Lite DETR : An Interleaved Multi-Scale Encoder for Efficient DETR

Feng Li[1,2]*, **Ailing Zeng**[2], **Shilong Liu**[2,3], **Hao Zhang**[1,2], **Hongyang Li**[2,4]
Lei Zhang[2]†, **Lionel M. Ni**[1,5]
[1]The Hong Kong University of Science and Technology.
[2]International Digital Economy Academy (IDEA).
[3]Dept. of CST., BNRist Center, Institute for AI, Tsinghua University.
[4]South China University of Science and Technology.
[5]The Hong Kong University of Science and Technology (Guangzhou).

{fliay,hzhangcx}@connect.ust.hk {liusl20}@mails.tsinghua.edu.cn {eeli.hongyang}@mail.scut.edu {leizhang}@idea.edu.cn {ni}@ust.hk

Abstract

Recent DEtection TRansformer-based (DETR) models have obtained remarkable performance. Its success cannot be achieved without the re-introduction of multi-scale feature fusion in the encoder. However, the excessively increased tokens in multi-scale features, especially for about 75% of low-level features, are quite computationally inefficient, which hinders real applications of DETR models. In this paper, we present Lite DETR, a simple yet efficient end-to-end object detection framework that can effectively reduce the GFLOPs of the detection head by 60% while keeping 99% of the original performance. Specifically, we design an efficient encoder block to update high-level features (corresponding to small-resolution feature maps) and low-level features (corresponding to large-resolution feature maps) in an interleaved way. In addition, to better fuse cross-scale features, we develop a key-aware deformable attention to predict more reliable attention weights. Comprehensive experiments validate the effectiveness and efficiency of the proposed Lite DETR, and the efficient encoder strategy can generalize well across existing DETR-based models. The code will be available in https://github.com/IDEA-Research/Lite-DETR.

1. Introduction

Object detection aims to detect objects of interest in images by localizing their bounding boxes and predicting the corresponding classification scores. In the past decade, remarkable progress has been made by many classical detection models [23, 24] based on convolutional networks.

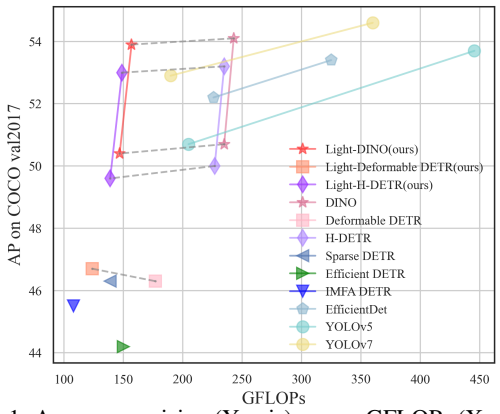

Figure 1. Average precision (Y axis) versus GFLOPs (X axis) for different detection models on COCO without extra training data. All models except EfficientDet [29] and YOLO series [12, 30] use ResNet-50 and Swin-Tiny as backbones. Specifically, two markers on the same line use ResNet-50 and Swin-Tiny, respectively. Individual markers only use ResNet-50. Each dashed line connects algorithm variants before and after adding our algorithm. The size of the listed models vary from 32M to 82M.

Recently, DEtection TRansformer [1] (DETR) introduces Transformers into object detection, and DETR-like models have achieved promising performance on many fundamental vision tasks, such as object detection [13, 36, 37], instance segmentation [5, 6, 14], and pose estimation [26, 28].

Conceptually, DETR [1] is composed of three parts: a backbone, a Transformer encoder, and a Transformer decoder. Many research works have been improving the backbone and decoder parts. For example, the backbone in DETR is normally inherited and can largely benefit from a pre-trained classification model [10, 20]. The decoder part in DETR is the major research focus, with many research works trying to introduce proper structure to DETR query and improve its training efficiency [11, 13, 18, 21, 36, 37]. By con-

*This work was done when Feng Li was an intern at IDEA.
†Corresponding author.

trast, much less work has been done to improve the encoder part. The encoder in vanilla DETR includes six Transformer encoder layers, stacked on top of a backbone to improve its feature representation. Compared with classical detection models, it lacks multi-scale features, which are of vital importance for object detection, especially for detecting small objects [9, 16, 19, 22, 29]. Simply applying Transformer encoder layers on multi-scale features is not practical due to the prohibitive computational cost that is quadratic to the number of feature tokens. For example, DETR uses the C5 feature map, which is 1/32 of the input image resolution, to apply the Transformer encoder. If a C3 feature (1/8 scale) is included in the multi-scale features, the number of tokens from this scale alone will be 16 times of the tokens from the C5 feature map. The computational cost of self-attention in Transformer will be 256 times high.

To address this problem, Deformable DETR [37] develops a deformable attention algorithm to reduce the self-attention complexity from quadratic to linear by comparing each query token with only a fixed number of sampling points. Based on this efficient self-attention computation, Deformable DETR introduces multi-scale features to DETR, and the deformable encoder has been widely adopted in subsequent DETR-like models [11, 13, 18, 36].

However, due to a large number of query tokens introduced from multi-scale features, the deformable encoder still suffers from a high computational cost. To reveal this problem, we conduct some analytic experiments as shown in Table 1 and 2 using a DETR-based model DINO [36] to analyze the performance bottleneck of multi-scale features. Some interesting results can be observed. First, the low-level (high-resolution map) features account for more than 75% of all tokens. Second, direct dropping some low-level features (DINO-3scale) mainly affects the detection performance for small objects (AP_S) by a 10% drop but has little impact on large objects (AP_L).

