$. We obtain the importance score $r$ for each $S$ by summing up the scores for all $HP$ within that $S$. Then we define the evolution space (all performed in the current least important $S$): a) remove $I$ ; b) remove the first $SWMSA$ ; c) remove the first $WMSA$; d) reduce the width by multiples of 32. Then we calculate the current FPS of each evolution through a lookup table, and evaluate the accuracy drop of each evolution. We choose the evolution based on $FPS\_AP\_drop(\frac{-\%}{fps})$. This process is repeated until the target throughput is reached (Appendix 9).

Table 1: The references used in Table 2.

| Model | Reference | Model | Reference | Model | Reference |
|---|---|---|---|---|---|
| DETR | Carion et al. (2020) | Mobile-Former | Chen et al. (2022b) | Focal-Tiny-RetinaNet | Yang et al. (2021b) |
| VIDT | Song et al. (2021) | Faster RCNN | Ren et al. (2015) | Anchor DETR | Wang et al. (2022) |
| YOLOF | Chen et al. (2021) | WB-DETR | Liu et al. (2021a) | Deformable DETR | Zhu et al. (2020) |
| DFFT | Chen et al. (2022a) | PnP-DETR | Wang et al. (2021a) | Conditional DETR | Meng et al. (2021) |
| RetinaNet | Lin et al. (2017b) | UP-DETR | Dai et al. (2021b) | SAM-DETR | Zhang et al. (2022) |
| SMCA | Gao et al. (2021) | TSP-FCOS | Sun et al. (2021) | Sparse-DETR | Roh et al. (2021) |
| YOLOS | Fang et al. (2021) | DAB-DETR | Liu et al. (2022) | Swin-Tiny-RetinaNet | Liu et al. (2021b) |

Table 2: Comparison of HeatDETR and common detection methods on MS COCO benchmark. The table is divided into four parts: 1) anchor-based methods; 2) HeatDETR trained for 12 epochs; 3) DETR-based methods; 4) HeatDETR trained for 36 epochs. HeatDETR achieves competitive precision with fewer inference FPS and training epochs. FPS is measured with a batch size 1 of 800×1333 resolution on a single Tesla V100 GPU.

| Method | Epochs | AP (%) | $AP_{50}$ (%) | $AP_{75}$ (%) | $AP_S$ (%) | $AP_M$ (%) | $AP_L$ (%) | GFLOPs | FPS |
|---|---|---|---|---|---|---|---|---|---|
| Faster RCNN-FPN-R50 | 36 | 40.2 | 61 | 43.8 | 24.2 | 43.5 | 52 | 180 | - |
| Focal-Tiny-RetinaNet | 12 | 43.7 | - | - | - | - | - | 265 | - |
| Swin-Tiny-RetinaNet | 12 | 42 | - | - | - | - | - | 245 | - |
| YOLOF-R50 | 12 | 39.2 | 58.6 | 42.7 | 22.3 | 43.9 | 50.8 | 103 | 24 |
| RetinaNet | 12 | 35.9 | 55.7 | 38.5 | 19.4 | 39.5 | 48.2 | 201 | - |
| Mobile-Former | 12 | 34.2 | 53.4 | 36.0 | 19.9 | 36.8 | 45.3 | 322 | - |
| HeatDETR_tiny | 12 | 40.9 | 60.2 | 43.5 | 21.4 | 44.6 | 55.7 | 67 | 25.2 |
| HeatDETR_small | 12 | 41.7 | 59.5 | 44.5 | 23.2 | 45.6 | 56.3 | 69 | 24.3 |
| HeatDETR_medium | 12 | 42.9 | 60.9 | 46.2 | 23.9 | 47.1 | 58.7 | 73 | 23.5 |
| DETR-R50 | 500 | 42 | 62.4 | 44.2 | 20.5 | 45.8 | 61.1 | 86 | 24 |
| WB-DETR | 500 | 41.8 | 63.2 | 44.8 | 19.4 | 45.1 | 62.4 | 98 | - |
| PnP-DETR | 500 | 41.8 | 62.1 | 44.4 | 21.2 | 45.3 | 60.8 | - | - |
| UP-DETR | 150 | 40.5 | 60.8 | 42.6 | 19 | 44.4 | 60 | - | - |
| YOLOS | 150 | 37.6 | - | - | - | - | - | 172 | 5.7 |
| Anchor DETR-DC5-R50 | 50 | 44.2 | 64.7 | 47.5 | 24.7 | 48.2 | 60.6 | 151 | 19 |
| Deformable DETR | 50 | 43.9 | 62.6 | 47.7 | 26.4 | 47.1 | 58 | 173 | 19.1 |
| SMCA-R50 | 50 | 43.7 | 63.6 | 47.2 | 24.2 | 47 | 60.4 | 152 | - |
| SAM-DETR-R50-DC5 | 50 | 43.3 | 64.4 | 46.2 | 25.1 | 46.9 | 61 | 210 | - |
| TSP-FCOS-R50 | 36 | 43.1 | 62.3 | 47 | 26.6 | 46.8 | 55.9 | 189 | 15 |
| DAB-DETR-R50 | 50 | 42.6 | 63.2 | 45.6 | 21.8 | 46.2 | 61.1 | 100 | - |
| Conditional DETR-R50 | 50 | 40.9 | 61.8 | 43.3 | 20.8 | 44.6 | 59.2 | 90 | - |
| VIDT | 50 | 40.4 | 59.6 | 43.3 | 23.2 | 42.5 | 55.8 | - | 20 |
| Sparse-DETR | 50 | 45.3 | 65.8 | 49.3 | **28.4** | 48.3 | 60.1 | 105 | 16.4 |
| DFFT | 36 | 44.5 | 63.6 | 48 | 24.5 | 49 | 60.7 | **62** | 22 |
| HeatDETR_tiny | 36 | 44 | 64.7 | 46.8 | 23 | 47.9 | 59.9 | 67 | **25.2** |
| HeatDETR_small | 36 | 44.8 | 64 | 47.8 | 24.9 | 48.9 | 60.5 | 69 | 24.3 |
| HeatDETR_medium | 36 | 46 | 65.3 | 49.8 | 26 | 50.2 | 62.8 | 73 | 23.5 |
| HeatDETR_large | 36 | **46.2** | **65.9** | **49.8** | 28.3 | **50.5** | **62.9** | 108 | 20.5 |

