FARTRACK: FAST AUTOREGRESSIVE VISUAL TRACKING WITH HIGH PERFORMANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference speed and tracking performance are two critical evaluation metrics in the field of visual tracking. However, high-performance trackers often suffer from slow processing speeds, making them impractical for deployment on resource-constrained devices. To alleviate this issue, we propose **FARTrack**, a **F**ast **A**uto-**R**egressive **T**racking framework. Since autoregression emphasizes the temporal nature of the trajectory sequence, it can maintain high performance while achieving efficient execution across various devices. FARTrack introduces **Task-Specific Self-Distillation** and **Inter-frame Autoregressive Sparsification**, designed from the perspectives of **shallow-yet-accurate distillation** and **redundant-to-essential token optimization**, respectively. Task-Specific Self-Distillation achieves model compression by distilling task-specific tokens layer by layer, enhancing the model's inference speed while avoiding suboptimal manual teacher-student layer pairs assignments. Meanwhile, Inter-frame Autoregressive Sparsification sequentially condenses multiple templates, avoiding additional runtime overhead while learning a temporally-global optimal sparsification strategy. FARTrack demonstrates outstanding speed and competitive performance. It delivers an AO of 70.6% on GOT-10k in real-time. Beyond, our fastest model achieves a speed of 343 FPS on the GPU and 121 FPS on the CPU. The code will be released.

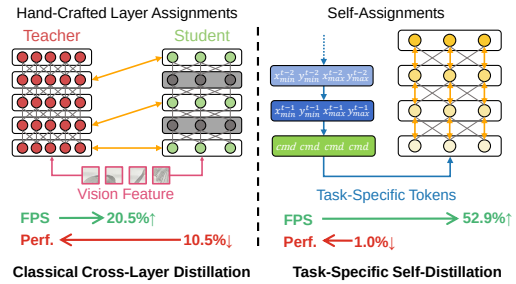
1 INTRODUCTION

Visual object tracking (VOT), aiming to continuously localize arbitrary objects in a video sequence, relies on the continuous positions of the objects and is highly sensitive to temporal information Asanomi et al. (2023); Mayer et al. (2022); Wu et al. (2023b); Zhao et al. (2023); Zhou et al. (2023a). In practical applications on edge devices with limited resources, it is often necessary to consider both speed and performance simultaneously. However, existing methods can only achieve either high speed Gopal & Amer (2024a); Li et al. (2023); Xie et al. (2023); Yang et al. (2023b); Zaveri et al. (2025) or high performance Cai et al. (2024); Chen et al. (2023); Hong et al. (2024); Xie et al. (2024); Yang et al. (2023a).

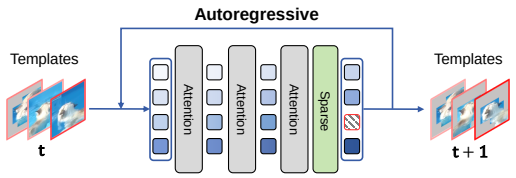
To address this dilemma, existing efforts to balance the tracking speed and performance can be broadly categorized into two approaches:

(i) Model distillation methods Cui et al. (2023); Guo et al. (2021); Hinton et al. (2015); Li et al. (2017); Romero et al. (2014) based on cross-layer train a student model to mimic a teacher's vision features. However, as shown in Figure 1(a), these methods rely on **hand-crafted layer assignments** to enable knowledge transfer Ahn et al. (2019); Passalis et al. (2020); Romero et al. (2014); Tung & Mori (2019); Yue et al. (2020); Zagoruyko & Komodakis (2016). Without prior knowledge of teacher-student layer pair assignment, manually designed ones often disrupt the hierarchical structure of feature extraction, thereby failing to achieve optimal results. Moreover, the distillation objectives of these methods focus on current-frame visual features, overlooking temporal information in trajectory sequences critical for tracking tasks.

(ii) Runtime token sparsification approaches Chen et al. (2022b); Liang et al. (2022); Rao et al. (2021); Ye et al. (2022) typically involve the gradual removal of a subset of tokens during inference. However, this process introduces **extra computational overhead** for identifying tokens to remove, ultimately reducing tracking efficiency. Moreover, as these methods prioritize the current frame rather than the



(a) Comparison of distillation methods



(b) Inter-frame Autoregressive Sparsification

Figure 1: **Overview.** (a) Comparison of our Task-Specific Self-Distillation and Classical Cross-Layer Distillation. (b) Inter-frame Autoregressive Sparsification for Multi-templates.

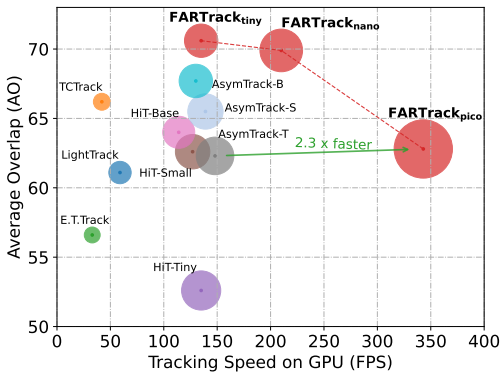


Figure 2: **FARTrack vs. Other Trackers: Performance-Speed Trade-off.** Comparison of our FARTrack with the state-of-the-art trackers on GOT-10k in terms of tracking speed (horizontal axis) on GPU and AO performence (vertical axis). The diameter of the circle is proportional to the ratio of the model's speed to its performance. FARTrack$_{naco}$ significantly surpasses existing trackers in both tracking performance and inference speed.

entire frame sequence, they fail to achieve a temporally-global optimal solution, which adversely impacts overall tracking performance.