Inspired by the above observations, we are keen to address a question: *can we use fewer feature scales but maintain important local details?* Taking advantage of the structured multi-scale features, we present an efficient DETR framework, named Lite DETR . Specifically, we design a simple yet effective encoder block including several deformable self-attention layers, which can be plug-and-play in any multi-scale DETR-base models to reduce $62\% \sim 78\%$ encoder GFLOPs and maintain competitive performance. The encoder block splits the multi-scale features into high-level features (e.g., C6, C5, C4) and low-level features (e.g., C3). High-level and low-level features will be updated in an interleaved way to improve the multi-scale feature pyramid. That is, in the first few layers, we let the high-level features query all feature maps and improve their representations, but keep low-level tokens intact. Such a strategy can effectively reduce the number of query tokens to $5\% \sim 25\%$ of the original tokens and save a great amount of computational cost. At the end of the encoder block, we let low-level tokens query all feature maps to update their representations, thus maintaining multi-scale features. In this interleaved way, we update high-level and low-level features in different frequencies for efficient computation.

Moreover, to enhance the lagged low-level feature update, we propose a key-aware deformable attention (KDA) approach to replacing all attention layers. When performing deformable attention, for each query, it samples both keys and values from the same sampling locations in a feature map. Then, it can compute more reliable attention weights by comparing the query with the sampled keys. Such an approach can also be regarded as an extended deformable attention or a sparse version of dense attention. We have found KDA very effective in bringing the performance back with our proposed efficient encoder block.

To summarize, our contributions are as follows.

- We propose an efficient encoder block to update high-level and low-level features in an interleaved way, which can significantly reduce the feature tokens for efficient detection. This encoder can be easily plugged into existing DETR-based models.
- To enhance the lagged feature update, we introduce a key-aware deformable attention for more reliable attention weights prediction.
- Comprehensive experiments show that Lite DETR can reduce the detection head GFLOPs by 60% and maintain 99% detection performance. Specifically, our Lite-DINO-SwinT achieves 53.9 AP with 159 GFLOPs.

2. Related Work

Preliminary: DETR [1] regards object detection as a direct set prediction problem and uses a set-based global loss to force unique predictions via bipartite matching. Vanilla DETR [1] only uses single-scale features from the last stage of the backbone ($\frac{1}{32}$ of the input image resolution), i.e., $X_{feat} \in R^{N \times D}$, where D is the feature dimension and N is the total number of flattened features. These features will then be processed by the encoder with dense self-attention layers for feature fusion and enhancement. The use of an encoder is similar to FPN [16] in CNN-based models. These refined features will be queried by the decoder to detect objects by predicting classification scores and regressing bounding boxes. In general, as DETR only uses high-level features that are of low resolution, these features lack rich local details that are critical for small object detection.

Improving Decoder Design of DETR: Recently, DETR-based detectors have seen more rapid progress [13, 18, 21, 33, 36] compared to classical detectors [2, 24]. As a result, DINO [36] achieved first place in COCO 2017 object detection for the first time as a DETR-like model. Most works focus

Model	Total GFLOPs	Backbone	(M.S.) Encoder	Decoder	Total Train Mem	AP	AP_s	AP_L
DINO-4scale (100%)	235	70	**137**	28	32G	50.7	**33.5**	64.7
DINO-3scale (25%)	122	70	31	21	13G	48.2	**30.1**	63.9

Table 1. GFLOPs of DINO based on ResNet-50 with four feature scales and three feature scales, respectively. We use ResNet-50 as the backbone and evaluate on COCO *val2017* trained with 12 epochs. 100% means we use all the feature tokens, while 25% means we use three high-level features, which accounts for 25% of all tokens.

on improving the Transformer decoder in DETR for better performance and faster convergence speed. Specifically, Meng *et al.* [21] propose to decouple content and positional information in the decoder to provide better spatial priors in localization. [18, 37] further design better formulations of positional queries than previous works. The one-to-one label-assignment scheme is also widely discussed in [4, 11, 13, 36] for a better assignment. Moreover, some models design [33, 36, 37] better decoder query initialization by utilizing dense priors from the encoder.

Improving Multi-Scale Feature Extraction of DETR: Though DETR-based models with multi-scale features have shown promising performance [36, 37], especially for small object detection, their efficiency is still a concern for many applications. In fact, multi-scale feature extraction has been widely studied in many CNN-based detectors for efficiency and effectiveness, such as FPN [16], BiFPN [29], PANET [19], and NAS-FPN [9], yet the efficiency of multi-scale DETR is under-explored. Recently, a few works [27, 34, 35, 37] have attempted to design efficient encoders.

Deformable DETR [37] proposes deformable attention, which can be used in the DETR encoder to sparsify the values in a self-attention layer by sampling only a few values for each query. The proposed deformable encoder leads to good detection results with an affordable computation cost, which has been widely acknowledged and applied in many vision tasks. However, compared with single-scale detectors, the computation cost of multi-scale deformable DETR is still high for efficient usage. Based on the strong deformable encoder, some works attempt to improve its efficiency. Efficient DETR [33] proposes to use *fewer encoder layers* by leveraging encoder dense priors for decoder query initialization. Sparse DETR [25] proposes to sparsely update salient tokens in the encoder to *reduce the number of queries* with a scoring network. In fact, the encoder is responsible for feature extraction, but Sparse DETR introduces multi-layer detection loss in encoder layers, making it hard to generalize to other DETR-based models.

Recently, DETR++ [34] proposes to replace the encoder with BiFPN [29] and VIDT [27] develops a stronger decoder to remove the encoder. IMFA [35] proposes to sample sparse scale-adaptive features from some interesting areas of multi-scale features. However, the performance of these models still largely lags behind improved detectors [13, 36] based on the deformable encoder.

Feature Scale (S)	S1 ($\frac{1}{64}$)	S2 ($\frac{1}{32}$)	S3 ($\frac{1}{16}$)	S4 ($\frac{1}{8}$)
Token Ratio	1.17%	4.71%	18.8%	75.3%

Table 2. The token ratio of each feature scale in a 4-scale feature pyramid.

3. Method

3.1. Motivation and Analysis

In this part, we first analyze why existing DETR-based models are still inefficient and then show some interesting observations. Multi-scale features are of vital importance for detecting objects of diverse scales. They are composed of multiple feature scales ranging from high-level (low resolution) to low-level (high resolution) features. Each lower-level feature map contains $4\times$ more tokens than its previous feature level. From Table 2, we can observe that the number of tokens in low-level features quadratically increases, whereas the three higher-level scales account for only about 25%.

Furthermore, we take a DETR variant DINO [36] as a preliminary example. What will happen if we simply drop the low-level feature (S4 in Table 2) in its deformable encoder to reduce the computational costs? In Table 1, a reduced DINO-3scale model trades a **48%** efficiency gain in terms of GFLOPs at the cost of a **4.9%** average precision (AP) and even **10.2%** AP on small object detection deterioration. However, the AP on large objects is competitive. That is, high-level tokens contain compact information and rich semantics to detect most objects. By contrast, a large number of low-level tokens are mainly responsible for local details to detect small objects. Meanwhile, multi-scale features contain many redundant tokens, especially low-level features. Therefore, we would like to explore *how to efficiently update multi-scale features by primarily focusing on constructing better high-level features.*

In this way, we can prioritize high-level feature updates in most layers, which could significantly reduce query tokens for a more efficient multi-scale encoder. To sum up, this work aims to design a general solution for highly efficient DETR-based detectors and maintain competitive performance.

3.2. Model Overview

Following the multi-scale deformable DETR [37], Lite DETR is composed of a backbone, a multi-layer encoder, and a multi-layer decoder with prediction heads. The overall model framework is shown in Fig. 2. Specifically, we split

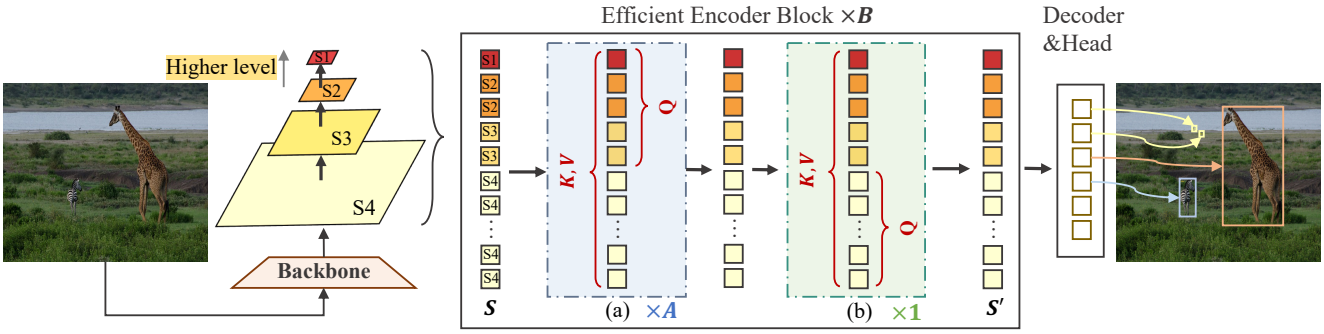

Figure 2. Illustration of the Lite DETR framework. We use $S_2 \sim S_4$ to indicate the features from different backbone stages. That is, they correspond to $C_5 \sim C3$ in ResNet-50 [10]. S_1 is acquired by further downsampling C_5 by a ratio of 0.5. In this figure, we take $S1 \sim S3$ as high-level features as an example. Moreover, (a) is the proposed high-level feature update discussed in Sec. 3.4 and (b) is the low-level feature cross-scale fusion discussed in Sec. 3.5. In each efficient encoder block, the multi-scale features will go through high-level feature update for A times and then conduct low-level feature update at the end of each block. The efficient encoder block will perform B times.

the multi-scale features from a backbone into high-level features and low-level features. These features will be updated in an interleaved manner (introduced in Sec. 3.3) with different updating frequencies (explained in Sec. 3.4 and 3.5) in the proposed efficient encoder block to achieve precision and efficiency trade-off. To enhance the lagged update of low-level features, we further introduce a key-aware deformable attention (KDA) approach (described in Sec. 3.6).

3.3. Interleaved Update

From our motivation, the bottleneck towards an efficient encoder is excessive low-level features, where most of which are not informative but contain local details for small objects. Moreover, multi-scale features S are structured in nature, where the small number of high-level features encodes rich semantics but lack important local features for some small objects. Therefore, we propose to prioritize different scales of the features in an interleaved manner to achieve a precision and efficiency trade-off. We split S into low-level features $F_L \in \mathbb{R}^{N_L \times d_{model}}$ and high-level features $F_H \in \mathbb{R}^{N_H \times d_{model}}$, where d_{model} is the channel dimension, and N_H and N_L are the corresponding token number ($N_H \approx 6\% \sim 33\% N_L$). F_H can contain the first three or two scales in different settings, for clarity, we set F_H to $S1, S2, S3$ and F_L to $S4$ by default. F_H is regarded as the primary feature and is updated more frequently, whereas F_L is updated less frequently. As deformable attention has a linear complexity with feature queries, the small number of frequently-updated high-level features largely reduces the computational cost. As shown in Fig. 2, we stack the efficient encoder block for B times, where each block updates high-level features for A times but only updates low-level features once at the end of the block. In this way, we can maintain a full-scale feature pyramid with a much lower computation cost. With this interleaved update, we design two effective updating mechanisms for F_L and F_H.

3.4. Iterative High-level Feature Cross-Scale Fusion

In this module, the high-level features F_H will serve as queries (**Q**) to extract features from *all-scales*, including the low-level and high-level feature tokens. This operation enhances the representation of F_H with both high-level semantics and high-resolution details. The detailed updating process is shown in Fig. 2(a). This operation is highly efficient. For example, using multi-scale feature queries in the first two scales or the first three scales will significantly reduce 94.1% and 75.3% queries, respectively, as shown in Table 2. We also use the proposed key-aware attention module KDA, which will be discussed in Sec 3.6, to perform attention and update tokens. Formally, the update process can be described as

$$\begin{aligned} \mathbf{Q} = F_H,\ \mathbf{K} = \mathbf{V} &= Concat(F_H, F_L) \\ F'_H &= KDA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ Output &= Concat(F'_H, F_L) \end{aligned} \tag{1}$$

where $Concat$ is to concatenate low-level and high-level features into *full-scale* features, query **Q** is the initial high-level features, **K** and **V** are initial features from all levels, and F_H is the high-level tokens, and F'_H are the updated high-level features.

A high-level feature update layer will be stacked for multiple (e.g., A times) layers for iterative feature extraction. Note that the updated F'_H will also update **Q** and the corresponding high-level features in the multi-scale feature pyramid iteratively, which makes a feature update in **K** and **V** in the next layer. Interestingly, this high-level feature updating module is similar to the Transformer decoder, where we use a small number of high-level tokens to query their features similar to a self-attention and query a large number of low-level features similar to cross-attention.

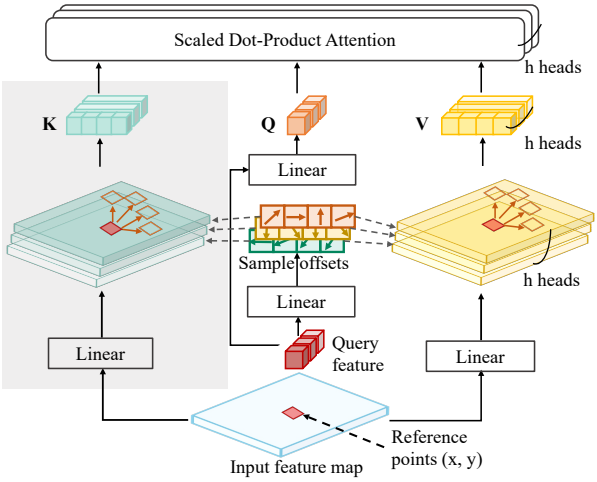

Figure 3. Illustration of the proposed Key-aware Deformable Attention (KDA) layer.

3.5. Efficient Low-level Feature Cross-Scale Fusion

As shown in Table 2, the low-level features contain excessive tokens, which is a critical factor for inefficient computation. Therefore, the efficient encoder updates these low-level features at a lower frequency after a sequence of high-level feature fusion. Specifically, we utilize the initial low-level features as queries to interact with the updated high-level tokens as well as the original low-level features to update their representation. Similar to the high-level feature update, we use the interaction with a KDA attention layer. Formally, we have

$$\begin{aligned} \mathbf{Q} = F_L,\ \mathbf{K} = \mathbf{V} = Concat(F'_H, F_L) \\ F'_L = KDA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ Output = Concat(F'_L, F'_H) \end{aligned} \tag{2}$$

where $\mathbf{Q}$ is from the original low-level features, F'_H and F'_L are the updated high-level and low-level features, respectively. After a KDA layer, we can obtain the F'_L. Finally, we construct the output multi-scale features S' by concatenating the updated low-level and high-level features. To further reduce the computational cost, we use a lightweight feed-forward network with a hidden dimension size $\frac{1}{\lambda}$ of the original size. λ is 8 in our model.

3.6. Key-aware Deformable Attention

In a vanilla deformable attention layer, the query $\mathbf{Q}$ will be split into M heads, and each head will sample K points from each of the L feature scales as value $\mathbf{V}$. Therefore, the total number of values sampled for a query is $N_v = M \times L \times K$. The sampling offsets Δp, and their corresponding attention weights are directly predicted from queries using two linear projections denoted as W^p and W^A. Deformable

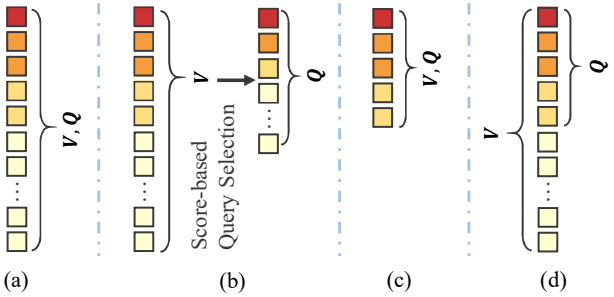

Figure 4. Comparison of previous efficient encoders strategies in (a) Deformable DETR [37], (b) Sparse DETR [25], and (c) Trivially using only the first three high-level scales. (d) Preliminary efficient encoder to only update high-level features. We also present the results of (c) and (d) in Table 5.

attention can be formulated as

$$\begin{aligned} \Delta p = \mathbf{Q}W^p, \mathbf{V} = Samp(S, p + \Delta p)W^V \\ DeformAttn(\mathbf{Q}, \mathbf{V}) = Softmax(\mathbf{Q}W^A)\mathbf{V} \end{aligned} \tag{3}$$

Where the projections are parameter matrices $W^A, W^p \in \mathbb{R}^{d_{model} \times N_v}$ and $W^V \in \mathbb{R}^{d_{model} \times d_{model}}$. p are the reference points of the query features and $\Delta p, p \in \mathbb{R}^{(N_H+N_L)\times N \times 2}$. S is the multi-scale feature pyramid. With the sampled offsets Δp, it computes the features with function $Samp(S, p + \Delta p)$ in the sampled locations $(p + \Delta p)$ of feature pyramid S with bilinear interpolation. Note that no key participates in the original deformable attention layer, indicating that a query can decide the importance of each sampled value by only its feature without comparing it with keys. As all the multi-scale features will be the queries to sample locations and attention weights, the original model can quickly learn how to evaluate the importance of each sampled location given the queries. Nevertheless, the interleaved update in our encoder makes it difficult for the queries to decide both the attention weights and sampling locations in other asynchronous feature maps, as shown in Fig. 5.

To better fit the efficient encoder designs, we propose a key-aware deformable attention (KDA) approach to sampling both keys and values for a query, as shown in Fig. 3. The sampled keys and values, together with the query, will then perform a standard scaled dot-product attention. Formally, we have

$$\begin{aligned} \mathbf{V} = Samp(S, p + \Delta p)W^V, \\ \mathbf{K} = Samp(S, p + \Delta p)W^K, \\ KDA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \end{aligned} \tag{4}$$

where d_k is the key dimension of a head. The computational complexity of KDA is the same as the original deformable attention as we sample the same number of values for each query. In this way, KDA can predict more reliable attention weights when updating features from different scales.

Model	#epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	GFLOPs	Encoder GFLOPs	Params
DETR-DC5 [1]	500	43.3	63.1	45.9	22.5	47.3	61.1	187	100	41M
Anchor DETR-DC5 [32]	50	44.2	64.7	47.5	24.7	48.2	60.6	151	70	39M
Conditional DETR-DC5 [21]	50	43.8	64.4	46.7	24.0	47.6	60.7	195	100	44M
DAB-DETR-DC5 [18]	50	44.5	65.1	47.7	25.3	48.2	62.3	202	100	44M
DN-DETR-DC5	50	46.3	66.4	49.7	26.7	50.0	64.3	202	100	44M
Deformable DETR efficient variants										
Deformable DETR† [37]	50	46.8	66.0	50.6	29.8	49.7	62.0	177	90	40M
Lite-Deformable DETR H2L2-(2+1)x3(5%, ours)	50	45.8	65.1	49.3	27.7	49.1	61.1	108	**23(↓ 74%)**	41M
Lite-Deformable DETR H3L1-(6+1)x1(25%, ours)	50	45.9	65.6	49.2	27.9	49.0	61.6	115	**30(↓ 66%)**	41M
Lite-Deformable DETR H3L1-(3+1)x2(25%, ours)	50	46.2	65.5	49.8	28.2	49.2	61.5	119	**35(↓ 61%)**	41M
Lite-Deformable DETR H3L1-(2+1)x3(25%, ours)	50	**46.7**	66.1	50.6	29.1	49.7	62.2	123	**39(↓ 57%)**	41M
Efficient DETR [33]	50	44.2	62.2	48.0	28.4	47.5	56.6	159	79	32M
Sparse DETR*-rho-0.1 [37]	50	45.3	65.8	49.3	28.4	48.3	60.1	111	24	41M
Sparse DETR*-rho-0.2 [37]	50	45.6	65.8	49.6	28.5	48.6	60.4	119	32	41M
Sparse DETR*-rho-0.3 [37]	50	46.0	65.9	49.7	29.1	49.1	60.6	127	40	41M
Sparse DETR*-rho-0.5 [37]	50	46.3	66.0	50.1	29.0	49.5	60.8	141	54	41M

Table 3. Results for single-scale DETR-based models which use a larger resolution feature map with dilation (DC5) and Deformable DETR-based models for improving efficiency. All models are based on ResNet-50. * Sparse DETR is based on an improved Deformable DETR baseline that combines the components from Efficient DETR [33]. 'rho' is the keeping ratio of encoder tokens in Sparse DETR. Value in the parenthesis indicates the percentage of our high-level tokens compared to the original features. † we adopt the result from the official Deformable DETR codebase. The meaning of different model variants is described in Sec. 4.1.

3.7. Discussion with Sparse DETR and other Efficient Variants

Another efficient way is to reduce encoder tokens by selecting salient tokens in the multi-scale features, like Sparse DETR [25]. However, there are three drawbacks to this kind of approach. First, it is hard to generalize across other DETR-based models since it breaks the structured feature organization. Second, the selected tokens via a scoring network may not be optimal due to limited and implicit supervision. Third, it introduces other components, such as multiple auxiliary encoder detection loss, to enhance its sparse encoder representation. As the encoder is responsible for feature extraction, adding detection supervision makes it difficult to apply to existing models[1].

Moreover, we illustrate the previous efficient encoders and the preliminary efficient designs in Figure 4 for a clear comparison.

4. Experiments

4.1. Setup

We demonstrate the generalization capability of our proposed efficient encoder on a series of DETR-based models. We also evaluate the effectiveness of each component with ablations.

Datasets: We study Lite DETR on the challenging MS COCO 2017 [17] detection dataset. Following the common practice, we train on the training split and report the detection performance on the validation split **val2017**. We report the standard mean average precision (AP) result under different IoU thresholds and object scales.

Implementation details: We evaluate the performance of Lite DETR on multiple DETR-based models, including Deformable DETR [37], H-DETR [11], and DINO [36]. These models share a similar structure that is composed of a backbone, a multi-layer Transformer encoder, and a multi-layer Transformer decoder. Therefore, we simply replace their encoder with our proposed efficient module. Other model components are kept the same as the original model. In our KDA attention, for a fair comparison, we follow deformable attention to use M=8 and K=4. Other settings follow the original models. We use two backbones ResNet-50 [10] and Swin-T [20] pre-trained on the ImageNet-1K [7] dataset in our experiments.

Efficient encoder variants: In our proposed efficient encoder block, three hyperparameters control the computational cost, including the number of high-level feature scales H used in F_H, the number of efficient encoder blocks B, and the number of iterative high-level feature cross-scale fusion A. Therefore, we use HL-$(A+1)\times B$ to denote each variant of our Lite DETR , where L is the number of low-level feature scales, and +1 denotes the default efficient low-level cross-scale feature fusion at the end of each block . For example, Lite-DINO H3L1-(3+1)×2 indicates we base on DINO to use three high-level feature scales (H3L1) and two efficient encoder blocks with three high-level fusion ((3+1)×2).

4.2. Efficiency Improvements on Deformable DETR

In Table 3, we use our proposed lite encoder to replace the deformable encoder in Deformable DETR and build Lite-

[1] In our experiments on DINO [36] with a ResNet-50 backbone, adding the encoder detection loss alone will cause a **1.4** AP drop.

Model	#epochs	AP	AP_{36}	AP_{75}	AP_S	AP_M	AP_L	GFLOPs	Encoder GFLOPs	Params
EfficientDet-D6 [29]	–	51.3	–	–	–	–	–	226	–	52M
YOLOv5-X [12]	–	50.7	–	–	–	–	–	206	–	87M
YOLOv7-X [30]	–	52.9	–	–	–	–	–	190	–	71M
Swin-T backbone										
VIDT+ [27]	50	49.7	67.7	54.2	31.6	53.4	65.9	–	–	38M
D^2ETR [15]	50	49.1	–	–	–	–	–	127	–	46M
DINO [36]	36	54.1	72.0	59.3	38.3	57.3	68.6	243	137	47M
Lite-DINO H2L2-(2+1)x3(5%, ours)	36	53.1	71.4	57.9	36.6	56.0	68.8	138	**30(↓78%)**	47M
Lite-DINO H3L1-(6+1)x1(25%, ours)	36	53.3	71.7	58.2	36.3	56.6	68.7	149	**41(↓70%)**	47M
Lite-DINO H3L1-(2+1)x3(25%, ours)	36	**53.9**	72.0	58.8	37.9	57.0	69.1	159	**53(↓62%)**	47M
H-DETR [11]	36	53.2	71.5	58.2	35.9	56.4	68.2	234	137	47M
Lite-H-DETR H2L2-(2+1)x3(5%, ours)	36	52.3	70.7	57.2	35.9	55.2	67.7	131	30	47M
Lite-H-DETR H3L1-(6+1)x1(25%, ours)	36	52.7	71.5	58.3	35.6	56.0	68.0	142	41	47M
Lite-H-DETR H3L1-(2+1)x3(25%, ours)	36	**53.0**	71.3	58.2	36.3	56.3	68.1	152	53	47M
ResNet-50 backbone										
DFFT [3]	36	46.0	–	–	–	–	–	101	18	–
PnP-DETR [31]	36	43.1	63.4	45.3	22.7	46.5	61.1	104	29	–
AdaMixer [8]	36	47.0	66.0	51.1	30.1	50.2	61.8	132	–	135M
IMFA-DETR [35]	36	45.5	45.0	49.3	27.3	48.3	61.6	108	≈ 20	53M
DINO [36]	36	50.7	68.6	55.4	33.5	54.0	64.8	235	137	47M
Lite-DINO H2L2-(2+1)x3 (ours)	36	49.9	68.2	54.6	32.3	52.9	64.7	130	30	47M
Lite-DINO H3L1-(6+1)x1(ours)	36	50.2	68.6	54.3	33.0	53.4	66.0	141	41	47M
Lite-DINO H3L1-(2+1)x3(ours)	36	**50.4**	68.5	54.6	33.5	53.6	65.5	151	53	47M
H-DETR [11]	36	50.0	68.3	54.4	32.9	52.7	65.3	226	137	47M
Lite-H-DETR H3L1-(2+1)x3 (ours)	36	49.5	67.6	53.9	32.0	52.8	64.0	142	53	47M

Table 4. Results for Deformable DETR-based models to improve efficiency with our light encoder design. We also compare with some efficient CNN-based models and other efficient DETR-based models. All models except EfficientDet and YOLO series are based on ResNet-50 and Swin-T pre-trained on ImageNet-1K. Percentage in the model name indicates the percentage of our compressed tokens compared to the original features. The meaning of different model variants is described in Sec. 4.1.

Deformable DETR. We achieve comparable performance as Deformable DETR with around 40% of the original encoder GFLOPs. We can also observe that DETR-based models with a single scale of larger feature maps are computationally inefficient and inferior to multi-scale models. In iterative high-level cross-scale fusion, we can effectively adopt high-level maps with only two or three high-level maps, which can reduce the queries in an encoder layer to $5\% \sim 25\%$ of the original tokens. Compared with other efficient variants based on Deformable DETR, we achieve better performance under the same computational cost. For example, we outperform Sparse DETR-rho-0.3 by 0.7 AP with fewer GFLOPs. In addition, Sparse DETR is based on an improved baseline that combines Efficient DETR and Deformable DETR. By contrast, our Lite-Deformable DETR is simple and effective.

4.3. Efficiency Improvements on Other DETR-based Models

Compared with other efficient variants, our efficient design is not constrained to a specific detection framework and can be easily plugged into other DETR-based models. We take DINO [36] and H-DETR [11] as examples to show the effectiveness of our efficient encoder. The results are shown in Table 4. Compared with other recently proposed efficient DETR-like models [8, 35], our model achieves significantly better performance with comparable computational cost. In addition, after plugging in our efficient encoder, the encoder GFLOPs can be reduced by $78\% \sim 62\%$ compared to the original ones while keeping 99% of the original performance. Specifically, based on Swin-Tiny, our Lite-DINO achieves **53.9** AP with only 159 GFLOPs, which also outperforms YOLO series models [12, 30] under the same GFLOPs.

4.4. Visualization of KDA

We also provide the visualization of our KDA attention in our interleaved encoder in Fig. 5. Compared with deformable attention, as we introduce keys, our KDA attention can predict more reliable weights, especially on low-level feature maps. For example, in Fig. 5(a), the sampled locations of deformable attention in S4 (denoted with triangles) are less reliable compared to KDA. In Fig. 5(b) and (c), we observe that it is difficult for deformable attention to focus on meaningful regions on the largest scale map S4 in our interleaved encoder. KDA effectively mitigates this phenomenon, which helps extract better local features to bring the performance of small objects back.

4.5. Ablation Studies

Effectiveness of each proposed component. In Table 5, we show the effectiveness of our proposed components. We choose DINO-3scale and DINO-2scale as our baseline, which only uses the first three and two high-level feature

(a) Attention map of a single query on all scales (b) Attention map of all queries on scale S3 (c) Attention map of all queries on scale S4

Figure 5. Visualization of KDA attention in our interleaved encoder. The first and second row are the attention maps of using deformable and our KDA attention. (a) We use the the center of an object from S1 (marked with "+" in green) as query and draw top 100 sampling locations on all four scales according to their attention weights. The sampling locations on S4 are marked with a triangle shape. (b)&(c) We show top 200 sampling locations on scale S3 (b) and S4 (c) for all query tokens. The visualization shows that KDA can produce more reliable attention weights on high-resolution maps. For clarity, we only draw the locations of top 200 attention weights out of all sampling locations ($N_q \times M \times K$, N_q is the total number of multi-scale query tokens) on S3 and on S4. More visualizations are shown in Appendix.

	HL	LL	KDA	AP ↑	AP_s ↑	GFLOPs ↓	Encoder GFLOPs ↓
DINO-4scale [36]				50.7	33.5	235	137
3scale	–	–	–	48.2	30.1	122	31
	–	–	✓	49.0(**+0.8**)	31.5	125	34
	✓	–	–	49.0(**+0.8**)	31.1	128	37
	✓	✓	–	49.8(**+0.8**)	33.0	147	49
	✓	✓	✓	50.4(**+0.6**)	**33.5**	151	53
2scale	–	–	–	45.2	24.1	113	14
	✓	✓	–	49.2(**+4.0**)	31.8	126	26
	✓	✓	✓	49.9(**+0.7**)	**32.3**	130	30

Table 5. Effectiveness of each component on COCO *val2017* trained with 36 epochs. The results are based on DINO with a ResNet-50 backbone trained for 36 epochs. HL means iterative high-level feature cross-scale fusion, LL means efficient low-level feature cross-scale fusion, and KDA is key-aware deformable attention.

Model	AP↑	AP_s ↑	Encoder GFLOPs ↓
Deformable DETR-4scale [37]	46.8	29.8	90
Deformable DETR-2scale	40.3	20.4	9
Lite-Deformable DETR H2L2-(2+1)x3 (ours)	45.8(**+5.5**)	27.7(**+7.3**)	23
Deformable DETR-3scale	44.0	26.6	16
Lite-Deformable DETR H2L2-(6+1)x1 (ours)	45.9	27.9	28
Lite-Deformable DETR H3L1-(3+1)x2(ours)	46.2	28.2	32
Lite-Deformable DETR H3L1-(2+1)x3(ours)	**46.7**(**+2.7**)	**29.1**(**+2.5**)	36
Lite-Deformable DETR H3L1-(2+1)x4(ours)	46.6	29.6	50

Table 6. Ablation study on stacking different number of each module in our efficient encoder block. All the models are built upon Deformable DETR-ResNet50 and trained for 50 epochs.

maps. The results indicate that each of our proposed components requires a small computational cost while improving the model performance by a decent margin. KDA mainly helps improve DINO performance. Specifically, these components effectively bring the performance on small objects back, for example, the AP_s of our efficient DINO-3scale is comparable with the original DINO-4scale model.

Influence of stacking the different number of modules. In Table 6, we explore the optimal choice to stack each module in our proposed efficient block. Based on Deformable DETR [37] with a ResNet-50 backbone, we vary three arguments that influence the computational complexity and detection performance, including the number of high-level scales H used as high-level features, efficient encoder block B, and iterative high-level feature cross-scale fusion A. The performance improves when we use more high-level feature scales and more encoder blocks to update the low-level features. However, further increasing the module number to $(2+1) \times 4$ will not improve the performance.

5. Conclusion

In this paper, we have analyzed that multi-scale features with excessive low-level features in the Transformer encoder are the primary cause of the inefficient computation in DETR-based models. We have presented Lite DETR with an efficient encoder block, which splits the encoder tokens into high-level and low-level features. These features will be updated in different frequency with cross-scale fusion to achieve precision and efficiency trade-off. To mitigate the effects of asynchronous feature, we further proposed a key-aware deformable attention, which effectively brings the detection performance of small objects back. As a result, our proposed efficient encoder can reduce computational cost by 60% while keeping 99% of the original performance. In addition, this efficient design can be easily plugged into many DETR-based detection models. We hope Lite DETR can provide a simple baseline for efficient detection in DETR-based models to benefit other resource-constrained applications.

Limitations: In this paper we mainly focus on reducing the computational complexity and do not optimize the run-time implementation of DETR-based model. We leave this to our future work.

References

[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*, pages 213–229. Springer, 2020. 1, 2, 6

[2] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 2

[3] 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*, 2022. 7

[4] Qiang Chen, Xiaokang Chen, Gang Zeng, and Jingdong Wang. Group DETR: Fast Training Convergence with Decoupled One-to-Many Label Assignment. *arXiv preprint arXiv:2207.13085*, 2022. 3

[5] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. 2022. 1

[6] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation. 2021. 1

[7] 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*, pages 248–255. Ieee, 2009. 6

[8] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. AdaMixer: A Fast-Converging Query-Based Object Detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5364–5373, 2022. 7

[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 2, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 4, 6

[11] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. DETRs with Hybrid Matching. *arXiv preprint arXiv:2207.13080*, 2022. 1, 2, 3, 6, 7

[12] Glenn Jocher, K Nishimura, T Mineeva, and R Vilariño. YoloV5, url=https://github.com/ultralytics/yolov5, 2021. 1, 7

[13] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. DN-DETR: Accelerate DETR Training by Introducing Query DeNoising. *arXiv preprint arXiv:2203.01305*, 2022. 1, 2, 3

[14] Feng Li, Hao Zhang, Shilong Liu, Lei Zhang, Lionel M Ni, Heung-Yeung Shum, et al. Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation. *arXiv preprint arXiv:2206.02777*, 2022. 1

[15] Junyu Lin, Xiaofeng Mao, Yuefeng Chen, Lei Xu, Yuan He, and Hui Xue. Dˆ 2ETR: Decoder-Only DETR with Computationally Efficient Cross-Scale Attention. *arXiv preprint arXiv:2203.00860*, 2022. 7

[16] 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*, pages 2117–2125, 2017. 2, 3

[17] 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*, pages 740–755. Springer, 2014. 6

[18] 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. 1, 2, 3, 6

[19] 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*, pages 8759–8768, 2018. 2, 3

[20] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 6

[21] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for Fast Training Convergence. *arXiv preprint arXiv:2108.06152*, 2021. 1, 2, 3, 6

[22] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021. 2

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1

[24] 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

[25] 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. 3, 5, 6

[26] Dahu Shi, Xing Wei, Liangqi Li, Ye Ren, and Wenming Tan. End-to-End Multi-Person Pose Estimation With Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11069–11078, 2022. 1

[27] Hwanjun Song, Deqing Sun, Sanghyuk Chun, Varun Jampani, Dongyoon Han, Byeongho Heo, Wonjae Kim, and Ming-Hsuan Yang. An Extendable, Efficient and Effective Transformer-based Object Detector. *arXiv preprint arXiv:2204.07962*, 2022. 3, 7

[28] Lucas Stoffl, Maxime Vidal, and Alexander Mathis. End-to-end trainable multi-instance pose estimation with transformers. *arXiv preprint arXiv:2103.12115*, 2021. 1

[29] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 1, 2, 3, 7

[30] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022. 1, 7

[31] 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*, pages 4661–4670, 2021. 7

[32] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. *arXiv preprint arXiv:2109.07107*, 2021. 6

[33] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient DETR: Improving End-to-End Object Detector with Dense Prior. *arXiv preprint arXiv:2104.01318*, 2021. 2, 3, 6

[34] Chi Zhang, Lijuan Liu, Xiaoxue Zang, Frederick Liu, Hao Zhang, Xinying Song, and Jindong Chen. DETR++: Taming Your Multi-Scale Detection Transformer. *arXiv preprint arXiv:2206.02977*, 2022. 3

[35] Gongjie Zhang, Zhipeng Luo, Yingchen Yu, Zichen Tian, Jingyi Zhang, and Shijian Lu. Towards Efficient Use of Multi-Scale Features in Transformer-Based Object Detectors. *arXiv preprint arXiv:2208.11356*, 2022. 3, 7

[36] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. *arXiv preprint arXiv:2203.03605*, 2022. 1, 2, 3, 6, 7, 8

[37] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 2021. 1, 2, 3, 5, 6, 8