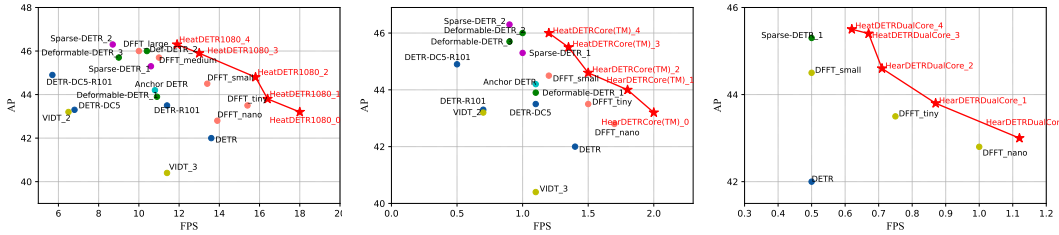
## 5 Experimental Results

We evaluate our proposed HeatDETR on the MS COCO benchmark (Lin et al., 2014), which uses the common settings. It contains about 160,000 images from 80 categories. We compared HeatDETR with traditional frameworks and DETR family as Table 1. The HOD backbone was pretrained on ImageNet (Deng et al., 2009) with the same setting as Liu et al. (2021b). We trained HeatDETR with standard 1× (12 epochs) and 3× (36 epochs) training configurations. We use the AdamW optimizer with a batch size of 32 and an initial learning rate of 1e-4, the same setting as VIDT. All experiments were running on a server with 8 Tesla V100 GPUs. The detailed detection training settings and model configuration are provided in Appendix 9.

### 5.1 Main Results

**Comparison with Traditional Frameworks.** As shown in Part 1 of Table 2, anchor-based methods converge fast within only 12 epochs, and the transformer-based methods generally outperform CNN-based methods but with higher GFLOPs. Focal-Tiny-RetinaNet achieves 7.8% higher AP than the original RetinaNet at the expense of 32% computation costs. Generally, such good performance comes with over 170 GFLOPs high computational costs. YOLOF is an efficient single-level feature method with 103 GFLOPs, while having 39.2%<40% AP.

Compared to the aforementioned approaches that encounter a clear unbalance between detection precision and inference speed (Part 2), our HeatDETR can improve both metrics based on its efficient design. HeatDETR_tiny is 9% faster than YOLOF with 1.7% higher AP (40.9%>39.2%), while the speed of HeatDETR_small and YOLOF are nearly equal but with 2.5% higher AP (41.7%>39.2%). In addition, HeatDETR reduces 200 GFLOPs from the best-performing Focal-Tiny-RetinaNet at the cost of only a 0.8% AP reduction.