To address these two issues, we present a fast, high-performance multi-template autoregressive framework for visual tracking, using multi-template design to boost accuracy. Our framework comprises two key components: *(i)* **Task-Specific Self-Distillation**. Unlike classical cross-layer distillation, our approach conducts layer-by-layer distillation of task-specific tokens, which represent the object's trajectory sequences. In this method, each layer acts as both student and teacher for the next, trained to fit teacher's trajectory sequence features via KL divergence. Our approach avoids suboptimal manual layer assignments while maintaining temporal information. *(ii)* **Inter-frame Autoregressive Sparsification**. Compared with the frame-wise runtime sparsification methods, our sparsification is a sequence-level method for template sequences. We treat attention weights as matrices to retain foreground tokens while discarding background tokens, and propagate sparsification results to subsequent frames in an autoregressive manner. In this process, we reduce the bandwidth without introducing extra computational load, while retaining temporal information to learn a temporally-global optimal sparsification strategy.

Overall, we present **FARTrack**, a fast and high-performance tracking framework. Extensive experiments demonstrate the effectiveness and efficiency of our approach. Our method achieves a better balance between inference speed and tracking performance than previous trackers. Specifically, as demonstrated in Figure 2, compared to the high-performance tracker AsymTrack-B, FARTrack$_{tiny}$ attains a 2.9% higher AO score on the GOT-10k benchmark while achieving comparable running speed on the GPU. Moreover, FARTrack$_{pico}$ delivers 0.5% better performance than AsymTrack-T on GOT-10k, along with superior GPU (343 FPS) and CPU (121 FPS) speeds.

2 Related Work

Efficient Tracking Framework. In practical application scenarios, it is imperative to deploy trackers that achieve both high speed Cai et al. (2023); Kou et al. (2023); Li et al. (2023); Wei et al. (2024); Zhang et al. (2023); Zhou et al. (2023b) and high performance Gao et al. (2023); Li et al. (2023); Shi et al. (2024); Tang et al. (2024); Wu et al. (2023a); Zheng et al. (2024) on resource-constrained edge devices. Over the past decade, researchers have been exploring efficient and effective tracking framework for real-world applications Bhat et al. (2019); Cao et al. (2022); Danelljan et al. (2019; 2017); Sun et al. (2025); Xie et al. (2022); Xu et al. (2020). While Siamese trackers Bertinetto et al.

(2016); He et al. (2023); Li et al. (2019; 2018); Shen et al. (2022); Tang & Ling (2022); Xing et al. (2022); Zaveri et al. (2025) with lightweight designs Yan et al. (2021b) or dynamic updates Borsuk et al. (2022) reduce computation, they often overlook temporal dependencies, limiting performance. Transformer-based methods Blatter et al. (2023); Chen et al. (2023); Gao et al. (2022; 2023); Kang et al. (2023); Lin et al. (2022); Song et al. (2023; 2022); Ye et al. (2022); Zhang et al. (2022) improve accuracy but add complexity through decoding heads. Recent generative paradigms Bai et al. (2024); Chen et al. (2023); Wei et al. (2023a) eliminate custom heads yet incur high computational costs. Existing frameworks thus face trade-offs between efficiency and performance. In this paper, we propose a more efficient generative tracking framework to better balance speed and performance.

Model Distillation. Model distillation Ahn et al. (2019); Cui et al. (2023); Shen et al. (2021); Tung & Mori (2019); Wu et al. (2024); Ma et al. (2025); Cao et al. (2025) transfers knowledge from a teacher to a lightweight student. Typical methods like AVTrack Wu et al. (2024) and MixformerV2 Cui et al. (2023) use multi-teacher maximization or layer skipping. However, such cross-layer distillation Wang et al. (2024); Zhang et al. (2023) often relies on suboptimal manual layer associations, leading to notable performance drops. Our approach compresses the model via self-distillation on task-specific tokens between adjacent layers, avoiding manual pair assignments and preserving temporal information.

Token Sparsification. Existing methods Chen et al. (2022b); Liang et al. (2022); Rao et al. (2021); Ye et al. (2022); Zhao et al. (2024a;b;c) reduce computation by progressively removing less important tokens during runtime. DynamicViT Rao et al. (2021) employs lightweight predictors for stepwise token pruning, while OSTrack Ye et al. (2022) removes background regions early in processing. However, such runtime approaches often introduce extra steps, increasing latency, and focus only on the current frame. We propose a sequence-level post-processing sparsification method that avoids additional runtime overhead, improves speed, and maintains high performance.

3 METHOD

3.1 REVISITING ARTRACK

ARTrack Wei et al. (2023a) is an end-to-end sequence generation framework for visual tracking, which represents object trajectories as discrete token sequences using a shared vocabulary. By quantizing discrete token items, we obtain the coordinates corresponding to each token, thereby enabling the model to depict object positions via discrete tokens. The framework then employs a Transformer Encoder to extract visual information and progressively model the sequential evolution of the trajectory prompted by the preceding coordinate tokens. ARTrackV2 Bai et al. (2024) adds a dynamic appearance reconstruction process on the basis of its predecessor. It models the trajectory while reconstructing the appearance in an autoregressive manner.

Motivation. Although the ARTrack series models maintain temporal information retention, their architectures incorporating excessive depth and numerous parameters exhibit bandwidth-unfriendly characteristics, ultimately reducing tracking efficiency. Conventional optimization methods, such as cross-layer distillation and runtime token sparsification, have been employed to mitigate these structural bottlenecks. However, cross-layer distillation relying on hand-crafted layer assignments disrupts the hierarchical structure of feature extraction and runtime token sparsification introduces extra computational overhead while neglecting temporal-global optimization within frame sequences. To address these limitations, we propose FARTrack, which reduces model depth via task-specific self-distillation for model compression and introduces an inter-frame autoregressive sparsification method to eliminate background redundancy and noise in the template images.

