Exploring the Feature Extraction and Relation Modeling For Light-Weight Transformer Tracking

Jikai Zheng[†1,4], Mingjiang Liang[†2], Shaoli Huang[3], and Jifeng Ning[*1,4]

[1] College of Information Engineering, Northwest A&F University, Shaanxi, China
[2] University of Technology Sydney, Sydney, Australia
[3] Tencent AI-Lab, Guangdong, China
[4] Shaanxi Engineering Research Center of Agricultural Information Intelligent Perception and Analysis, Shaanxi, China
jikai_zheng@163.com, mingjiang.liang@student.uts.edu.au, shaolihuang@tencent.com, jf_ning@sina.com

Abstract. Recent advancements in transformer-based lightweight object tracking have set new standards across various benchmarks due to their efficiency and effectiveness. Despite these achievements, most current trackers rely heavily on pre-existing object detection architectures without optimizing the backbone network to leverage the unique demands of object tracking. Addressing this gap, we introduce the Feature Extraction and Relation Modeling Tracker (FERMT) - a novel approach that significantly enhances tracking speed and accuracy. At the heart of FERMT is a strategic decomposition of the conventional attention mechanism into four distinct sub-modules within a one-stream tracker. This design stems from our insight that the initial layers of a tracking network should prioritize feature extraction, whereas the deeper layers should focus on relation modeling between objects. Consequently, we propose an innovative, lightweight backbone specifically tailored for object tracking. Our approach is validated through meticulous ablation studies, confirming the effectiveness of our architectural decisions. Furthermore, FERMT incorporates a Dual Attention Unit for feature preprocessing, which facilitates global feature interaction across channels and enriches feature representation with attention cues. Benchmarking on GOT-10k, FERMT achieves a groundbreaking Average Overlap (AO) score of 69.6%, outperforming the leading real-time trackers by 5.6% in accuracy while boasting a 54% improvement in CPU tracking speed. This work not only sets a new standard for state-of-the-art (SOTA) performance in light-weight tracking but also bridges the efficiency gap between fast and high-performance trackers. The code and models are available at https://github.com/KarlesZheng/FERMT.

Keywords: Light-weight Object Tracking · Transformer · Feature extraction · relation modeling

[†]Equal Contribution.
[*]Corresponding author.

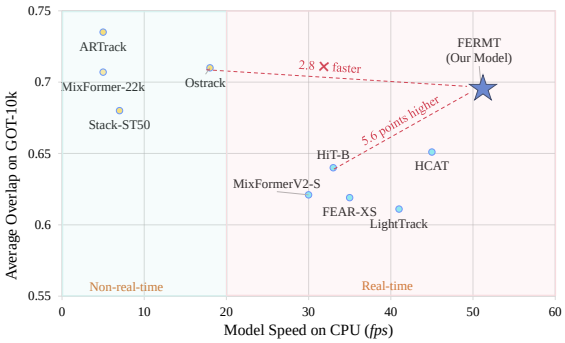

Fig. 1: Comparison of our FERMT with other trackers on GOT-10k in terms of speed (horizontal axis) on the Inter Core I9-9900k CPU and Average Overlap(vertical axis). We set the real-time tracking line at 20*fps*, and FERMT achieved state-of-the-art (SOTA) results among real-time trackers.

1 Introduction

With the development of deep learning [17,19,20] in recent years, object tracking has made great progress, especially the use of Transformer plays a big role in it. However, it is worth noting that most object tracking models only focus on high performance but ignore tracking speed. In real scenarios with limited device resources, the tracking speed of these models is difficult to achieve fast responses. Many works on light-weight object tracking and models have been proposed [2, 3, 5, 9, 18, 42], however, a significant performance disparity remains between these efficient trackers and the mainstream high-performance trackers.

Among previous light-weight trackers, trackers using CNN architecture [11, 42] often have the advantages of small model size and fast tracking, but there is a common problem of insufficient interaction between template and search features, which leads poor tracking performance. Later Transformer-based light-weight trackers [2, 5] used cross-attention to obtain global interaction between template and search features, effectively improving tracking performance. However, they often only use an existing light-weight backbone network such as LeViT [16], and do not conduct further exploration into the extraction and interaction of the template and search image features in object tracking, which makes the model backbone network parameters volume decline is limited.

In order to overcome the above problems, we decompose in detail the attention module [21,38] for feature extraction and relation modeling in the tracking process, and designed a novel light-weight tracking model within an one-stream architecture [46] to obvious decrease computation cost and simultaneously improve tracking accuracy. In experiments, we found that the feature extraction module in the deep layers of the attention model may be redundant for object tracking, which leads our solution method, i.e., pruning feature extraction in deep layers is useful to speed up tracking and optimize the tracking effect. In addition, in order to alleviate the weakness of light-weight network feature extraction, we design a CNN [7, 47] based Dual Attention Unit to enhance feature representation before images are fed into the transformer backbone network, which further improve the tracking performance by adding a few parameters.

Based on the above work, we proposed an light-weight tracking model FERMT, and verified the high performance and efficiency of FERMT through a large number of experiments. Compared with HiT [18], the current best real-time tracking model, FERMT improved by 5.6% on the GOT-10k, and also performed better than it on several other conventional benchmarks. In addition, FERMT's tracking speed on CPU is improved by 54% compared to HiT, while using fewer parameters (2.40G *v.s.* 4.34G) and fewer Flops (7.98M *v.s.* 42.14M). Compared with high-performance trackers, FERMT is able to achieve similar performance to Ostrack [46], but runs 2.8 times faster.

Our main contributions are summarized as follows:

(1) We conducted a detailed exploration on the role of feature extraction and relation modeling in object tracking, and designed a new way of feature fusion where feature extraction is no longer performed in the deep layers of the backbone network. Instead, it solely relies on template frames and search frames to interact with features, reducing the computational load of feature fusion. This, in turn, enables faster tracking speed without compromising performance.

(2) The feature extraction capability of light-weight models has declined, especially the information interaction at the channel layer is insufficient. Therefore, we proposed a dual attention module Dual Attention Unit (DAU), which interacts with features from different channels in the image and incorporates attention information to enhance the expressive capability of the features.

(3) Based on the above work, we propose an one-stream efficient object tracker based on a light-weight pre-trained backbone network. It reaches the SOTA of real-time tracking models in multiple benchmarks including GOT-10k, and performs excellently in terms of tracking speed.

2 Related Work

Visual Tracking based on Transformer. In the past object tracking research, trackers usually use a set of parameter-sharing Siamese networks [22, 23] for feature extraction, and then perform convolution operations on the template features and search features to complete the object prediction. In recent years, with the rise of Transformer in the field of computer vision, models such as TransT [6], STARK [43], and SwinTrack [25], after using CNN [19, 35] to extract image features, use Transformer [38] to interact with features of template frame images and search frame images, improving the tracking performance of the model. Recently, the one-stream [4] architecture tracking model has demonstrated strong competitiveness.Among them, Ostrack [46] first abandons the traditional two-stream tracking architecture and performs feature extraction and relation modeling in parallel. Mixformer [8] has further discussed that it uses asymmetric interaction in the backbone network to improve tracking performance while reducing the amount of calculations. However, although these models have good tracking accuracy, they often have a large number of parameters and cannot demonstrate their advantages in a resource-constrained environment.

Vision Transformer. Since ViT [12] brought Transformer into the field of computer vision, more and more Transformer-based visual networks [27, 33, 37, 40] have appeared, and they have achieved impressive results in fields such as classification and object detection. Among many models, there are some light-weight visual Transformer models [16, 30, 49] that have advantages in speed. In addition to reducing feature dimensions and the number of model layers, most of these light-weight vision transformer models adopt a pyramid structure of layer-by-layer downsampling to reduce the number of model parameters and increase speed. In this work, we used the pre-trained model ViT-Tiny [39], a light-weight Transformer model with a non-pyramid structure as the backbone network. ViT-Tiny reduces the dim of features extracted by Vit-Base [12] and uses feature distillation [44, 45, 48] to obtain a light-weight visual model that performs well in applications such as image classification.

Different from previous one-stream trackers, we use ViT-Tiny's shallow-level module to synchronize feature extraction and feature fusion, and in its deep-level module, only template frame images and search frame images are processed fusion of features.

Light-weight Tracking. In real-world scenarios, object trackers may need to run on platforms with limited computational capabilities, which places demands on their runtime speed. Early on, ECO [10] and ATOM [11] emerged as representatives of efficient trackers, meeting the requirements of real-time tracking. However, their performance across various datasets was unsatisfactory. LightTrack [42] proposed the use of NAS to construct light-weight object tracking networks, performing well on some short-term tracking datasets but falling short on long-term tracking datasets like LaSOT. Recently, several light-weight trackers have made promising progress. FEAR [3] achieved a series of efficient and accurate trackers by adopting dual-template representation and pixel-wise fusion methods. HCAT [5] effectively reduced model computation by employing a hierarchical design of cross-attention. HiT [18] utilized an one-stream architecture, using pre-trained LeViT [16] as the backbone network to design a light-weight object tracking model, showcasing excellent tracking capabilities while pursuing efficiency. However, there still exists a significant performance gap between these models and the current mainstream high-performance trackers.

In comparison to the aforementioned models, our FERMT focuses on a detailed exploration of modeling interactions between template frame images, search frame images, and their relation. We further validate the impact of feature extraction and relation modeling on object tracking at a deeper level. As a result, we design a backbone network with fewer parameters while achieving superior performance.

3 The Proposed Method

This section provides a detailed exposition of our one-stream light-weight tracking model, FERMT. Firstly, we provide a brief overview of the FERMT frame-

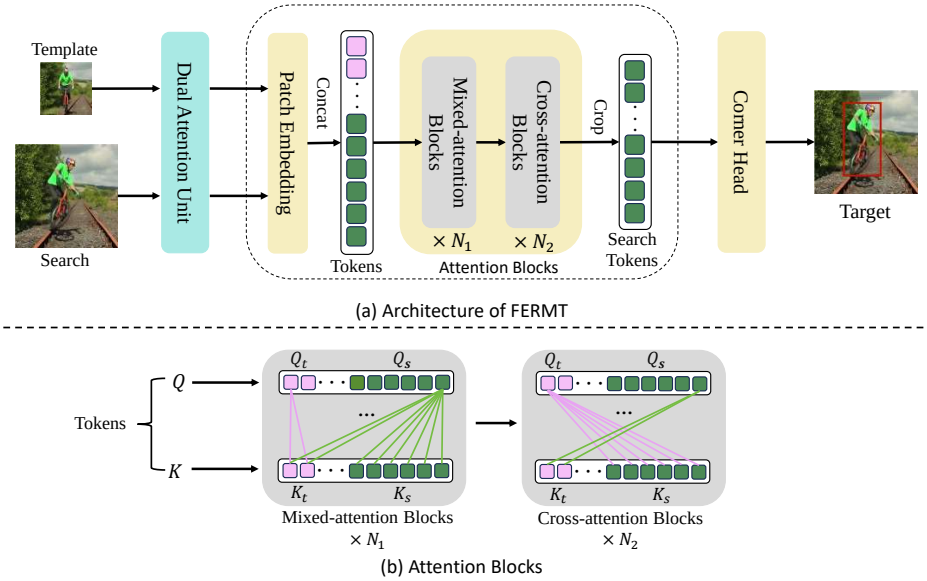

Fig. 2: The overall framework of the proposed FERMT(a). The model consists of three parts: Dual Attention Unit for image preprocessing, backbone network for feature extraction and relation modeling, and a prediction head. Figure (b) details the composition of Attention Blocks in the backbone network, where Mixed-attention Blocks are used for shallow feature operations of Attention Blocks, and Cross-attention Blocks are used for deep feature operations of Attention Blocks.

work. Next, we present the specific architecture of the model, including a novel feature extraction and interaction approach, CNN based Dual Attention Unit and the head network. Finally, we describe the training and inference processes of the model.

3.1 Overview

As shown in Figure Fig. 2(a),FERMT is an one-stream real-time tracking model, including CNN based Dual Attention Unit for image preprocessing, a light-weight visual Transformer module for feature extraction and relation modeling, and a head network for object prediction. When the template images and search images are input to the model, they are first sent to the DAU module for preprocessing to enhance their feature representation.

Then we sent the image to the visual Transformer backbone network for feature extraction and feature interaction. In this process, we designed two different types of Mixed-attention Block and Cross-attention Block for the shallow and deep layers of the network, respectively. It replaces the standard attention module [46] with high computational complexity in popular one-stream lightweight tracking [18]. Finally, we send the obtained search image features into

the prediction head [43] to predict the target position to obtain the tracking results.

3.2 Model Architecture

Feature extraction and relation modeling. In previous tracking models, the asymmetric interaction, i.e., relation modeling, proposed by Mixformer [8] not only reduces the amount of calculations, but also eliminates the information interference of the search frame on the template frame during the tracking process, thereby increasing the tracking accuracy. It indicates that attention module representing feature extraction and interaction need be explored to obtain more efficient backbone network for object tracking.

In the attention module, we let t and x represent the template frame and the search frame respectively, and generate tokens through linear changes. Then Q_t, K_t, and V_t are generated from the tokens of the template frame, and Q_s, K_s, and V_s are generated from the tokens of the search frame [21]. Let $Q = [Q_t; Q_s]$, $K = [K_t; K_s]$, $V = [V_t; V_s]$, the calculation process of attention in one-stream tracker can be expressed as Eq. (1).

$$\mathrm{A(Q,\ K,\ V)} = \mathrm{softmax}\left(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d}}}\right) \cdot \mathrm{V}. \tag{1}$$

In the following, to simplify the formula, $\sqrt{d}$ is omitted. In order to deeply explore the relation between feature extraction and interaction, we further write Eq. (1) to Eq. (2):

$$\begin{aligned} \mathrm{A}\left(\left[\mathrm{Q}_t; \mathrm{Q}_s\right], \left[\mathrm{K}_t; \mathrm{K}_s\right], \left[\mathrm{V}_t; \mathrm{V}_s\right]\right) &= \mathrm{softmax}\left(\left[\mathrm{Q}_t; \mathrm{Q}_s\right]\left[\mathrm{K}_t; \mathrm{K}_s\right]^T\right) \cdot \left[\mathrm{V}_t; \mathrm{V}_s\right] \\ &= \mathrm{softmax}\left(\begin{bmatrix} \mathrm{Q}_t\mathrm{K}_t^T & \mathrm{Q}_t\mathrm{K}_s^T \\ \mathrm{Q}_s\mathrm{K}_t^T & \mathrm{Q}_s\mathrm{K}_s^T \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V}_t \\ \mathrm{V}_s \end{bmatrix}, \end{aligned} \tag{2}$$

where $Q_tK_t^T$ and $Q_sK_s^T$ are self-attention modules, which respectively represent the self-attention of the respective internal areas of the template frame and search frame, and play the role of feature extraction; $Q_tK_s^T$ and $Q_sK_t^T$ are cross-attention modules, which respectively represent the interaction from the template frame to the search frame and the attention from the search frame to the template frame, which play the role of feature interaction, i.e., relation modeling of target template and search frame.

The calculation amount of Eq. (2) is a crucial factor in determining the tracking speed. Let the total number of tokens in the template be n, and the total number of tokens searched be m. The time complexity of $Q_tK_t^T$ and $Q_sK_s^T$ is $O(n^2)$ and $O(m^2)$, and the time complexity of $Q_tK_s^T$ and $Q_sK_t^T$ is both $O(mn)$. Generally, the image size of the search frame is much larger than that of the template frame, *i.e.*, $m >> n$, so in the calculation process of the attention module, the feature extraction module for search frame occupies a large part amount

of calculation. Therefore, if this module can be reduced, it will be beneficial to reduce the amount of parameters.

In object tracking, The key content is the interaction between template frames and search frames since its importance to estimate object location. So we believe that the feature extraction effect decreases in the later stages of the network, while the interaction effect increases. Therefore, we divide the backbone network into two stages. The first stage is based on feature extraction and supplemented by feature interaction, in the form of MixFormer [8]. The second stage only retains feature interaction and discards templates and search features extraction that require large calculations.

As shown in Fig. 2(b), the first N_1 layers Mixed-attention Blocks responsible for shallow feature processing adopts an asymmetric structure to perform feature extraction and one-way transmission of interactive information from template to search; while the second N_2 layers responsible for deep feature processing Cross-attention Blocks deletes the self-attention within template and search, and only models the relation between them.

Eq. (3) shows the attention calculation methods of two different structures in our backbone network. In the experiment, when we set $N_1 = 6$, $N_2 = 6$, and the sizes of template and search are $(128, 128)$ and $(256, 256)$, calculation of our attention module is 42% less than that of the previous architecture represented by Eq. (1). In the ablation experiments of Sec. 4.3, we verified the superiority of this structure.

$$\begin{aligned} \mathrm{A}_{\mathrm{mixed}} &= \mathrm{softmax}\left(\begin{bmatrix} \mathrm{Q_t K_t^T} & \\ \mathrm{Q_s K_t^T} & \mathrm{Q_s K_s^T} \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \\ \mathrm{A}_{\mathrm{cross}} &= \mathrm{softmax}\left(\begin{bmatrix} & \mathrm{Q_t K_s^T} \\ \mathrm{Q_s K_t^T} & \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \end{aligned} \tag{3}$$

CNN based Dual Attention Unit. After using the light-weight backbone network, the reduction of *dim* limits the feature extraction ability of Transformer. In order to compensate for this effect, inspired by the Temporal Attention Unit [36], we took advantage of the strong local perception ability of CNN to propose a Dual Attention Unit, DAU, to preprocess image feature.

The DAU consists of spatial attention and channel attention, as shown in Fig. 3. When calculating spatial attention, we use small kernel depth-wise convolutions (DW_Conv) [7], depth-wise convolution with dilations (DWD_Conv) [47] and 1×1 convolutions for spatial attention modeling to capture the maximum feeling within the channel wild. Furthermore, we obtain channel attention using an average pooling operation. Finally, we multiply the spatial attention with the inter-channel attention to get the result. Eq. (4) shows its calculation process:

$$\begin{aligned} &\mathrm{SA} = \mathrm{Conv}_{1\times1}\left(\mathrm{DWD\_Conv}\left(\mathrm{DW\_Conv(X)}\right)\right), \\ &\mathrm{CA} = \mathrm{FC}\left(\mathrm{AvgPool(X)}\right), \\ &\mathrm{X'} = (\mathrm{SA} \otimes \mathrm{CA}) \odot \mathrm{X}, \end{aligned} \tag{4}$$

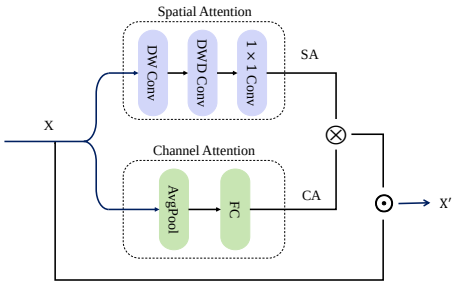

Fig. 3: Architecture of Dual Attention Unit. It consists of two parts : Spatial Attention and Channel Attention, which are used for image preprocessing and enhance the interaction ability of features between channels.

where, $\mathrm{X} \in \mathbb{R}^{\mathrm{B}\times\mathrm{C}\times\mathrm{H}\times\mathrm{W}}$ is the image input to the DAU,$\mathrm{SA} \in \mathbb{R}^{\mathrm{B}\times\mathrm{C}\times\mathrm{H}\times\mathrm{W}}$ and $\mathrm{CA} \in \mathbb{R}^{\mathrm{B}\times\mathrm{C}\times1\times1}$ represents spatial attention and channel attention respectively. FC and AvgPool are fully connected layers and average pooling layers. At the same time, we use $\odot$ and $\otimes$ to represent Kronecker product [29] and Hadamard product [1]. DAU adds only 10M FLops to the model, but increases tracking performance.

Prediction Head. We use Corner Head [43] to predict the location of target. When the image features to be predicted are sent to Corner Head, we first reweight the features based on the attention map, and enhance or suppress local features based on global information. For this component, we extract search region features via the encoder, which are then enriched using the attention map. These enhanced features are reshaped to (H_x, W_x), and subsequently fed into a fully convolutional network (FCN). This FCN comprises multiple sequential Conv-BN-ReLU layers, tasked with generating separate probability maps indicating the likelihood of the top-left and bottom-right corners of the target bounding box. Finally, by calculating the expectation of the probability distribution of the corner point, we can obtain the predicted coordinates.

3.3 Training Loss

In the process of model training, we use both classification loss and regression loss. The regression loss includes l_1 loss and GIOU loss [34]. We summarize the overall loss function as Eq. (5):

$$\mathrm{L}_{\mathrm{track}} = \mathrm{L}_{\mathrm{cls}} + \lambda_1 \mathrm{L}_1 + \lambda_{\mathrm{Giou}} \mathrm{L}_{\mathrm{Giou}} \ , \tag{5}$$

where λ_1 and λ_{Giou} represent the parameter weights of l_1 loss and GIOU loss respectively. In the experiment, we set them to $\lambda_1 = 5$ and $\lambda_{\mathrm{Giou}} = 2$.

4 Experiments

After introducing the implementation details of FERMT, in this section we show its experimental results compared with other SOTA trackers on 6 benchmarks. Subsequent ablation studies verified the rationality of the proposed structure. All trackers are implemented using Python3.8 and Pytorch1.11.0.

4.1 Implementation Details

Backbone network. ViT-tiny [39], a light-weight visual Transformer pre-trained model, is used as the backbone network of our trackers. Based on it, we use different feature extraction and relation modeling methods to design 6 different network architectures and select the best among them to construct our tracker FERMT, see Sec. 4.3.

Training. The training datasets for our model include the train-splits of GOT-10k [24], TrackingNet [32], LaSOT [14] and COCO2017 [26]. Some common data enhancement methods such as image scaling, translation and dithering are used during the training process. In particular, for the evaluation of the GOT-10k test set, we only used the training set of GOT-10k for training. The template and search images sizes are set to 128×128 and 256×256 respectively. In addition, we use the AdamW [28] optimizer to train the model and set the initial learning rate to 4e-4 and the weight decay to 1e-4.

We use 4 Nvidia GeForce RTX 3090 GPUs, with the Batch Size of each GPU set to 32, for a total of 128 total Batch Size, to train the model for 300 epoches.

Inference. During the inference process, we utilize the first frame of the video sequence as the initial template. When a new frame image is input to the model, we set the search region based on the target box from the previous frame. Then, we perform feature extraction and interaction by combining the template frame, ultimately obtaining the target result.

4.2 Comparison with the state-of-the-art trackers

To demonstrate the superiority of the proposed model, we compare it with current state-of-the-art trackers on 6 benchmarks. In addition, we conducted tracking speed evaluation on Nvidia GeForce RTX 3090 GPU and Inter Core I9-9900K CPU respectively. The results are shown in Tab. 1 and Tab. 2.

GOT-10k. GOT-10k [24] is a large dataset for object tracking. It covers 563 types of objects and has a total of 180 test sequences. The average sequence length is about 150 frames. Only the training set of GOT-10k is used for training. As shown in Tab. 1, FERMT achieved the best AO score among real-time trackers, improving 4.5 % AO compared to the previous best HCAT. Compared to the non-real-time tracker Ostrack, FERMT achieves comparable performance (69.6 vs 71.0) while being 1.7× faster on GPU and 2.8× faster on CPU.

TrackingNet. TrackingNet [32] is a large dataset selected from YouTube Bounding Boxes, containing various situations and multiple categories in natural scenes, with a total of 511 video test sequences. As shown in Tab. 1, FERMT achieved

	Method	Source	GOT-10k(*)			TrackingNet			LaSOT			Speed(*fps*)	
			AO	$SR_{0.5}$	$SR_{0.75}$	AUC	P_{Norm}	P	AUC	P_{Norm}	P	GPU	CPU
Real-time	ECO [10]	CVPR2017	31.6	30.9	11.1	55.4	61.8	49.2	32.4	33.8	30.1	240	15
	ATOM [11]	CVPR2019	55.6	63.4	40.2	70.3	77.1	64.8	51.5	57.6	50.5	83	18
	LightTrack [42]	CVPR2021	61.1	71.0	-	72.5	77.8	69.5	53.8	-	53.7	128	41
	FEAR-XS [3]	ECCV2022	61.9	72.2	-	-	-	-	53.5	-	54.5	96	35
	HCAT [5]	ECCV2022	65.1	76.5	56.7	76.6	82.6	72.9	59.3	68.7	61.0	195	45
	E.T.Track [2]	WACV2023	-	-	-	75.0	80.3	70.6	59.1	-	-	150	45
	MixformerV2-S [9]	NeurIPS2023	-	-	-	75.8	81.1	70.4	60.6	69.9	60.4	325	30
	HiT-B [18]	ICCV2023	64.0	72.1	58.1	80.0	84.4	77.3	64.6	73.3	68.1	175	33
	FERMT	Ours	69.6	80.1	63.2	80.8	85.9	78.1	65.1	74.6	69.1	225	51
Non-real-time	TransT [6]	CVPR2021	67.1	76.8	60.9	81.4	86.7	80.3	64.9	73.8	69.0	63	6
	Stark-ST50 [43]	ICCV2021	68.0	77.7	62.3	81.3	86.1	-	66.6	-	-	50	7
	SwinTrack [25]	NeurIPS2022	71.3	81.9	64.5	81.1	-	78.4	67.2	-	70.8	96	16
	MixFormer-22k [8]	CVPR2022	70.7	80.0	67.8	83.1	88.1	81.6	69.2	78.7	74.7	32	5
	$Ostrack_{384}$ [46]	CVPR2022	73.7	83.2	70.8	83.9	88.5	83.2	71.1	81.1	77.1	58	12
	$ARTrack_{384}$ [41]	CVPR2023	75.5	84.3	74.3	85.1	89.1	84.8	72.6	81.7	79.1	28	4

Table 1: Comparison of the results between our model and state-of-the-art methods on GOT-10k [24], TrackingNet [32], and LaSOT [14] benchmarks. We use gray color to denote our tracker. The best two real-time results are shown in red and blue fonts.

the best real-time tracking results of 80.8%, 85.9%, and 78.1% in the three indicators AUC, P_{Norm}, and P respectively. In addition, compared with non-real-time trackers such as SwinTrack [25] and Stark-ST50 [43], FERMT also achieved similar results.

LaSOT. LaSOT [14] is a large-scale long-term object tracking dataset, containing 1400 video sequences, with an average length of 2512 frames per video sequence. The dataset covers 70 types of objects, each containing 20 video sequences. As shown in Tab. 1, FERMT also obtained SOTA results of the real-time tracking model on the LaSOT dataset.

Speed. As shown in Tab. 1, FERMT ran 225*fps* and 51*fps* results on the GPU and CPU respectively. In the case where the performance of multiple benchmarks is stronger than HiT-B [18], FERMT still achieves faster speed than it. Furthermore, although MixformerV2-S [9] runs faster on the GPU than our model, it does so at the expense of substantial performance. So overall, our FERMT achieves fast and high-performance tracking performance on multiple devices.

NFS, UAV123 and $\mathbf{LaSOT_{ext}}$. NFS [15] is a dataset containing a series of high-speed moving objects, which includes 100 video sequences. UAV123 [31] is a manually annotated drone scene video, containing 123 video sequences. $LaSOT_{ext}$ [13] is an extension of LaSOT [14] and consists of 150 videos in 15 object classes. As shown in Tab. 2, our FERMT achieved results of 65.1, 67.5 and 46.1 on NFS, UAV123 and $LaSOT_{ext}$ respectively, all achieving the best performance of real-time trackers.

	Method	Source	NFS	UAV123	LaSOT$_{ext}$
Real-time	ECO [10]	CVPR2017	46.6	53.2	22.0
	ATOM [11]	CVPR2019	58.4	64.2	37.6
	LightTrack [42]	CVPR2021	55.3	62.5	-
	FEAR-XS [3]	ECCV2022	61.4	-	-
	HCAT [5]	ECCV2022	63.5	62.7	-
	E.T.Track [2]	WACV2023	59.0	62.3	-
	MixformerV2-S [9]	NeurIPS2023	-	65.1	-
	HiT-B [18]	ICCV2023	63.6	65.6	44.1
	FERMT	Ours	65.1	67.5	46.1
Non-real-time	TransT [6]	CVPR2021	65.7	69.1	-
	SwinTrack [25]	NeurIPS2022	-	-	47.6
	MixFormer-22k [8]	CVPR2022	-	70.4	-
	Ostrack$_{384}$ [46]	CVPR2022	66.5	70.7	50.5
	ARTrack$_{384}$ [41]	CVPR2023	66.8	70.5	51.9

Table 2: The results of our model are compared with state-of-the-art methods on NFS [15], UAV123 [31], LaSOT$_{ext}$ [13] benchmarks. We use gray color to denote our tracker. The best two real-time results are shown in red and blue fonts.

4.3 Ablation Study and Analysis

In order to verify the rationality of the FERMT model we designed, we designed detailed ablation experiments and conducted result analysis on the GOT-10k [24] benchmark.

Analysis on feature extraction and relation modeling. According to the different functions of each sub-module of the attention module, we designed five attention structures, and Eq. (6) shows their calculation methods.

$$
\begin{aligned}
\mathrm{A}_1 &= \mathrm{softmax}\left(\begin{bmatrix} \mathrm{Q_t K_t^T} & \\ & \mathrm{Q_s K_s^T} \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \\
\mathrm{A}_2 &= \mathrm{softmax}\left(\begin{bmatrix} \mathrm{Q_t K_t^T} & \\ \mathrm{Q_s K_t^T} & \mathrm{Q_s K_s^T} \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \\
\mathrm{A}_3 &= \mathrm{softmax}\left(\begin{bmatrix} \mathrm{Q_t K_t^T} & \mathrm{Q_t K_s^T} \\ \mathrm{Q_s K_t^T} & \mathrm{Q_s K_s^T} \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \\
\mathrm{A}_4 &= \mathrm{softmax}\left(\begin{bmatrix} & \\ \mathrm{Q_s K_t^T} & \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix}, \\
\mathrm{A}_5 &= \mathrm{softmax}\left(\begin{bmatrix} & \mathrm{Q_t K_s^T} \\ \mathrm{Q_s K_t^T} & \end{bmatrix}\right) \cdot \begin{bmatrix} \mathrm{V_t} \\ \mathrm{V_s} \end{bmatrix},
\end{aligned}
\tag{6}
$$

where, A_1 only uses the subdiagonal part of the attention module, and only performs self-attention on template and search. A_2 uses the same asymmetric structure as Mixformer. While extracting features, information is transmitted one-way from template to search. A_3 utilizes all the information of the attention

module, and feature extraction and two-way interaction are performed simultaneously. A_4 and A_5 cancel the feature extraction work and only use the information of the attention subdiagonal. The difference is that A_4 only performs one-way information transmission from template to search, while A_5 performs two-way transmission. Our FERMT adopts a design combining A_2 and A_5.

In order to verify the rationality of the attention module design adopted by FERMT, we compared different combination methods for experiments. Among the five attention modules of A_{1-5}, A_{1-3} is used for the first N_1 layers, and A_{4-5} is used for the last N_2 layers, thus obtaining FERMT_A_1A_4, FERMT_A_1A_5, FERMT_A_2A_4, FERMT_A_2A_5 , FERMT_A_3A_4, FERMT_A_3A_5, these 6 model architectures, Tab. 3 shows their tracking results.

Mixed and *Cross* represent the architectural types of Mixed-attention Blocks and Cross-attention Blocks. In this ablation experiment, the Mixed layers and the Cross layers adopt a 1:1 layer ratio, each with 6 layers. Dividing the experimental results into three groups: #1#2, #3#4, and #5#6 for comparison, we found that when the Cross-attention Blocks architecture is consistent, the A_2 architecture performs best in the Mixed-attention Blocks. Similarly, the experimental results are divided into two groups of #1#3#5 and #2#4#6 for comparison. We find the A_5 architecture performs best in Cross-attention Blocks.

Table 3: The performance of the model on the GOT-10k test set when Mixed-attention Blocks and Cross-attention Blocks adopt different architectures respectively. The best results are displayed in red font.

#	Mixed	Cross	AO	$SR_{0.5}$	$SR_{0.75}$
1	A_1	A_4	65.0	75.0	59.1
2	A_1	A_5	67.7	78.0	61.5
3	A_2	A_4	67.3	77.5	61.8
4	A_2	A_5	68.6	79.6	62.5
5	A_3	A_4	66.8	76.9	60.5
6	A_3	A_5	68.2	78.1	62.1

Comparison of new architecture and usual architecture. In order to further verify the redundancy of feature extraction work in the later process of object tracking, we compared this combined architecture (#1) with models that fully adopt the A_2 architecture (#2) and the entire A_3 architecture (#3). For comparison, the results are shown in Tab. 4. It is easy to find that the FERMT_A_2A_5 architecture shows strong competitiveness in terms of performance. Moreover, compared with FERMT_A_2 and FERMT_A_3, the combined architecture we designed has an advantage in computational complexity. Eq. (7) shows the time complexity of their Attention module.

$$\begin{aligned} \mathrm{O}\left(\mathrm{FERMT\_}A_2A_5\right) &= \mathrm{O}\left(\frac{1}{2}n^2 + \frac{3}{2}nm + \frac{1}{2}m^2\right), \\ \mathrm{O}\left(\mathrm{FERMT\_}A_2\right) &= \mathrm{O}\left(n^2 + nm + m^2\right), \\ \mathrm{O}\left(\mathrm{FERMT\_}A_3\right) &= \mathrm{O}\left(n^2 + 2nm + m^2\right), \end{aligned} \tag{7}$$

where n and m are the total number of tokens in template and search respectively. In the experiment, we set the image sizes of template and search to $(128, 128)$ and $(256, 256)$ respectively, thus obtaining $n = 64$ and $m = 256$. According to the calculations, compared with FERMT_A_2 and FERMT_A_3, the calculation amount is reduced by 42% and 31% respectively in the Transformer Attention part.

In addition, we have also designed models such as #4 and #5 that only perform feature interaction without relation modeling. Their tracking performance has been greatly reduced, which shows that appropriate feature extraction work is very important in object tracking. necessary.

Table 4: When using the architecture combining A_2 and A_5 and using the A_2, A_3, A_4 or A_5 architecture throughout, the model's performance on the GOT-10k test set, the best results are shown in red font.

#	Model	AO	$SR_{0.5}$	$SR_{0.75}$
1	FERMT_A_2A_5	68.6	79.6	62.5
2	FERMT_A_2	68.4	78.8	62.4
3	FERMT_A_3	68.3	78.2	62.8
4	FERMT_A_4	59.8	69.0	49.1
5	FERMT_A_5	63.2	73.2	55.4

Influence of different feature extraction layer ratios. In order to verify the importance of feature extraction for object tracking, we designed an experiment to examine the impact of the layer ratio of Mixed-attention Blocks and Cross-attention Blocks on tracking performance. We set the ratio N_1:N_2 of the layers of A_2 and A_5 to 3:9 (#1) and 9:3 (#3) respectively, and compare the results with the original 6:6 (#2). The results are shown in Tab. 5 shown. The experimental results show that the ratio of the number of layers between A_2 and A_5 is 6:6, which is the most appropriate choice.

Table 5: When the combined architecture of A_2 and A_5 uses different layer ratios, the model's performance on the GOT-10k test set, the best results are shown in red font.

#	$N_1 : N_2$	AO	$SR_{0.5}$	$SR_{0.75}$
1	3 : 9	66.4	76.6	59.9
2	6 : 6	68.6	79.6	62.5
3	9 : 3	68.3	77.7	62.6

Effect of preprocessing image features using DAU. We introduced CNN based DAU for image preprocessing based on the FERMT_A_2A_5 model architecture, and compared the performance of the tracker with or without DAU on multiple benchmarks. The results are shown in Tab. 6, which show that the DAU module improves the performance of trackers on multiple benchmarks. In addition, the representative tracking results from three sequences in Fig. 4 demonstrate this more intuitively.

#	w/o DAU	GOT-10k	LaSOT$_{ext}$	NFS	UAV123
1	without DAU	68.6	45.5	64.1	67.4
2	with DAU	69.6(+1.0)	46.1(+0.6)	65.1(+1.0)	67.5(+0.1)

Table 6: Based on the FERMT_A_2A_5 model architecture, whether DAU is added, the model's performance on the GOT-10k test set, the best results are displayed in red font.

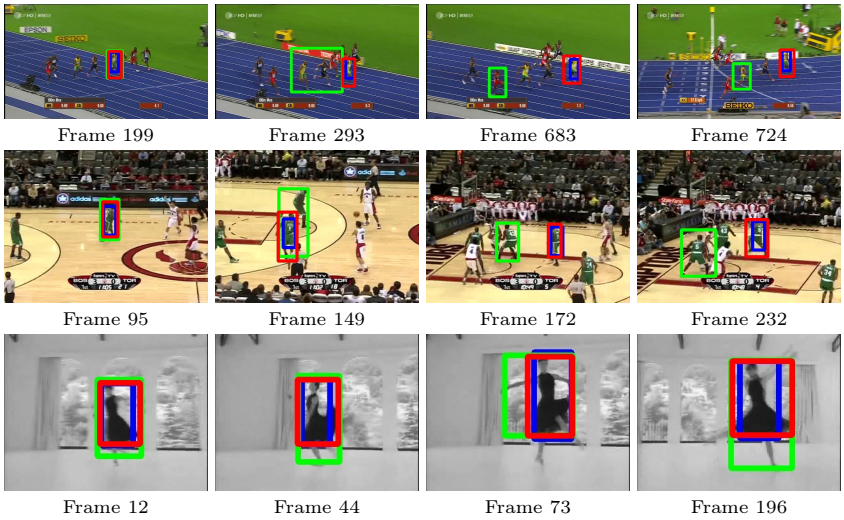

Fig. 4: This set of pictures displays the comparison of tracking results before and after DAU is added to the tracker, where red represents the tracking results of FERMT with DAU, green stands for the tracking results of trackers without DAU, and blue indicates ground truth boxes.

5 Conclusion

This paper conducts an in-depth exploration of the impact of feature extraction and relation modeling in light-weight object tracking, and proposes a new light-weight tracker based on Transformer, named FERMT. By using a backbone network composed of mixed-attention blocks and cross-attention blocks and a dual attention unit, FERMT improves the tracking speed and significantly reduces the amount of model calculations. A large number of experiments show that FERMT achieves the SOTA results among real-time trackers and further narrows the gap with high-performance trackers in terms of tracking performance.

In the future, we will conduct a more detailed analysis for whole network architecture including to light-weight backbone and predict head for better tracking accuracy and faster speed.

Acknowledgement

This project is supported by the National Natural Science Foundation of China under Grant No.61876153.

References

1. Acharya, J., Sun, Z., Zhang, H.: Hadamard response: Estimating distributions privately, efficiently, and with little communication. In: PMLR (2019)
2. Blatter, P., Kanakis, M., Danelljan, M., van Gool, L.: Efficient visual tracking with exemplar transformers. In: WACV (2023)
3. Borsuk, V., Vei, R., Kupyn, O., Martyniuk, T., Krashenyi, I., Matas, J.: Fear: Fast, efficient, accurate and robust visual tracker. In: ECCV (2022)
4. Chen, B., Li, P., Bai, L., Qiao, L., Shen, Q., Li, B., Gan, W., Wu, W., Ouyang, W.: Backbone is all y our need: A simplified architecture for visual object tracking. In: ECCV (2022)
5. Chen, X., Kang, B., Wang, D., Li, D., Lu, H.: Efficient visual tracking via hierarchical cross-attention transformer. In: ECCV (2022)
6. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: CVPR (2021)
7. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR (2017)
8. Cui, Y., Jiang, C., Wang, L., Wu, G.: Mixformer: End-to-end tracking with iterative mixed attention. In: CVPR (2022)
9. Cui, Y., Song, T., Wu, G., Wang, L.: Mixformerv2: Efficient fully transformer tracking. In: NeurIPS (2023)
10. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: ECCV (2016)
11. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: CVPR (2019)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
13. Fan, H., Bai, H., Lin, L., Yang, F., Chu, P., Ge Deng, S.Y., Huang, M., Liu, J., Xu, Y.: Lasot: A high-quality large-scale single object tracking benchmark. In: IJCV (2021)
14. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: CVPR (2019)
15. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV (2017)
16. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M.: Levit: a vision transformer in convnet's clothing for faster inference. In: CVPR (2021)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. Kang, B., Chen, X., Wang, D., Peng, H., Lu, H.: Exploring lightweight hierarchical vision transformers for efficient visual tracking. In: ICCV (2023)

19. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP (2014)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012)
21. Lan, J.P., Cheng, Z.Q., He, J.Y., Li, C., Luo, B., Bao, X., Xiang, W., Geng, Y., Xie, X.: Procontext: Exploring progressive context transformer for tracking. In: ICASSP (2023)
22. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019)
23. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR (2018)
24. Lianghua Huang, Xin Zhao, K.H.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE TPAMI (2021)
25. Lin, L., Fan, H., Zhang, Z., Xu, Y., Ling, H.: Swintrack: A simple and strong baseline for transformer tracking. In: NeurIPS (2022)
26. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context. In: ECCV (2014)
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
29. Martens, J., Grosse, R.: Optimizing neural networks with kronecker-factored approximate curvature. In: PMLR (2015)
30. Mehta, S., Rastegari, M.: Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In: ICLR (2022)
31. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: ECCV (2016)
32. Müller, M., Bibi, A., Giancola, S., Al-Subaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018)
33. Peng, Z., Huang, W., Gu, S., Xie, L., Wang, Y., Jiao, J., Ye, Q.: Conformer: Local features coupling global representations for visual recognition. In: ICCV (2021)
34. Rezatofighi, H., Tsoi, N., Gwak, J., Amir Sadeghian and, I.R., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: CVPR (2019)
35. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., anhoucke, V.V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
36. Tan, C., Gao, Z., Wu, L., Xu, Y., Xia, J., Li, S., Li, S.Z.: Temporal attention unit: Towards efficient spatiotemporal predictive learning. In: CVPR (2023)
37. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
38. Vaswani, A., Shazeer, N., Parmar, N., Uszko-reit, J., Jones, L., Gomez, A.N., Kaiser, L., IlliaPolosukhin: Attention is all you need. In: NeurIPS (2017)
39. Wang, S., Gao, J., Li, Z., Zhang, X., Hu, W.: A closer look at self-supervised lightweight vision transformers. In: ICML (2023)
40. Wang, W., Xie, E., Li, X., Fan, D., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021)
41. Wei, X., Bai, Y., Zheng, Y., Shi, D., Gong, Y.: Autoregressive visual tracking. In: CVPR (2023)

42. Yan, B., Peng, H., Wu, K., Wang, D., Fu, J., Lu, H.: Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In: CVPR (2021)
43. Yan, B., Penga, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: CVPR (2021)
44. Yang, Z., Li, Z., Shao, M., Shi, D., Yuan, Z., Yuan, C.: Masked generative distillation. In: ECCV (2022)
45. Yang, Z., Li, Z., Zeng, A., Li, Z., Yuan, C., Li, Y.: Vitkd: Practical guidelines for vit feature knowledge distillation. In: ICLR (2023)
46. Ye, B., Chang, H., Ma, B., Shan, S., Chen, X.: Joint feature learning and relation modeling for tracking: A one-stream framework. In: ECCV (2022)
47. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
48. Yue, Kaiyu, Deng, J., Zhou, F.: Matching guided distillation. In: ECCV (2020)
49. Zhang, J., Peng, H., Wu, K., Liu, M., Xiao, B., Fu, J., Yuan, L.: Minivit: Compressing vision transformers with weight multiplexing. In: CVPR (2022)