(a) HeatDETR on GeForce GTX 1080 Ti. (b) HeatDETR on Intel(R) Core(TM) i7. (c) HeatDETR on Dual-Core Intel i7 (13-inch).

Figure 6: Comparison on multiple resource-limited devices of HeatDETR.

**Comparison with DETR Family.** As shown in Part 3 of Table 2, DETR-based methods can achieve a better trade-off between performance and inference speed but converge slower. DETR only needs 86 GFLOPs and 24 FPS to achieve 42.0% AP during inference but requires 500 epochs to converge. WB-DETR and PnP-DETR need even 500 epochs to achieve 41.8% AP merely. Other optimized DETR-based methods improve the convergence speed but are more inefficient. Deformable DETR, Anchor DETR, and TSP-FCOS need 50 or 36 epochs to converge, but their GFLOPs are around 76%∼120% larger than DETR. And even though all of these three have 15∼19 FPS on the high-end GPU, Tesla V100, they cannot achieve acceptable speeds on resource-constraint devices, e.g., Intel(R) Core(TM) i7 and Dual-Core Intel i7 (13-inch), as shown in Fig. 6, because of their high computation and memory complexity.

On the contrary, HeatDETR can achieve state-of-the-art detection prevision without sacrificing both the convergence and inference speed. Compared with classical DETR, our HeatDETR_tiny model improves 14× training efficiency with 1.2 FPS inference speedup while achieving superior detection performance (42.0% vs. 44%). Considering current state-of-the-art efficient DETR frameworks, VIDT, Sparse-DETR, and DFFT, our model can reach 9%∼43% speedup with 0.9%∼5.8% AP higher precision. HeatDETR reduces 28%–92% training epochs of existing methods as well.

### 5.2 Hardware Speed on Resource-limited Devices

To evaluate the hardware throughput, we implement the model thinning on three resource-limited devices: GeForce GTX 1080 Ti which has a 2.8M size L2 cache with 11 Gbps memory bandwidth; Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz which has 1M (×12 kernel) size L2 cache with 8 Gbps memory bandwidth; 13-inch Dual-Core Intel Core i7 CPU @ 2.50GHz, which has a 4M size L2 cache with 2.5 Gbps memory bandwidth. We report the average FPS of over 100 inferences. As shown in Fig. 6, our method outperforms existing efficient DETR frameworks on both hardware efficiency and detection performance. Note that HeatDETR even achieves real-time inference speed on the most resource-constrained device, i.e., Dual-Core Intel Core i7 CPU.

Since other methods do not take into account the memory limitations and parallelism of the on-device runtime, in general, their performance can be degraded further on resource-limited devices and do not even give results in a reasonable time. This also constrains their scalability for practical application scenarios. Deformable-DETR and VIDT cannot run on the Dual-Core Intel Core i7 (2.5 Gbps memory bandwidth), which has the highest memory I/O limitation, while on both other devices, they are faster than Sparse-DETR-1 blocks. The possible reason is that the backbone of Deformable-DETR has a quadratic memory complexity, and VIDT has an inefficient decoder neck, without which precision degradation is not trivial. Therefore, we infer that HeatDETR can compensate for this hardware-software disconnect when custom accelerators are available in the regime of efficient architectures.

## 6 Conclusion

In this paper, we propose a novel hardware-oriented detection pipeline named HeatDETR, which achieves both higher accuracy and better training-inference efficiency across multiple resource-limited devices. Combined with the visual modeling process and hardware specifications, HeatDETR exceeds current DETR-based methods by 0.3%∼6.2% AP with 5%∼68% speedup on high-end GPU with 28%∼92% fewer training epochs. Compared to other approaches, HeatDETR is more potential for practical applications due to its scalability on resource-limited devices. How to improve the precision of transformer-based detectors on edge will be our next effort.

## 7 ETHICS STATEMENT

I will abide by the laws, rules and regulations of my community, school, work and country. I will conduct myself with integrity, loyalty and honesty. I will be openly accountable for my actions and will only do what I intend to do to comply with the protocol. Vision Transformer is already making a splash in the purely algorithmic world, but optimized hardware which can support it is just emerging. I hope that our entire community will work together to advance the use of Vision Transformer on the edge side.

## 8 REPRODUCIBILITY STATEMENT

In order to promote academic development, and efficient conversion of capacity, the relevant codes for this article are provided in the supplementary material.

## REFERENCES

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021. 5, 6

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015. 1

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018. 3

Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. *arXiv preprint arXiv:2105.05537*, 2021. 1

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229. Springer, 2020. 1, 3, 8

Peixian Chen, Mengdan Zhang, Yunhang Shen, Kekai Sheng, Yuting Gao, Xing Sun, Ke Li, and Chunhua Shen. Efficient decoder-free object detection with transformers. *arXiv preprint arXiv:2206.06829*, 2022a. 8

Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13039–13048, 2021. 3, 8

Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5270–5279, 2022b. 3, 8

Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7373–7382, 2021a. 7

Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1601–1610, 2021b. 8

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 8

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021. 4

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1

Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6569–6578, 2019. 3

Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34:26183–26197, 2021. 8

Qihua Feng, Peiya Li, Zhixun Lu, Chaozhuo Li, Zefang Wang, Zhiquan Liu, Chunhui Duan, and Feiran Huang. Evit: Privacy-preserving image retrieval via encrypted vision transformer in cloud computing. *arXiv preprint arXiv:2208.14657*, 2022. 6

Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3621–3630, 2021. 2, 8

Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12259–12269, 2021. 4

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Advances in Neural Information Processing Systems*, 2021. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 5

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 1, 3

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, 2018. 4

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021. 2

Ann Franchesca Laguna, Michael Niemier, and X Sharon Hu. Design of hardware-friendly memory enhanced neural networks. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1583–1586. IEEE, 2019. 5

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014. 8

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017a. 3

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017b. 1, 3, 8

Fanfan Liu, Haoran Wei, Wenzhe Zhao, Guozhen Li, Jingquan Peng, and Zihao Li. Wb-detr: Transformer-based detector without backbone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2979–2987, 2021a. 8

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018a. 7

Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022. 8

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018b. 3

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *International Conference on Computer Vision (ICCV)*, 2021b. 4, 8

Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3651–3660, 2021. 8

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, 2016. 1

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 2, 8

Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. *arXiv preprint arXiv:2111.14330*, 2021. 2, 8

Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11563–11572, 2020. 7

Hwanjun Song, Deqing Sun, Sanghyuk Chun, Varun Jampani, Dongyoon Han, Byeongho Heo, Wonjae Kim, and Ming-Hsuan Yang. Vidt: An efficient and effective fully transformer-based object detector. *arXiv preprint arXiv:2110.03921*, 2021. 2, 8

Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3611–3620, 2021. 2, 8

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019. 3

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021. 4, 6

Tao Wang, Li Yuan, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Pnp-detr: Towards efficient visual analysis with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4661–4670, 2021a. 8

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021b. 1

Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2567–2575, 2022. 3, 8

Guanglei Yang, Hao Tang, Mingli Ding, Nicu Sebe, and Elisa Ricci. Transformer-based attention networks for continuous pixel-wise prediction. In *ICCV*, 2021a. 1

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers, 2021b. 8

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10819–10829, 2022. 6

Gongjie Zhang, Zhipeng Luo, Yingchen Yu, Kaiwen Cui, and Shijian Lu. Accelerating detr convergence via semantic-aligned matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 949–958, 2022. 8

Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 9259–9266, 2019. 3

Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6881–6890, 2021. 1

Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8739–8748, 2018. 1

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3, 8

# 9 APPENDIX

## 9.1 DEVICE-ADAPTIVE MODEL THINNING STRATEGY

We provide the details of the proposed device-adaptive model thinning strategy in Alg. 1. Relative formulations can be found in Sec. 4.3. The proposed device-adaptive model thinning strategy is speed-oriented for the target device, which does not need retraining for each sub-network. The importance score for each device-design choice is estimated based on the trainable architecture parameter $r$.

**Algorithm 1:** Device-Adaptive Model Thinning Strategy based on Importance Estimations

**Given**: speed look up table $F = \{RepCNN, WMSA_{bn}, SWMSA_{bn}\}_{dim=32\times}$, $\{RepCNN, WMSA_{bn}, SWMSA_{bn}, Channel\_wise, Original\_atten\}_{dim=32\times}$;
**Requirement**: Final throughput satisfy budget: $\sum F \approx \top$ ;
*Super-net Pretraining:*
**foreach** *epoch* **do**
  **foreach** *each iteration* **do**
    **foreach** $HP_{i,j}$ **do**
      $\kappa_{i+1} = \sum_n (\frac{e^{(r_i^n + \varepsilon_i^n)/\tau}}{\sum_n^e (r_i^n + \varepsilon_i^n)/\tau} \cdot HP_{i,j}(\kappa_i))$;
    **end**
    $\pounds = criterion\_loss\_function(output, label)$;
    backpropagate (£), update parameters;
  **end**
**end**
*Speed-Driven Model thinning:*
E ∈ {Layer Reduction (LR), Width Reduction (WR), I Reduction (IR), WMSA Reduction (WMR), SWMSA Reduction (SR)};
Calculate the importance of $HP_{i,j}$ through $\mathbb{M}_{i,j} = \frac{r_i^{RepCNN} + r_i^{WMSA} + r_i^{SWMSA}}{r_i^I}$ or $\frac{r_i^{WMSA} + r_i^{SWMSA}}{r_i^I}$;
**while** $\sum F > \top$ **do**
  $LR \leftarrow argmin_{\mathbb{M}_{i,j}}(HP_{i,j})$, $IR \leftarrow argmin_{\sum_j \mathbb{M}_{i,j}}(HP_{i,j})$, $SR \leftarrow argmin_{\sum_j \mathbb{M}_{i,j}}, (HP_{i,j})$, $WMR \leftarrow argmin_{\sum_j \mathbb{M}_{i,j}}(HP_{i,j})$, $DR \leftarrow argmin_{\sum_j \mathbb{M}_{i,j}}(HP_{i,j})$;
  Conduct Evolution = $argmin_{\frac{AP_{drop}}{F_{i,j}}}(E)$;
**end**
*Train the searched architecture from scratch:*
SDG-Training method.

## 9.2 DETAILED TRAINING SETTINGS

Since HeatDETR performs single-stage detection on a single feature map, the predefined anchor points are sparse. Applying Max-IoU matching based on sparse anchor points will lead to the imbalance problem of positive anchor points, which means that the detector will focus on large ground truth boxes and ignore small ground truth boxes during the training process. To overcome this problem, we use the uniform matching strategy proposed by YOLOF to keep all ground truth frames uniformly matched with the same number of positive anchors regardless of their sizes. Similar to the setup of most traditional detection frameworks, our loss function is composed of a focal loss for classification and a generalized IOU loss for regression. In the inference stage, we effectively perform an object detection based on the final aggregated feature map $l^{SAM}$.

## 9.3 MODEL SETTINGS

Based on our proposed device-adaptive model thinning strategy, we obtained a serious of detectors on Tesla V100 as Table 3, where $D_i$ represents the number of dimenrion of output feature $l_i$ of the $S_i$, and the number of HOD blocks and the corresponding types of local-wise blocks within each DOT stage is also provided. Only one global attention block is added to the end of each stage. Meanbile, the target device is high-end GPUs, so we utilize original self-attention with adaptive sparsity on the

Table 3: The definition and performance of HeatDETR models with different magnitudes.

| Models | Backbone Settings | | | Performance | | Efficiency | |
|---|---|---|---|---|---|---|---|
| | $D_i$ | Number of Blocks | Type of Blocks | Accuracy | AP | GFLOPS | FPS |
| HeatDETR_tiny | 128, 128, 256, 384 | 1, 1, 6, 1 | RepCNN$\times$1, RepCNN$\times$1, ( WMSA, SWMSA)$\times$3 WMSA$\times$1 | 81.5 | 44 | 67 | 25.2 |
| HeatDETR_small | 128, 128, 256, 384 | 2, 2, 6, 2 | RepCNN$\times$2, (RepCNN, SWMSA)$\times$2, (RepCNN, SWMSA)$\times$1, (WMSA, SWMSA)$\times$2,(RepCNN, SWMSA)$\times$1 | 82.4 | 44.8 | 69 | 24.3 |
| HeatDETR_medium | 128, 128, 224, 384 | 2, 2, 18, 2 | RepCNN$\times$2, (RepCNN, SWMSA)$\times$2, (RepCNN, SWMSA)$\times$4, ( WMSA, SWMSA)$\times$5,(RepCNN, SWMSA)$\times$1 | 83 | 46 | 73 | 23.5 |
| HeatDETR_large | 192,192, 256, 384 | 2, 2, 18, 2 | RepCNN$\times$2, (RepCNN, SWMSA)$\times$2, (RepCNN, SWMSA)$\times$4, ( WMSA, SWMSA)$\times$5,(RepCNN, SWMSA)$\times$1 | 83.3 | 46.2 | 108 | 20.5 |

final HOD stage, SAM, SCM and TCM parts. For each model, accuracy refers to the accuracy of backbone on ImageNet and AP refers to the precision after training on the MS COCO dataset. All GFLOPs are obtained on the MS COCO dataset. Please note that the model configuration is adapted to the runtime specifications of the target devices.