3.2 NETWORK ARCHITECTURE

As presented in Figure 3, the Transformer Encoder Dosovitskiy et al. (2020); He et al. (2022) encodes features and predicts the four coordinate tokens of the bounding box in an inter-frame autoregressive manner. Initially, all templates and the search image are divided into patches, flattened, and projected into a sequence of token embeddings. Subsequently, FARTrack maps the object positions across frames to a unified coordinate system with a shared vocabulary Chen et al. (2021; 2022a); Wei et al. (2023b), forming object trajectory tokens. Then, we concatenate the visual tokens, trajectory tokens, and four command tokens (representing the target bounding box coordinates), and input them into

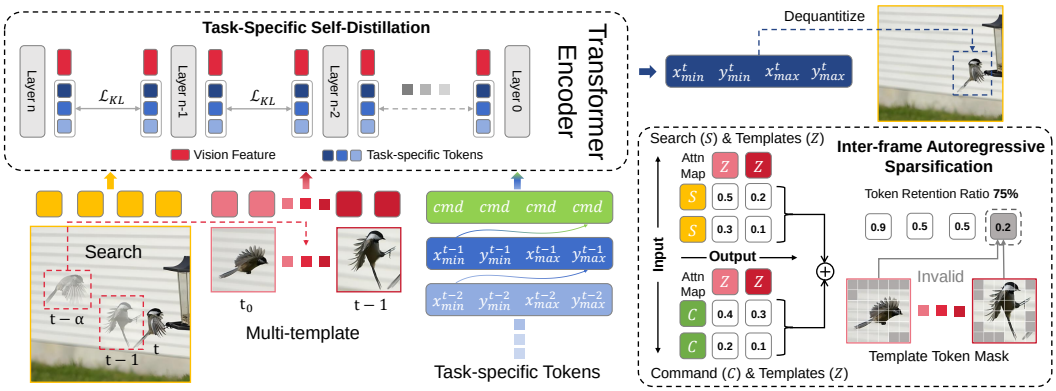


Figure 3: **FARTrack Framework.** FARTrack is a fast, high-performance multi-template autoregressive framework, comprising two key components: Task-Specific Self-Distillation for model compression and Inter-frame Autoregressive Sparsification for template sequences.

the Transformer Encoder. Finally, the Transformer encoder models the trajectory evolution in an autoregressive manner by leveraging the preceding trajectory tokens.

Multi-templates. To enhance tracking accuracy, we employ a multi-template design, further supported by a linear update strategy. To prevent the potential loss of temporal information caused by occlusion or disappearance of the target during the tracking process, we ensure that the updated multi-templates always include the first frame and the preceding frame.

3.3 Task-Specific Self-Distillation

Knowledge distillation-based model compression reduces the model size, thereby improving tracking efficiency. However, current cross-layer distillation methods rely on suboptimal manual layer assignments Ahn et al. (2019); Cui et al. (2023); Shen et al. (2021); Tung & Mori (2019); Wu et al. (2024), disrupting hierarchical feature extraction and causing accuracy degradation in shallow models, while also neglecting temporal information in trajectory sequences.

To address this issue, we propose a simple yet effective model compression method known as task-specific self-distillation, as depicted in Figure 3. In our method, one model layer serves as the student layer and the corresponding next layer acts as the teacher layer, establishing layer-wise self-distillation Hou et al. (2024); Zhang et al. (2021; 2019) that inherently circumvents suboptimal manual layer assignments. Furthermore, our method operates on task-specific tokens which represent the object's trajectory sequences. The student layer is trained to fit the trajectory sequence features of the teacher layer by minimizing the KL divergence. Therefore, the temporal information in the trajectory sequences propagates backward among the layers, enabling the model to be distilled to a shallow level while maintaining accuracy. Ultimately, our method improves the tracking speed of the model while maintaining the tracking performance.

3.4 Inter-frame Autoregressive Sparsification

The template image contains target object features alongside persistent background and noise interference that reduces tracking efficiency. Traditional sparsification methods Chen et al. (2022b); Liang et al. (2022); Rao et al. (2021); Ye et al. (2022) prioritize runtime token elimination, but these approaches introduce redundant time overhead and frame-specific optimization rather than sequence-level processing, resulting in slower speed and lower performance.

To eliminate template redundancy and noise in the tracking process while achieving a temporally-global optimal sparsification strategy, we propose an inter-frame autoregressive sparsification, as visualized in Figure 3. After processing through the attention layers, we obtain the attention weights for template tokens with respect to search tokens and the four command tokens based on the attention map. This sequential processing method considers the template's correlation with both the search area and the predicted coordinates, respectively. We then simply add these two attention weights and retain the parts with the largest values based on the predefined token retention rate. This process leverages

the inter-frame correlations to mask the unnecessary background parts in the templates while retaining the key foreground parts. Subsequently, the sparsification results of the current frame are saved and propagated to subsequent frames in an autoregressive manner, thereby learning a temporally-global optimal sparsification strategy. Overall, our method requires no additional time-consuming processes and meanwhile preserves the temporal information, achieving faster speed and better performance.

Notably, masked tokens are excluded from processing to avoid distortion of normalization statistics. LayerNorm is exclusively applied to valid tokens, preventing improper scaling and shifting caused by statistical deviations that could undermine both inference stability and model performance.

3.5 Training and Inference

FARTrack is a fast tracker that enhances tracking efficiency by reducing model depth via task-specific self-distillation and eliminating redundant and noisy information from the templates using an inter-frame autoregressive sparsification.

Training. Similar to its predecessor, FARTrack undergoes both frame-level and sequence-level training Bai et al. (2024); Kim et al. (2022); Wei et al. (2023a); Liang et al. (2025). Initially, task-specific self-distillation is employed to progressively transfer trajectory sequence features from deeper layers to shallower layers, thereby reducing the model depth and achieving model compression. To ensure effective knowledge transfer, we minimize the KL divergence between the teacher and student layers during training. Building upon this model compression, we introduce an inter-frame autoregressive sparsification that computes a mask matrix based on inter-frame correlations and removes unimportant template background tokens according to a predefined token retention ratio.

Moreover, we incorporate the SIoU loss Gevorgyan (2022) to better capture the spatial correlation between the predicted and ground truth bounding boxes. For each video clip, the initial trajectory prompt is initialized using the object bounding box from the first frame and is propagated into subsequent frames in an autoregressive manner. The overall loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{SIoU}} + \lambda_2 \mathcal{L}_{\text{KL}}. \tag{1}$$

where $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{SIoU}}$ and $\mathcal{L}_{\text{KL}}$ denote the cross-entropy loss, SIoU loss and KL divergence, respectively. The λ values are used as weights to balance the contribution of each loss term.

Inference. During inference, the trajectory is initialized using the object bounding box in the first frame. The inter-frame sparsification method removes redundant and noisy tokens from templates, retaining and propagating these sparsification results through subsequent tracking processes to reduce computational complexity. The trajectory tokens are iteratively propagated into subsequent frames in an autoregressive manner. Unlike the training phase, since sparsification has already been performed on the templates, LayerNorm is applied to all tokens as usual.

4 Experiments

4.1 Implementation Details

The model is trained with 8 NVIDIA RTX A6000 GPUs. The inference speed is evaluated with NVIDIA TiTan Xp, Intel(R) Xeon(R) Gold 6230R CPU @ 3.00GHz, and Ascend 310B.

Model Variants. We trained five variants of FARTrack with different configurations as Table 2.

The tiny is a 15-layer encoder model. Nano distills the 15-layer encoder into 10 layers, and pico into 6 layers. Tiny matches AsymTrack-B on GPU while keeping AsymTrack-T CPU speed. Nano outperforms state-of-the-art trackers in key metrics across datasets. Pico outperforms MixFormerV2-S by 0.9% in AO on GOT-10k Huang et al. (2019), showing FARTrack's efficiency.

Training. To conduct a fair comparison with mainstream trackers, we carried out Frame-level Pretraining on the COCO2017 Lin et al. (2014) dataset. Subsequently, we performed Task-Specific Self-Distillation Training to compress our model. Finally, we conducted Inter-frame Autoregressive Sparsification Training to further accelerate our model. The detailed training process can be found in the supplementary material.

Table 1: State-of-the-art comparison on GOT-10k Huang et al. (2019), TrackingNet Muller et al. (2018), LaSOT Fan et al. (2019) and LaSOText Fan et al. (2020). Best in **bold**, second best underlined.

Methods	GPU	CPU	NPU	GOT-10k			TrackingNet			LaSOT			LaSOText		
	FPS	FPS	FPS	AO(%)	SR_{50}(%)	SR_{75}(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)	P_{Norm}(%)	P(%)
DiMP Bhat et al. (2019)	68	22	-	-	-	-	74.0	80.1	70.6	56.9	65.0	56.7	-	-	-
SiamFC++ Xu et al. (2020)	76	28	-	-	-	-	75.4	80.0	68.7	54.4	62.3	54.7	-	-	-
LightTrack Yan et al. (2021b)	59	27	-	61.1	71.0	54.3	72.5	77.8	69.5	53.8	60.5	53.7	-	-	-
TCTrack Cao et al. (2022)	42	29	-	66.2	75.6	61.0	74.8	79.6	73.3	60.5	69.3	62.4	-	-	-
FEAR Borsuk et al. (2022)	123	28	-	61.9	72.2	52.5	70.2	80.8	71.5	53.5	59.7	54.5	-	-	-
E.T. Track Blatter et al. (2023)	33	16	-	56.6	64.6	42.5	72.5	77.8	69.5	59.1	66.8	60.1	-	-	-
HiT-Tiny Kang et al. (2023)	135	42	56	52.6	59.3	42.7	74.6	78.1	68.8	54.8	60.5	52.9	35.8	-	-
HiT-Small Kang et al. (2023)	121	35	47	62.6	71.2	54.4	77.7	81.9	73.1	60.5	68.3	61.5	40.4	-	-
HiT-Base Kang et al. (2023)	116	30	33	64.0	72.1	58.1	80.0	84.4	77.3	64.6	**73.3**	**68.1**	44.1	-	-
MixformerV2 Cui et al. (2023)	133	31	35	61.9	71.7	51.3	75.8	81.1	70.4	60.6	69.9	60.4	43.6	-	46.2
AsymTrack-T Zhu et al. (2025)	145	55	-	62.3	71.3	54.7	76.2	80.9	71.6	60.8	68.7	61.2	42.5	-	-
AsymTrack-S Zhu et al. (2025)	136	48	-	65.5	74.8	58.9	77.9	82.2	74.0	62.8	71.2	64.8	43.3	-	-
AsymTrack-B Zhu et al. (2025)	135	32	-	67.7	76.6	61.4	80.0	84.5	77.4	**64.7**	73.0	67.8	44.6	-	-
$FARTrack_{pico}$	**343**	**121**	**101**	62.8	72.6	50.9	75.6	81.3	70.5	58.6	67.1	59.6	41.8	50.8	44.4
$FARTrack_{nano}$	210	77	61	69.9	81.2	61.4	79.1	84.5	75.6	61.3	69.7	64.1	43.8	53.3	47.8
$FARTrack_{tiny}$	135	53	42	**70.6**	**81.0**	**63.8**	**80.7**	**85.6**	**77.5**	63.2	71.6	66.7	**45.0**	**54.0**	**49.2**

4.2 Main Results

We evaluated the performance of our proposed $FARTrack_{tiny}$, $FARTrack_{nano}$, and $FARTrack_{pico}$ on several benchmarks, including GOT-10k Huang et al. (2019), TrackingNet Muller et al. (2018), LaSOT Fan et al. (2019), LaSOText Fan et al. (2020) and VastTrack Peng et al. (2024).

GOT-10k Huang et al. (2019). GOT-10k is a real world general object detection dataset. As shown in Table 1, $FARTrack_{tiny}$ outperforms AsymTrack-B by 2.9% in AO score, achieving a GPU speed of 135 FPS and a CPU speed of 53 FPS. Furthermore, the most lightweight version, $FARTrack_{pico}$, outperforms MixFormerV2-S by 0.9% in AO, while delivering nearly three times the GPU speed and four times the CPU speed.

TrackingNet Muller et al. (2018). TrackingNet is a large-scale dataset featuring over 30,000 videos from diverse real world scenes. The evaluation on this extensive dataset highlights the efficiency and effectiveness of FARTrack. As illustrated in Table 1, $FARTrack_{nano}$ achieves performance close to that of AsymTrack-B, the top-performing tracker on this benchmark, while running nearly twice as fast as AsymTrack-B on the GPU.

LaSOT Fan et al. (2019). LaSOT is a large-scale benchmark designed to assess the robustness of long-term tracking. As demonstrated in Table 1, $FARTrack_{tiny}$ achieves AUC of 2.6% over MixFormerV2-S on this dataset. While maintaining a comparable running speed on the GPU, it nearly doubles the speed on the CPU. $FARTrack_{nano}$ matches AUC of MixFormerV2-S but surpasses it in both GPU and CPU running speed.

LaSOText Fan et al. (2020). LaSOText is an extended subset of LaSOT, encompassing 150 additional videos from 15 new categories. As shown in Table 1, $FARTrack_{tiny}$ outperforms AsymTrack-B, with a 0.4% AUC improvement, demonstrating its effectiveness in small object tracking.

VastTrack Peng et al. (2024). VastTrack is a dataset aimed at advancing the development of more general visual tracking technology, covering 2,115 object categories and containing 50,610 video sequences. As shown in Table 3, on this dataset, $FARTrack_{tiny}$ achieves an AUC comparable to that of MixFormerV2-B, which fully highlights the robustness of FARTrack.

4.3 Experimental Analyses

We analyze the main properties of the FARTrack. For the following experimental studies, we follow the GOT-10k test protocol unless otherwise noted. Default settings are marked in gray.

Table 2: Details of our FARTrack model variants

Model	$\text{FARTrack}_{\text{tiny}}$	$\text{FARTrack}_{\text{nano}}$	$\text{FARTrack}_{\text{pico}}$
Backbone	ViT-Tiny	ViT-Tiny	ViT-Tiny
Encoder Layers	15	10	6
Input Sizes	[112,224]	[112,224]	[112,224]
Templates	5	5	5
MACs (G)	2.65	1.78	1.08
Params (M)	6.82	4.59	2.81

Table 3: Comparison on more benchmarks.

Methods	VastTrack		
	AUC(%)	P_{Norm}(%)	P(%)
$\text{FARTrack}_{\text{tiny}}$	**35.2**	**36.5**	<u>32.3</u>
$\text{FARTrack}_{\text{nano}}$	<u>33.9</u>	<u>35.1</u>	30.3
$\text{FARTrack}_{\text{pico}}$	30.3	31.0	25.7
MixformerV2-B	**35.2**	**36.5**	**33.0**
DiMP	29.9	31.7	25.7

Table 4: vs. Deep-to-Shallow Distillation Cui et al. (2023).

Methods	Layer	AO	SR_{50}	SR_{75}
layer-by-layer	10	69.9	81.2	61.4
	6	62.8	72.6	50.9
deep-to-shallow	10	67.8	78.0	60.6
	6	61.9	70.9	50.4

Table 5: Sparsification Comparison.

run-time	sequence-time	MACs	Params	CPU	GPU	AO
		2.99G	6.82M	49	128	70.0
✓		3.14G	7.21M	36	114	69.5
	✓	**2.65G**	**6.82M**	**53**	**135**	**70.6**

Distillation Strategy. To validate the advantages of our method, we compared layer-by-layer distillation with cross-layer distillation. In cross-layer distillation, student-teacher layer correspondence exhibits an intermittent pattern rather than one-to-one alignment. While facilitating knowledge transfer, this approach introduces feature consistency challenges. Additionally, given ViT-Tiny's weaker representational capacity than ViT-Base, we further compared ViT-Base-guided FARTrack distillation to demonstrate layer-by-layer distillation's superiority.

As shown in Table 4, we manually designed a model layer reduction strategy with reference to the Deep-to-Shallow distillation method of MixformerV2, where *REMOVE_LAYERS* for the distillation of the 10-layer and 6-layer models are set to [0, 3, 6, 9, 12] and [0, 2, 4, 6] respectively. By removing model layers, the remaining layers can perform inter-layer matching in sequence. However, hand-crafted layer assignments fail to ensure reasonable inter-layer matching, leading to semantic mismatch in cross-layer distillation. This mismatch disrupts feature alignment, resulting in more significant accuracy degradation in deeper layers. In contrast, our method maintains the consistency of feature representations and effectively mitigates the issue of semantic mismatch.

As illustrated in Figure 4(a), FARTrack with ViT-Base shares an identical hierarchical structure with $\text{FARTrack}_{\text{tiny}}$. We distill each layer of ViT-Tiny using trajectory features from corresponding ViT-Base layers. Although the 15-layer model distilled via base-to-tiny method performs well, forcing Tiny to mimic Base's layer outputs causes significant accuracy degradation in deeper layers (10-14) and progressive decline in shallower ones, due to their representational capability mismatch. In contrast, our method maintains feature consistency and information density, preserving nearly identical accuracy in layers 10-15 while limiting shallow layers' accuracy loss within acceptable bounds.

Distillation Target. To verify the necessity of distilling the trajectory sequence, we conducted an exploratory experiment, as shown in Figure 4(b). Experiments reveal that directly distilling the search/template or jointly distilling visual features disrupts the hierarchical structure, causing accuracy degradation across all layers. In contrast, distilling the trajectory sequence minimally impacts the feature extraction hierarchy. Its autoregressive properties enhance knowledge propagation from deep to shallow layers, maintaining nearly unchanged accuracy in layers 10-15 while preventing significant performance drops in shallow layers.

Distillation Loss. Experiments (Figure 4(c)) demonstrate that combining trajectory sequence loss ($\mathcal{L}_{\text{traj}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{SIoU}}$) and KL divergence loss ($\mathcal{L}_{\text{KL}}$) effectively preserves accuracy in deep layers (*e.g.*, layers 10-15) while controlling performance degradation in shallow layers. Removing trajectory sequence loss causes significant feature degradation and performance drop, proving its critical role in preserving temporal information. In contrast, omitting KL divergence loss triggers rapid feature collapse in shallow layers, leading to catastrophic degradation of tracking performance, highlighting its indispensability for maintaining object tracking capability.

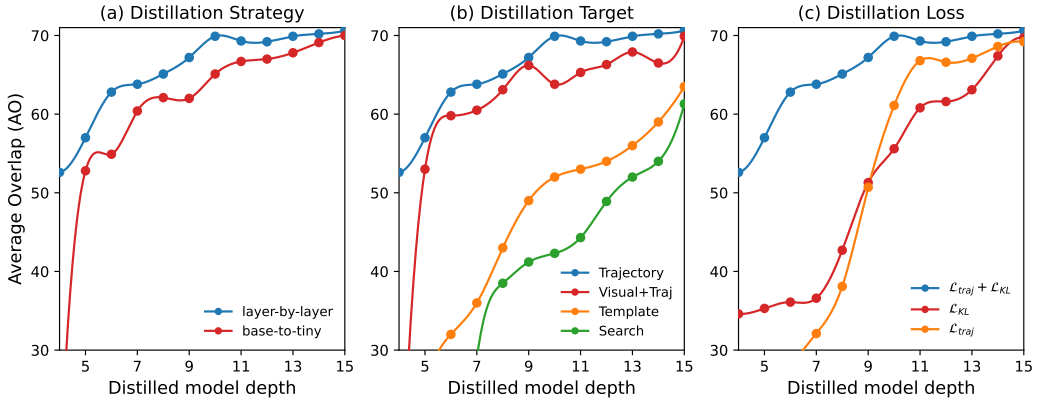


Figure 4: Layer-by-layer distillation accuracy curve.

Table 6: Token Retention Ratio.

Ratio	MACs	CPU	GPU	AO	SR_{50}	SR_{75}
100%	2.99G	49	128	70.0	80.0	64.4
75%	**2.65G**	**53**	**135**	**70.6**	**81.0**	**63.8**
50%	2.35G	56	139	68.3	78.2	61.8
25%	2.01G	58	140	67.3	78.3	59.8
10%	1.82G	63	141	62.3	74.1	53.3

Table 7: Attention Map Sampling Method.

$S_{1\times1}$	$S_{3\times3}$	S_{all}	Z	C	AO	SR_{50}	SR_{75}
					70.1	80.6	63.3
✓					69.5	80.4	62.4
	✓				70.0	80.4	62.9
		✓			68.6	79.1	61.7
	✓		✓		69.4	79.7	62.1
	✓			✓	**70.6**	**81.0**	**63.8**

Sparsification Comparison. The runtime sparsification introduces extra computational processes during the inference, while our sequence-level sparsification avoids this situation. To support this, we conducted experiments on runtime sparsification and sequence-level sparsification respectively based on the base model, and compared the final results, as shown in Table 5.

Runtime sparsification processes template tokens during each forward propagation in inference, introducing extra computational overhead that increases MACs from 2.99G to 3.14G and Params from 6.82M to 7.21M. This redundant computation reduces CPU and GPU speeds by 26.5% and 10.9% respectively. In contrast, our sequence-level sparsification leverages intermediate results for decision-making without introducing additional computations, while propagating sparsification outcomes to subsequent frames autoregressively to eliminate redundant processing. Consequently, MACs are reduced to 2.65G with improved inference speeds on both CPU and GPU.

Token Retention Ratio. Token retention ratio in inter-frame autoregressive sparsification affects both performance and efficiency. As Table 6 shows, reducing the ratio from 100% to 75% decreased MACs by 11.4% (2.99G to 2.65G) while achieving peak AO (70.6%). This indicates significant redundancy in target templates—removing 25% background tokens preserves temporal modeling and improves accuracy. Further reduction to 25% caused 3.3% AO drop (67.3%), demonstrating excessive removal harms tracking robustness.

Attention Map Sampling Method. We analyze different sampling strategies for inter-frame autoregressive sparsification in Table 7. $S_{1\times1}$ sparsifies the central 1×1 feature, $S_{3\times3}$ uses a 3×3 central region, S_{all} covers the entire search area. Z denotes template self-attention, while C refers to command-template attention.

$S_{3\times3}$ avoids target exclusion from center shift or hollow structures and reduces background overfocus versus S_{all}, improving accuracy. Combining S and Z underperforms due to self-overfocus in Z. Instead, integrating S and C enables effective template sparsification: S separates foreground and background coarsely, while C refines edge features. This reduces redundancy and improves accuracy and efficiency.

Layer-wise cross-attention visualization. As shown in Figure 5, cross-attention between trajectory sequences and search regions evolves across layers: shallow layers (1–5) capture edges and back-

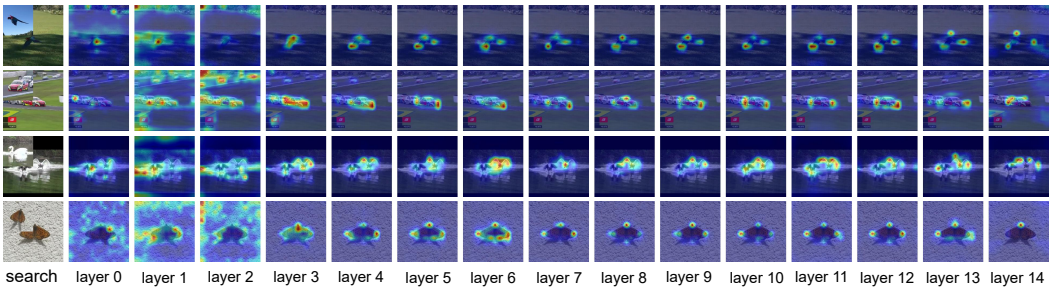


Figure 5: **Layer-wise cross-attention visualization.** search: Search region and template. layer 0-14: Trajectory sequences to search cross-attention maps at each layer.

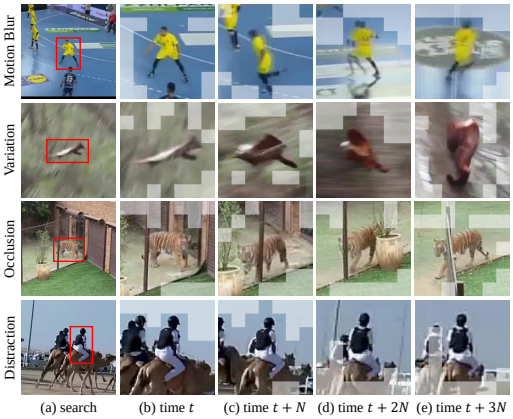


Figure 6: **Template sparsification visualization.** (a): Search region. The red boxes denote the ground truth. (a)-(e): Templates sampled at a fixed interval of N and sparsified with a 75% token retention ratio.

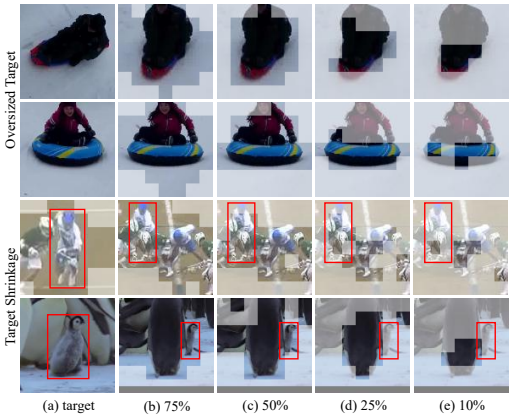


Figure 7: **Template retention visualization.** (a): The initial template of the video sequence. The red boxes denote the ground truth. (b)–(e): The template at time step t, sparsified with different token retention ratios.

ground, while deeper ones (10–15) transition toward target contours. Hierarchical feature learning is retained via distillation, maintaining consistent trajectory propagation and boosting accuracy, especially in intermediate layers (*e.g.*, 6, 10).

Template sparsification visualization. Figure 6 shows that our method retains critical tokens and removes redundancy under motion blur, appearance change, and occlusion. Unlike frame-wise sparsification, which often fails due to single-frame errors, our inter-frame autoregressive approach uses multi-template complementarity and temporal modeling to track dynamic targets and preserve structure even with inconsistent cues.

Template retention visualization. Figure 7 visualizes retained tokens at different retention ratios. For oversized targets, low ratios (<50%) lose essential features; for shrinking ones, they harm deformation representation and cause misclassification. Thus, we set a 75% retention ratio to preserve sufficient target information while reducing redundancy, balancing accuracy and efficiency.

5 Conclusion

We propose FARTrack, a fast and high-performance multi-template autoregressive tracking framework. It integrates task-specific self-distillation and inter-frame autoregressive sparsification. While slightly behind top-performing methods Bai et al. (2024); Wei et al. (2023a), it excels in speed-performance balance, especially in speed. Our distillation preserves temporal information of trajectory sequences via layer-wise task-specific tokens distillation, avoiding suboptimal manual layer assignments. The sparsification method propagates multi-template sparsification results autoregressively, achieving temporally-global optimality without extra cost. FARTrack performs well across GPU, CPU, and NPU platforms, offering a efficient solution for practical deployment.

REFERENCES

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *CVPR*, 2019.

Takanori Asanomi, Kazuya Nishimura, and Ryoma Bise. Multi-frame attention with feature-level warping for drone crowd tracking. In *WACV*, 2023.

Yifan Bai, Zeyang Zhao, Yihong Gong, and Xing Wei. Artrackv2: Prompting autoregressive tracker where to look and how to describe. In *CVPR*, 2024.

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019.

Philippe Blatter, Menelaos Kanakis, Martin Danelljan, and Luc Van Gool. Efficient visual tracking with exemplar transformers. In *WACV*, 2023.

Vasyl Borsuk, Roman Vei, Orest Kupyn, Tetiana Martyniuk, Igor Krashenyi, and Jiři Matas. Fear: Fast, efficient, accurate and robust visual tracker. In *ECCV*, 2022.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Hiptrack: Visual tracking with historical prompts. In *CVPR*, 2024.

Yidong Cai, Jie Liu, Jie Tang, and Gangshan Wu. Robust object modeling for visual tracking. In *ICCV*, 2023.

Anjia Cao, Xing Wei, and Zhiheng Ma. Flame: Frozen large language models enable data-efficient language-image pre-training. In *CVPR*, 2025.

Ziang Cao, Ziyuan Huang, Liang Pan, Shiwei Zhang, Ziwei Liu, and Changhong Fu. Tctrack: Temporal contexts for aerial tracking. In *CVPR*, 2022.

Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv*, 2021.

Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey E Hinton. A unified sequence interface for vision tasks. In *NeurIPS*, 2022a.

Xin Chen, Ben Kang, Dong Wang, Dongdong Li, and Huchuan Lu. Efficient visual tracking via hierarchical cross-attention transformer. In *ECCV*, 2022b.

Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *CVPR*, 2023.

Yutao Cui, Tianhui Song, Gangshan Wu, and Limin Wang. Mixformerv2: Efficient fully transformer tracking. In *NeurIPS*, 2023.

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020.

Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019.

Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Harshit, Mingzhen Huang, Juehuan Liu, Yong Xu, Chunyuan Liao, Lin Yuan, and Haibin Ling. Lasot: A high-quality large-scale single object tracking benchmark. *IJCV*, 2020.

Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. Aiatrack: Attention in attention for transformer visual tracking. In *ECCV*, 2022.

Shenyuan Gao, Chunluan Zhou, and Jun Zhang. Generalized relation modeling for transformer tracking. In *CVPR*, 2023.

Zhora Gevorgyan. Siou loss: More powerful learning for bounding box regression. *arXiv*, 2022.

Goutam Yelluru Gopal and Maria A. Amer. Separable self and mixed attention transformers for efficient object tracking. In *WACV*, 2024a.

Goutam Yelluru Gopal and Maria A Amer. Separable self and mixed attention transformers for efficient object tracking. In *WACV*, 2024b.

Jianyuan Guo, Kai Han, Yunhe Wang, Han Wu, Xinghao Chen, Chunjing Xu, and Chang Xu. Distilling object detectors via decoupled features. In *CVPR*, 2021.

Kaijie He, Canlong Zhang, Sheng Xie, Zhixin Li, and Zhiwen Wang. Target-aware tracking with long-term context attention. In *AAAI*, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 2015.

Lingyi Hong, Shilin Yan, Renrui Zhang, Wanyun Li, Xinyu Zhou, Pinxue Guo, Kaixun Jiang, Yiting Chen, Jinglun Li, Zhaoyu Chen, and Wenqiang Zhang. Onetracker: Unifying visual object tracking with foundation models and efficient tuning. In *CVPR*, 2024.

Lingyi Hong, Jinglun Li, Xinyu Zhou, Shilin Yan, Pinxue Guo, Kaixun Jiang, Zhaoyu Chen, Shuyong Gao, Runze Li, Xingdong Sheng, et al. General compression framework for efficient transformer object tracking. In *ICCV*, 2025.

Xiaojun Hou, Jiazheng Xing, Yijie Qian, Yaowei Guo, Shuo Xin, Junhao Chen, Kai Tang, Mengmeng Wang, Zhengkai Jiang, Liang Liu, and Yong Liu. Sdstrack: Self-distillation symmetric adapter learning for multi-modal visual object tracking. In *CVPR*, 2024.

Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *PAMI*, 2019.

Ben Kang, Xin Chen, Dong Wang, Houwen Peng, and Huchuan Lu. Exploring lightweight hierarchical vision transformers for efficient visual tracking. In *ICCV*, 2023.

Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017.

Minji Kim, Seungkwan Lee, Jungseul Ok, Bohyung Han, and Minsu Cho. Towards sequence-level training for visual tracking. In *ECCV*, 2022.

Yutong Kou, Jin Gao, Bing Li, Gang Wang, Weiming Hu, Yizheng Wang, and Liang Li. Zoomtrack: Target-aware non-uniform resizing for efficient visual tracking. In *NeurIPS*, 2023.

Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

Quanquan Li, Shengying Jin, and Junjie Yan. Mimicking very efficient network for object detection. In *CVPR*, 2017.

Xin Li, Yuqing Huang, Zhenyu He, Yaowei Wang, Huchuan Lu, and Ming-Hsuan Yang. Citetracker: Correlating image and text for visual tracking. In *ICCV*, 2023.

Shiyi Liang, Yifan Bai, Yihong Gong, and Xing Wei. Autoregressive sequential pretraining for visual tracking. In *CVPR*, 2025.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv*, 2022.

Liting Lin, Heng Fan, Zhipeng Zhang, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. In *NeurIPS*, 2022.

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 *ECCV*, 2014.

Zhiheng Ma, Anjia Cao, Funing Yang, Yihong Gong, and Xing Wei. Curriculum dataset distillation. *IEEE Transactions on Image Processing*, 2025.

Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *CVPR*, 2022.

Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018.

Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Heterogeneous knowledge distillation using information flow modeling. In *CVPR*, 2020.

Liang Peng, Junyuan Gao, Xinran Liu, Weihong Li, Shaohua Dong, Zhipeng Zhang, Heng Fan, and Libo Zhang. Vasttrack: Vast category visual object tracking. *NeurIPS*, 2024.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv*, 2014.

Jianbing Shen, Yuanpei Liu, Xingping Dong, Xiankai Lu, Fahad Shahbaz Khan, and Steven Hoi. Distilled siamese networks for visual tracking. *PAMI*, 2021.

Qiuhong Shen, Lei Qiao, Jinyang Guo, Peixia Li, Xin Li, Bo Li, Weitao Feng, Weihao Gan, Wei Wu, and Wanli Ouyang. Unsupervised learning of accurate siamese tracking. In *CVPR*, 2022.

Liangtao Shi, Bineng Zhong, Qihua Liang, Ning Li, Shengping Zhang, and Xianxian Li. Explicit visual prompts for visual object tracking. In *AAAI*, 2024.

Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Transformer tracking with cyclic shifting window attention. In *CVPR*, 2022.

Zikai Song, Run Luo, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Compact transformer tracker with correlative masked modeling. In *AAAI*, 2023.

Yiming Sun, Fan Yu, Shaoxiang Chen, Yu Zhang, Junwei Huang, Yang Li, Chenhui Li, and Changbo Wang. Chattracker: Enhancing visual tracking performance via chatting with multimodal large language model. In *NeurIPS*, 2025.

Chuanming Tang, Kai Wang, Joost van de Weijer, Jianlin Zhang, and Yongmei Huang. Avitmp: A tracking-specific transformer for single-branch visual tracking. *IEEE Transactions on Intelligent Vehicles*, 2024.

Feng Tang and Qiang Ling. Ranking-based siamese visual tracking. In *CVPR*, 2022.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

Xiao Wang, Shiao Wang, Chuanming Tang, Lin Zhu, