2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

LiteTrack: Layer Pruning with Asynchronous Feature Extraction for Lightweight and Efficient Visual Tracking

Qingmao Wei[1], Bi Zeng[1], Jianqi Liu[1]*, Li He[2], Guotian Zeng[1]

***Abstract*— The recent advancements in transformer-based visual trackers have led to significant progress, attributed to their strong modeling capabilities. However, as performance improves, running latency correspondingly increases, presenting a challenge for real-time robotics applications, especially on edge devices with computational constraints. In response to this, we introduce LiteTrack, an efficient transformer-based tracking model optimized for high-speed operations across various devices. It achieves a more favorable trade-off between accuracy and efficiency than the other lightweight trackers. The main innovations of LiteTrack encompass: 1) asynchronous feature extraction and interaction between the template and search region for better feature fushion and cutting redundant computation, and 2) pruning encoder layers from a heavy tracker to refine the balnace between performance and speed. As an example, our fastest variant, LiteTrack-B4, achieves 65.2% AO on the GOT-10k benchmark, surpassing all preceding efficient trackers, while running over 100 *fps* with ONNX on the Jetson Orin NX edge device. Moreover, our LiteTrack-B9 reaches competitive 72.2% AO on GOT-10k and 82.4% AUC on TrackingNet, and operates at 171 *fps* on an NVIDIA 2080Ti GPU. The code and demo materials will be available at https://github.com/TsingWei/LiteTrack.**

I. Introduction

Visual object tracking is a fundamental task in computer vision, which aims to track an arbitrary object given its initial state in a video sequence. In recent years, with the development of deep neural networks [1]–[4], tracking has made significant progress. In particular, the utilization of transformers [4] has played a pivotal role in the development of several high-performance trackers [5]–[11]. Unfortunately, a majority of recent research efforts [5], [12], [13] has concentrated solely on achieving high performance without considering tracking speed.

While these state-of-the-art trackers might deliver real-time performance on powerful GPUs, their efficiency diminishes on devices with limited computational resources. For instance, ARTrack [14], considered as a top-tier tracker, reaches a tracking speed of 37 frames per second (*fps*) on the NVIDIA RTX 2080Ti GPU but drops to 5 *fps* on the Nvidia Jetson Orin NX, a common edge device. This underscores the pressing need for trackers that effectively strike a balance between performance and speed.

The one-stage structure has gained popularity in tracking applications [9], [10], [16], [17]. This structure combines feature extraction and fusion as a joint process as pictured in Fig. 2 (a), leveraging the capabilities of the transformer network, especially the ViT [18] that has been pre-trained by mask-image-modeling(MIM) [19], [20]. Conversely, two-stage trackers [5], [6], [21], operating by sequentially extracting features and then fusing them, benefit from caching template features during the testing phase, as shown in Fig. 2 (b). However, the two-stage trackers who extract feature first then perform feature fusion, can cache the template feature during testing, while the one-stage trackers can not. Even though most of one-stage trackers are running faster than the two-stage, we can further accelerate the former by the similar caching technique. Inspired by ViTDet [22], we find that only the last layer of template feature is sufficient and better for fusion with the search feature of various earlier layer, which can be cached in the testing like two-stage trackers. Therefore, this naturally decides our overall design: the feature extraction of template is performed first and individually, then the extracted last-layer template features interact with the feature extraction of the search region, as shown in Fig. 2(c).

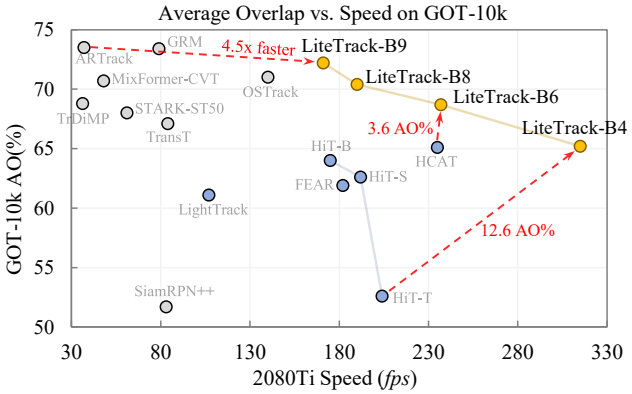


Fig. 1. Performance comparison of LiteTrack against state-of-the-art trackers on GOT-10k in terms of Average Overlap and RTX 2080Ti Speed. ○ and ● reperesent for non-real-time and real-time trackers respectively, based on Nvidia Jetson Orin NX speed (see Tab.II). Our LiteTrack (●) family offers comparable accuracy to all other trackers, significantly outpacing them in inference speed. Notably, LiteTrack-B4 achieves over 300 *fps* on 2080Ti and 100 *fps* (ONNX) on edge device. Notice that our LiteTrack delivering the best real-time accuracy trained without extra data, unlike the other efficient trackers.

Traditional efficient trackers have primarily sought to achieve faster runtimes by directly incorporating an initially lightweight-designed network as their backbone. These lightweight networks are designed for efficiency, which results in relatively mediocre performance in their upstream tasks like image classification. Consequently, when such

*Corresponding author
[1]Q. Wei, B. Zeng, J. Liu, and G. Zeng are with the School of Computer, Guangdong University of Technology, Guangzhou 510006, China. liujianqi@ieee.org
[2]L. He is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China.

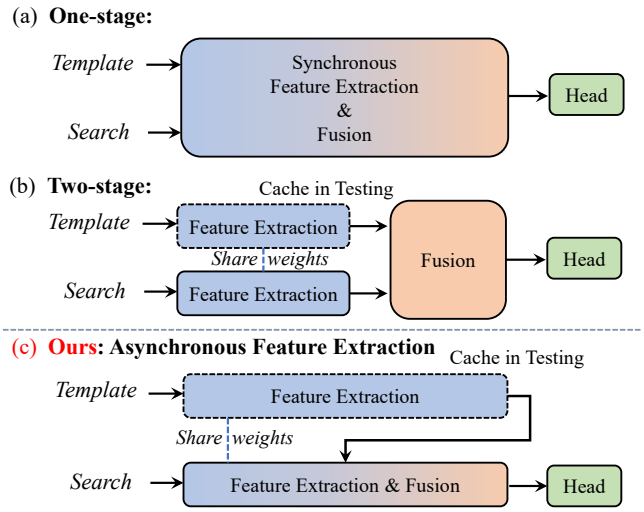


Fig. 2. Comparison of the popular architectures for visual tracking. Our method (c) is able to cache the template features like two-stage (b) in testing and also enjoy the powerful pretrain technique like one-stage method (a).

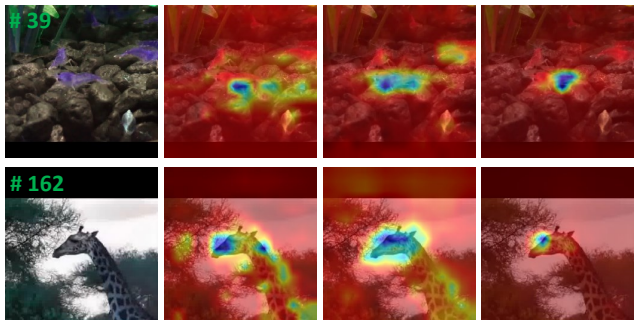


Fig. 3. Visualization of attention map (average attention value over all template features attending to the search features) of 2nd , 6th and 8th layer in the twelve-layer enoder of JNTrack [15]. The model focuses nearly precisely on the target even in the early stage of the encoder.

networks are utilized in visual tracking, their performance leaves much to be desired. In contrast, our approach derive efficient model by scaling down a high-performing heavy tracker, instead of starting with a lightweight architecture. This strategy is inspired by our observation, as depicted in Fig.3, that early layers pay sufficient attention to the target. By pruning network layers and integrating our novel asynchronous feature extraction technique, we ensure only a marginal drop in performance even when multiple layers are excised. Consequently, LiteTrack not only rivals the performance of its heavyweight peers but also competes in runtime with lightweight models, presenting an optimal trade-off. Fig.1 reinforces this assertion, showcasing LiteTrack's commendable performance on the challenging GOT-10k [23] benchmark, standing shoulder-to-shoulder with state-of-the-art (SOTA) trackers.

Our contributions are summarized as follows:

- A efficient tracking achitechture which feature extractions of template and search region are asynchronous is proposed for reducing redundant computation.
- A novel scaling principle of tracking model is introduced by adjusting encoder layers for trade-off between accuracy and speed.
- Comprehensive evaluations on authoritative generic visual tracking benchmarks have validated the excellent performance of LiteTrack compared with other SOTA trackers. Edge device deployment are tested with promising performance, demonstrating the superior effectiveness of LiteTrack on robotics applicability.

II. Related Works

A. Visual Tracking with Transformers.

Visual tracking has seen the rise of Siamese-based methods [12], [21], [24]–[30] that typically employ dual-backbone networks with shared parameters. They have been instrumental in the field due to their efficiency in feature extraction of the template and search region images. Further advancements introduced transformers [4] into the tracking community [5]–[7], [31]–[34], leveraging them for feature interaction training from scratch. The emergence of the one-stream framework [9], [10], [16], [17], [35] showcased improved performance by integrating feature extraction and fusion within the backbone network, enjoying the powerful mask-image-modeling (MIM) pretraining method [19], [20], [36]. Despite their effectiveness, these methods, tailored for powerful GPUs, often falter in speed on edge devices. In response, our research incorporates the last layer feature of the template directly into the search region's feature extraction, provide a similar cache-in-testing ability like two-stage tracker while also enjoying the powerful pretraining.

B. Lightweight Trackers.

Efficiency in tracking is crucial for practical robotics applications, especially on edge devices. Early methods such as ECO [37] and ATOM [38] focused on real-time operation but didn't achieve the accuracy levels of newer trackers. Recent advancements [39]–[41] have employed lightweight-designed backbones for efficient real-time tracking. However, these solutions still show a performance gap when compared to SOTA heavyweight trackers [6], [9], [10]. There have been efforts to refine these advanced trackers: OSTrack [10] considered pruning non-essential features unrelated to the foreground, while SimTrack [16] suggested removing the last four layers to cut computational costs. Despite these modifications, there remains a lack of deep exploration into true real-time lightweight tracking architectures. Our proposed LiteTrack fills this gap, combining efficiency and performance for effective tracking on edge devices.

III. Proposed Method

A. Overview

As shown in Fig. 4, LiteTrack is a combination of one-stage and two-stage tracking framework consisting of two components: the lightweight transformer encoder and the head network. The template of target to be tracked is fed into the lightweight transformer encoder first for feature extraction individually. Then the image of search region

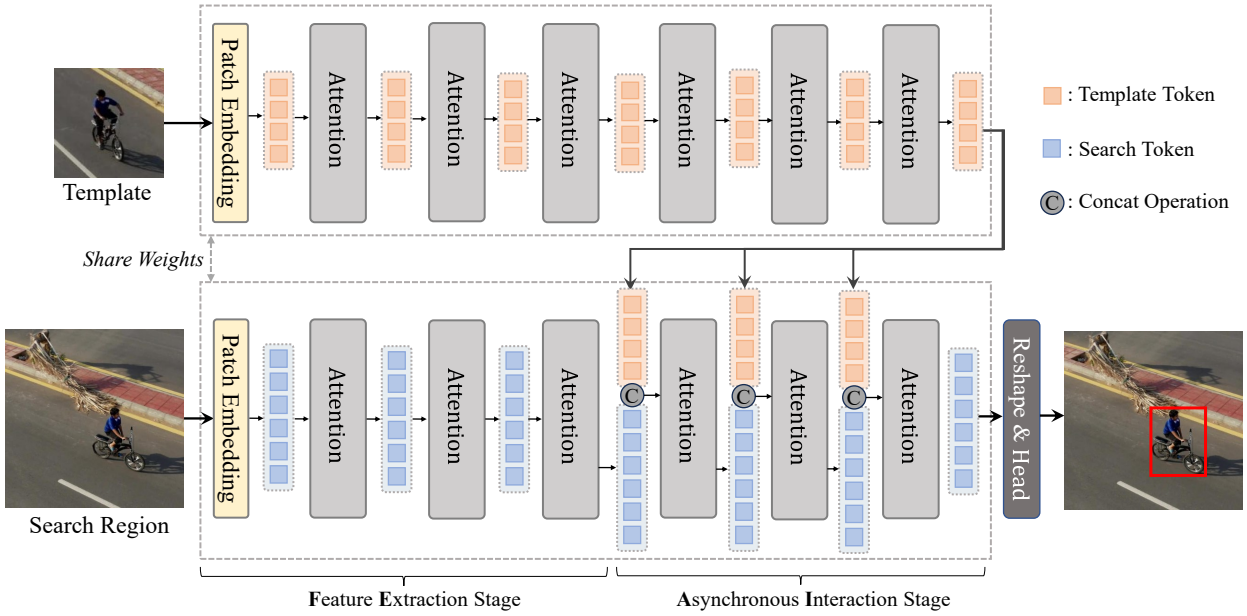


Fig. 4. Overview of the proposed LiteTrack-B6 tracker, consist of 3 layers in feature extraction (FE) stage and 3 layers in asynchronous interaction (AI) stage. For simplicity, we omit the position enocding, skip connetion and MLP in the figure. Two branchs of network for template and search region share the same weights.

is also fed into the same encoder but only the first n layers. We call these first n layers as *Feature Extraction Stage* (FE). Next, the extracted template features of last layer together with the intermediate search features after the feature extraction stage, are fed as a concatenated sequence into the remaining encoder layers. We call these final layers as *Asynchronous Interaction Stage* (AI). Finally, only the part belongs to the search of the final sequence are selected and flatten to 2D feature map, and fed into the head network for the tracking result.

B. Asynchronous Feature Extraction

The so-called *asynchronous* means that we extract the template feature first and then the search feature. The feature extraction is done by the transformer encoder, in which each layer mainly consists of multi-head attention. Specifically, for a template image $\mathbf{Z} \in \mathbb{R}^{3\times H_z \times W_z}$, the patch embedding layer transform the image into sequence of tokens $\mathbf{Z_p} \in \mathbb{R}^{C\times \frac{H_z}{16}\times \frac{W_z}{16}}$. In each layer of the transformer encoder, the main operation is multi-head self attention:

$$\mathrm{Attn_z} = \mathrm{softmax}(\frac{\mathbf{Q}_z\mathbf{K}_z^\top}{\sqrt{d_k}})\mathbf{V}_z. \tag{1}$$

The featrue extraction of template is performed by a serially stacked encoder layers. For a search image $\mathbf{X} \in \mathbb{R}^{3\times H_x \times W_x}$, the same pacth embedding layer also transform the image into sequence of tokens $\mathbf{X_p} \in \mathbb{R}^{C\times \frac{H_x}{16}\times \frac{W_x}{16}}$. In the *feature extraction stage*, the search tokens only attent to itself in the multi-head attention operation:

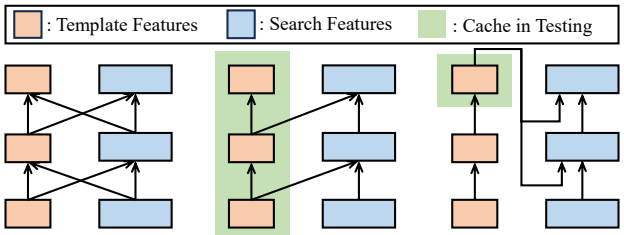
$$\mathrm{Attn_x} = \mathrm{softmax}(\frac{\mathbf{Q}_x\mathbf{K}_x^\top}{\sqrt{d_k}})\mathbf{V}_x. \tag{2}$$

In the *asynchronous interaction stage*, the tokens of extracted template features of last layer concatenated together with the intermediate search features after the *feature extraction stage* are concatenated, as the input of the encoder layer. Inspired by MixFormer [9], the attention in the layers within the interaction stage is a little different from the standard self-attention: we generate queries $\mathbf{Q}$ only by the search features. Thus the attention process can be written as:

$$\begin{aligned}\mathrm{Attn_{xz}} =& \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \\ =& \mathrm{softmax}(\frac{[\mathbf{Q}_x][\mathbf{K}_x^\top;\mathbf{K}_z^\top]}{\sqrt{d_k}})[V_x;V_z].\end{aligned} \tag{3}$$

Though the attention computation are different between two stages, the parameters of the networks are still in the same structure. Therefore the template branch and search branch of the network can share the weight.

Analysis: Our method works depending on one critical development of recent trackers: the application of homogeneous structure backbone such as the ViT [18] encoder. The chanel of the features, or the dimension of the tokens in context of the transformer, remains unchange during the layers of encoder, therefore the template features can interact with the search features from any intermediate encoder layer. This introduce our first model scaling principle: adjusting the layers for the feature extraction stage and the asynchronous stage for the accuracy-speed trade-off as discussed in Sec. IV-C. During testing, synchronous and symmetric feature extraction methods, such as OSTrack [10] shown in Fig.5(a), often result in redundant computations for the template. Given that the template, typically the initial frame of a video sequence, remains unchanged, its feature also remains constant. By caching these template features, we eliminate unnecessary computations during testing. In contrast to MixFormer, which caches every layer of template features as depicted in Fig.5(b), our method conserves memory by only storing the last layer of template features, as shown in Fig. 5(c).



(a) Synchronous & Symmetric (b) Synchronous & Asymmetric (b) Asynchronous & Asymmetric

Fig. 5. Comparative architectures of recent one-stage one-stream trackers versus our proposed method. (a) Symmetric interaction: both template and search region features engage layer by layer. (b) Template-only updates: features of the template update without search region interactions. (c) Last-layer interaction: only the final-layer template feature engages with the search region.

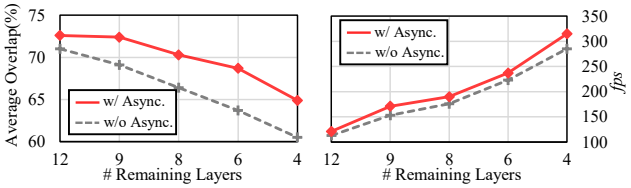


Fig. 6. Performance (left) and speed (right) vs. number of pruning layers on GOT-10k benchmark. The baseline without asynchronous feature extraction is built upon OSTrack [10].

C. Layer Pruning

In the pursuit of enhancing object tracking performance, deep neural networks have grown increasingly complex, often at the expense of computational efficiency. Layer pruning offers an avenue to mitigate this by systematically reducing the number of layers in the network. Starting with a 12-layer ViT encoder, we adopted a top-down pruning strategy, progressively eliminating layers and assessing performance against a baseline. As illustrated in Fig.6, the performance dropped with the layers pruned while the speed rise up significantly. However, when paired with asynchronous feature extraction, the decreasing of performance by the layer pruning is moderated. For example, our 9-layer variant combined with asynchronous as outperformed its 12-layer counterpart as shown in Fig.6.

D. Head and Training Objective

We employ the center head [10] for prediction, which consists of three convolution branches for center classification, offset regression and size regression, respectively. The center classification branch outputs a centerness score map, where each score represents the confidence of the target center locating at the corresponding position. The prediction of offset regression branch is for the discretization error of the center. The size regression branch predicts the height and width of the target. The position with the highest confidence in the center score map is selected as the target position and the corresponding regressed coordinates are used to compute a bounding box as the final prediction.

We apply the weighted focal loss [42] for classification. For localization, we combine ℓ_1 loss and the generalized GIoU loss [43] as the training objective. The overall loss function can be formulated as

$$\mathcal{L} = \mathcal{L}_{\text{focal}} + \lambda_G \mathcal{L}_{\text{GIoU}} + \lambda_l \mathcal{L}_l, \tag{4}$$

where $\lambda_G = 2$ and $\lambda_l = 5$ are trade-off weights following [10] to balance optimization.

Model		LiteT-B9	LiteT-B8	LiteT-B6	LiteT-B4
# FE Layers		6	6	3	2
# AI Layers		3	2	3	2
fps	2080Ti (PyTorch)	171	190	237	315
	OrinNX (PyTorch)	21	25	31	44
	OrinNX (ONNX fp16)	64	70	82	102
MACs(G)		14.17	12.77	10.09	6.78
Params(M)		54.92	49.60	38.97	26.18

TABLE I Details and variants of our LiteTrack model. FE and AI denote for Feature Extraction and Asynchronous Interaction, respectively.

IV. Experiments

A. Implementation Details

Model Variants. We use ViT-B [18] pretrained by CAE [20] as our backbone. We develop four variants of LiteTrack with using different layers of the transformer, as elaborated in Tab. I. We adopt first 9, 8, 6 and 4 layers of the 12-layer ViT-B model for LiteTrack-B9, LiteTrack-B8, LiteTrack-B6, and LiteTrack-B4, respectively. FE layers and AI layers in the table denote for the layers used in feature extraction stage and asynchronous interaction stage, respectively. In addition, Tab. I reports model parameters, Multiply-Accumulate operations (MACs), and inference speed on multiple environments. The running setups are detailed decribed in Sec IV-B.

Training. For the GOT-10k [23] benchmark, we only use the training split of GOT-10k following the one-shot protocols and train the model for 100 epochs. For the other benchmarks, the training splits of GOT-10k, COCO [53], LaSOT [45] and TrackingNet [44] are used for training in 300 epochs. For video datasets, we sample the image pair from a random video sequence. For the image dataset COCO, we randomly select an image and apply data augmentations to generate an image pair. Common data augmentations such as scaling, translation, and jittering are applied on the image pair. The search region and the template are obtained by expanding the target box by a factor of 4 and 2, respectively. The optimizer is the AdamW optimizer [54], with the weight decay of 1e-4. The initial learning rate of the encoder and the decoder are 4e-5 and 4e-4, respectively. We reduce the learning rate to 10% in the last 20% epochs. Each GPU holds 64 image-pairs. Training of LiteTrack-B8 takes about 9 hours for GOT-10k and 24 hours for the other benchmarks on one RTX 3090 GPU, and the lighter model trains faster.

Testing. During inference, the template is initialized in the first frame of a video sequence. For each subsequent frame, the search region is cropped based on the target's bounding box of the previous frame. We adopt Hanning window penalty to utilize positional prior like scale change and motion smoothness in tracking, following the common

	Method	Source	TrackingNet [44]			LaSOT [45]			GOT-10k* [23]			Speed (*fps*)	
			AUC	P_{Norm}	P	AUC	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$	2080Ti	OrinNX
Real-time	LiteTrack-B9	Ours	**82.4**	**87.3**	**80.4**	**67.0**	**77.0**	**72.7**	**72.2**	**82.3**	**69.3**	171	21
	LiteTrack-B8	Ours	81.4	86.4	79.4	66.4	76.4	71.4	70.4	80.1	66.4	190	25
	LiteTrack-B6	Ours	80.8	85.7	78.2	64.6	73.9	68.9	68.7	78.2	64.2	237	31
	LiteTrack-B4	Ours	79.9	84.9	76.6	62.5	72.1	65.7	65.2	74.7	57.7	315	44
	HiT-Base[1] [41]	ICCV'23	80.0	84.4	77.3	64.6	73.3	68.1	64.0	72.1	58.1	175	-
	E.T.Track [46]	WACV'23	75.0	80.3	70.6	59.1	-	-	-	-	-	67	21
	FEAR-XS [40]	ECCV'22	-	-	-	53.5	-	54.5	61.9	72.2	-	182	50
	HCAT [47]	ECCVW'22	76.6	82.6	72.9	59.3	68.7	61.0	65.1	76.5	56.7	235	34
	LightTrack [39]	CVPR'21	72.5	77.8	69.5	53.8	-	53.7	61.1	71.0	-	107	38
	HiFT [48], [49]	ICCV'21	66.7	73.8	60.9	45.1	52.7	42.1	-	-	-	230	50
	ECO [37]	CVPR'17	55.4	61.8	49.2	32.4	33.8	30.1	39.5	40.7	17.0	113	22
Non-real-time	ARTrack-256 [14]	CVPR'23	*84.2*	*88.7*	83.5	*70.4*	*79.5*	*76.6*	*73.5*	82.2	*70.9*	37	6
	GRM-256 [35]	CVPR'23	84.0	*88.7*	83.3	69.9	79.3	75.8	73.4	*82.9*	70.4	79	10
	OSTrack-256 [10]	ECCV'22	83.1	87.8	82.0	69.1	78.7	75.2	71.0	80.4	68.2	140	18
	MixFormer [9]	CVPR'22	83.1	88.1	81.6	69.2	78.7	74.7	70.7	80.0	67.8	48	12
	Sim-B/16 [16]	ECCV'22	82.3	86.5	-	69.3	78.5	-	68.6	78.9	62.4	131	15
	STARK-ST50 [6]	ICCV'21	81.3	86.1	-	66.6	-	-	68.0	77.7	62.3	61	13
	TransT [5]	CVPR'21	81.4	86.7	80.3	64.9	73.8	69.0	67.1	76.8	60.9	84	13
	DiMP [13]	ICCV'19	74.0	80.1	68.7	56.9	65.0	56.7	61.1	71.7	49.2	100	16
	SiamRPN++ [26]	CVPR'19	73.3	80.0	69.4	49.6	56.9	49.1	51.7	61.6	32.5	83	16
	ATOM [38]	CVPR'19	70.3	77.1	64.8	51.5	57.6	50.5	55.6	63.4	40.2	175	15

TABLE II State-of-the-art comparison on TrackingNet [44], LaSOT [45], and GOT-10k [23] benchmarks. The best three real-time results are shown in **red**, blue and green fonts, and the best non-real-time results are shown in *underline* font. * denotes results on GOT-10k obtained following the official one-shot protocol, with gray font indicating training using extra data.

Method	NFS [50] AUC	UAV123 [51] AUC	VOT'21 [52] EAO
LiteTrack-B9	**65.4**	**67.7**	**0.269**
LiteTrack-B8	64.6	67.1	0.261
LiteTrack-B6	64.4	66.2	0.254
LiteTrack-B4	63.4	66.4	0.251
HiT-Base [41]	63.6	65.6	0.252
HCAT [47]	63.5	62.7	-
FEAR [40]	61.4	-	0.250
E.T.Track [46]	59.0	62.3	0.224
LightTrack [39]	55.3	62.5	0.225
HiFT [48]	-	58.9	-
ECO [37]	46.6	53.2	-

TABLE III Comparison with state-of-the-art real-time trackers on additional benchmarks. We report the results on the NFS [50] and UAV123 [51] in AUC score, and the EAO for VOT2021 [52] real-time benchmark.

practice [10], [33]. The output scores are simply element-wise multiplied by the Hanning window with the same size, and we choose the box with the highest multiplied score as the target box.

B. State-of-the-art Comparisons

LiteTrack is benchmarked against state-of-the-art trackers, both real-time and non-real-time, across six tracking datasets. We evaluated the speed of these trackers on two distinct platforms: an Nvidia GeForce RTX 2080Ti GPU (with an Intel i5-11400F CPU) and an Nvidia Jetson Orin NX 16GB edge device. For these tests, we utilized PyTorch 1.12.0 on the former and PyTorch 2.0.0 @ JetPack 5.1 on the latter. Trackers are categorized into real-time and non-real-time based on their PyTorch speed on the Orin NX device, following the 20 *fps* real-time setting of VOT [55]. Detailed comparative results are showcased in Tables II and III. We also report our tracker's speed accelerated with ONNX fp16 in Tab. I.

GOT-10k. GOT-10k [23] is a large-scale and challenging dataset that contains 10k training sequences and 180 test sequences which are zero-overlapping in classes. The official one-shot protocol requires the evaluated tracker training without extra data, which encourages the model designed for challenging scenes like unseen objects. We report the Average Overlap (AO), Success Rate over a overlapping rate of 50% ($SR_{0.5}$) and the same one over 75% ($SR_{0.75}$) obtained by submitting the result to the official evaluation server. As shown in Table II, LiteTrack-B9 achieves the best real-time results of 72.2% AO score, which is also competitive to the best non-real-time tracker ARTrack-256 [14] (73.5% AO score). Our LiteTrack-B4 surpassing all the real-time trackers with AO score of 65.2%, even though our trackers are trained without extra data.

TrackingNet. TrackingNet [44] is a large-scale dataset containing a variety of situations in natural scenes and multiple categories, and its test set includes 511 video sequences. We report the Area Under Curve (AUC), Normalized Precision (P_{Norm}) and Precision (P) obtained by submitting the tracking result to the official evaluation server. As reported in Table II, LiteTrack series achieve competitive results compared with the previous real-time trackers. Compared to non-real-time tracker ARTrack [14], LiteTrack-B9 achieves comparable performance to it in AUC (82.4 $vs.$ 84.2) while being $4.5\times$ faster on the GPU and $6\times$ faster on the Jetson edge platform.

LaSOT. LaSOT [45] is a large-scale, long-term dataset containing 1400 video sequences, with 1120 training videos and 280 test videos. We report the same matrices as in

[1]The *fps* on GPU of HiT [41] is directly from their paper due to limited access to their code. We consider HiT as a real-time tracker based on their reported results on the AGX Xavier, which has a performance near to our Orin NX. For a hardware comparison, see the benchmarks on https://developer.nvidia.com/embedded/jetson-benchmarks.

#	Method	AO	*fps*
1	OSTrack(w/o CE)	71.0	121
2	Strong baseline(1 + CAE [20] pretrained)	71.7	120
3	2 + Last 3 layers pruned	69.1	162
4	3 + Asynchronous Feature Extraction	**72.2**	**171**

TABLE IV Comparison of key components for object tracking performance. We use gray color to denote for our final configuration.

# Total Layers	# FE Layers	# AI Layers	GOT AO	LaSOT AUC	*fps*
8	6	2	70.3	66.1	190
	5	3	**70.4**	**66.4**	185
	0	8	68.3	65.9	173
6	4	2	68.0	64.5	241
	3	3	**68.7**	**64.6**	237
4	3	1	64.6	61.3	318
	2	2	**65.2**	**62.5**	315

TABLE V Performance comparison based on varying ratios of feature extraction layers to asynchronous interaction layers. We use gray color to denote our final configuration.

TrackingNet evaluated by PyTracking[2] tools. The results on LaSOT are shown in Table II. LiteTrack-B9 achieves the best real-time results of 67.0%, 77.0%, and 72.7% in AUC, P_{Norm}, and P, respectively.

NFS, UAV123 and VOT2021. On the NFS dataset [50], known for its fast-moving objects spanning 100 video sequences, our LiteTrack variants B9, B8, and B6 emerge as the top three in real-time performance as highlighted in Table III. Meanwhile, on the UAV123 dataset [51], which features 123 video clips from low-altitude UAVs, even our fastest LiteTrack-B4 takes the lead among real-time trackers with an AUC score of 66.4%, surpassing competitors such as HiT [41] and HCAT [47] by margins of 0.8% and 3.7%, respectively. Similarly, our VOT-2021 real-time experiments on the VOT2021 benchmark [52] witnessed LiteTrack-B9 achieving the highest EAO score of 26.9% among real-time trackers, as tabulated in Table III.

C. Ablation Study and Visualization

Component-wise Analysis. The significance of our proposed methods is underscored through a comparative study built upon OSTrack [10]. For setting a solid baseline, we enhanced OSTrack by substituting its MAE [19] pretrained weights with those of CAE [20], the outcome of which is enumerated in Tab. IV, Row 2. Direct layer pruning, as seen in Row 3, led to a marked decline in performance. However, when integrated with our novel asynchronous feature extraction (Row 4), not only was the deficit recovered, but the model also achieved superior accuracy and efficiency, surpassing even the strong baseline.

Layer Configuration Analysis. We explored various configurations concerning the ratio of feature extraction (FE) layers to asynchronous interaction (AI) layers, as depicted in Tab. V. For configurations with 8 total layers, peak performance was achieved with a majority of the layers dedicated to FE. The 6-layer configuration showed comparable results, especially with an even FE-to-AI ratio. Notably, in the 4-layer configurations, a balanced 2:2 FE-to-AI setup still produced respectable results. The data highlights the model's adaptability across different layer configurations and offers insights into achieving an optimal balance between FE and AI layers.

[2]https://github.com/visionml/pytracking

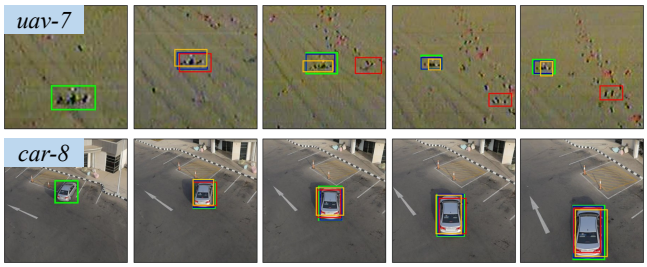


Fig. 7. Prediction comparison from UAV123 [51]. We use green lines to demonstrate the Ground Truth bounding box of the target. Blue boxes represent our LiteTrack's predictions, while yellow and red boxes denote the predictions of trackers HCAT [47] and E.T.Track [46] respectively.

Qualitative Results. To better present the superiority of LiteTrack, we highlight representative scenes in Fig. 7. In a challenging UAV tracking scenario under a noisy and jittery UAV camera feed, LiteTrack consistently maintains its track, outperforming other trackers. Similarly, when tracking a moving car from a UAV's perspective, LiteTrack demonstrates pinpoint precision, ensuring more accurate alignment with the ground truth than competing methods. These real-world tests underscore LiteTrack's proficiency in handling diverse tracking challenges.

Failure Cases Analysis. Failure for robotics applications can be catastrophic. We do not handle the occlusion or other scenarios explicitly which may cause failure as the existing lightweight methods do. In our experiment we notice that the drop of overall performance on the lighter model is mainly caused by the inaccurate bounding box regression and poor ability to capture the detail of the target when the most part of it is occluded, which may be focused in the future works.

V. Conclusions

In this work, we've presented LiteTrack, a pioneering approach to object tracking tailored for robotics applications and edge devices. By combining layer pruning with asynchronous feature extraction, we've achieved significant improvements in both accuracy and execution speed across diverse datasets. Our results underscore LiteTrack's potential, as it not only outperforms leading real-time trackers but also addresses the constraints of computational resources often found in robotics and edge deployments. With its efficient design, LiteTrack promises to be a valuable basline for real-time robotics applications.

Acknowledgment

This work was supported in part by the National Science Foundation of China under Grant 62172111, National Joint Fund Key Project (NSFC - Guangdong Joint Fund) under Grant U21A20478.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[5] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer Tracking," in *CVPR*, 2021, pp. 8126–8135.

[6] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning Spatio-Temporal Transformer for Visual Tracking," in *ICCV*, 2021, pp. 10 428–10 437.

[7] N. Wang, W. Zhou, J. Wang, and H. Li, "Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking," in *CVPR*, 2021, pp. 1571–1580.

[8] B. Yu, M. Tang, L. Zheng, G. Zhu, J. Wang, H. Feng, X. Feng, and H. Lu, "High-Performance Discriminative Tracking with Transformers," in *ICCV*, 2021, pp. 9836–9845.

[9] Y. Cui, C. Jiang, L. Wang, and G. Wu, "Mixformer: End-to-End Tracking with Iterative Mixed Attention," in *CVPR*, 2022, pp. 13 598–13 608.

[10] B. Ye, H. Chang, B. Ma, S. Shan, and X. Chen, "Joint Feature Learning and Relation Modeling for Tracking: A One-Stream Framework," in *ECCV*, 2022, pp. 341–357.

[11] X. Chen, H. Peng, D. Wang, H. Lu, and H. Hu, "Seqtrack: Sequence to sequence learning for visual object tracking," in *CVPR*, 2023, pp. 14 572–14 581.

[12] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High Performance Visual Tracking With Siamese Region Proposal Network," in *CVPR*, 2018, pp. 8971–8980.

[13] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning Discriminative Model Prediction for Tracking," in *ICCV*, 2019, pp. 6181–6190.

[14] X. Wei, Y. Bai, Y. Zheng, D. Shi, and Y. Gong, "Autoregressive visual tracking," in *CVPR*, June 2023, pp. 9697–9706.

[15] Q. Wei, B. Zeng, and G. Zeng, "Towards efficient training with negative samples in visual tracking," *arXiv preprint arXiv:2309.02903*, 2023.

[16] B. Chen, P. Li, L. Bai, L. Qiao, Q. Shen, B. Li, W. Gan, W. Wu, and W. Ouyang, "Backbone is All Your Need: A Simplified Architecture for Visual Object Tracking," in *ECCV*, 2022, pp. 375–392.

[17] F. Xie, C. Wang, G. Wang, Y. Cao, W. Yang, and W. Zeng, "Correlation-aware deep tracking," in *CVPR*, 2022, pp. 8751–8760.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021.

[19] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. of CVPR*, 2022.

[20] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang, "Context autoencoder for self-supervised representation learning," *International Journal of Computer Vision*, pp. 1–16, 2023.

[21] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast Online Object Tracking and Segmentation: A Unifying Approach," in *CVPR*, 2019, pp. 1328–1338.

[22] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 280–296.

[23] L. Huang, X. Zhao, and K. Huang, "Got-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild," *IEEE TPAMI*, vol. 43, no. 5, pp. 1562–1577, 2021.

[24] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," in *ECCV*, 2016, pp. 850–865.

[25] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese Instance Search for Tracking," in *CVPR*, 2016, pp. 1420–1429.

[26] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks," in *CVPR*, 2019, pp. 4282–4291.

[27] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines." in *AAAI*, 2020, pp. 12 549–12 556.

[28] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking," in *CVPR*, 2020, pp. 6268–6276.

[29] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese Box Adaptive Network for Visual Tracking," in *CVPR*, 2020, pp. 6667–6676.

[30] Z. Zhang and H. Peng, "Deeper and Wider Siamese Networks for Real-Time Visual Tracking," in *CVPR*, 2019, pp. 4591–4600.

[31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *ICCV*, 2021, pp. 9992–10 002.

[32] C. Mayer, M. Danelljan, G. Bhat, M. Paul, D. P. Paudel, F. Yu, and L. Van Gool, "Transforming model prediction for tracking," in *CVPR*, 2022, pp. 8731–8740.

[33] Z. Song, J. Yu, Y.-P. P. Chen, and W. Yang, "Transformer tracking with cyclic shifting window attention," in *CVPR*, 2022, pp. 8791–8800.

[34] S. Gao, C. Zhou, C. Ma, X. Wang, and J. Yuan, "AiATrack: Attention in attention for transformer visual tracking," in *ECCV*, 2022, pp. 146–164.

[35] S. Gao, C. Zhou, and J. Zhang, "Generalized relation modeling for transformer tracking," in *CVPR*, 2023, pp. 18 686–18 695.

[36] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *ICML*, 2021, pp. 8748–8763.

[37] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient Convolution Operators for Tracking," in *CVPR*, 2017, pp. 6931–6939.

[38] ——, "ATOM: Accurate tracking by overlap maximization," in *Proc. of CVPR*, 2019.

[39] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, "LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search," in *CVPR*, 2021, pp. 15 180–15 189.

[40] V. Borsuk, R. Vei, O. Kupyn, T. Martyniuk, I. Krashenyi, and J. Matas, "FEAR: Fast, Efficient, Accurate and Robust Visual Tracker," in *ECCV*, 2022, pp. 644–663.

[41] B. Kang, X. Chen, D. Wang, H. Peng, and H. Lu, "Exploring lightweight hierarchical vision transformers for efficient visual tracking," in *ICCV*, 2023.

[42] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[43] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," in *CVPR*, 2019, pp. 658–666.

[44] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild," in *ECCV*, 2018, pp. 310–327.

[45] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking," in *CVPR*, 2019, pp. 5374–5383.

[46] P. Blatter, M. Kanakis, M. Danelljan, and L. Van Gool, "Efficient Visual Tracking with Exemplar Transformers," in *WACV*, 2023, pp. 1571–1581.

[47] X. Chen, B. Kang, D. Wang, D. Li, and H. Lu, "Efficient Visual Tracking via Hierarchical Cross-Attention Transformer," in *ECCVW*, 2022, pp. 461–477.

[48] Z. Cao, C. Fu, J. Ye, B. Li, and Y. Li, "Hift: Hierarchical feature transformer for aerial tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 457–15 466.

[49] J. Thangavel, T. Kokul, A. Ramanan, and S. Fernando, "Transformers in single object tracking: An experimental survey," *arXiv preprint arXiv:2302.11867*, 2023.

[50] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking," in *ICCV*, 2017, pp. 1134–1143.

[51] M. Mueller, N. Smith, and B. Ghanem, "A Benchmark and Simulator for UAV Tracking," in *ECCV*, 2016, pp. 445–461.

[52] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, H. J. Chang, M. Danelljan, L. Cehovin, A. Lukežič *et al.*, "The ninth visual object tracking vot2021 challenge results," in *ICCV*, 2021, pp. 2711–2738.

[53] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014, pp. 740–755.
[54] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *ICLR*, 2019.
[55] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav *et al.*, "The eighth visual object tracking VOT2020 challenge results," in *ECCV*, 2020, pp. 547–601.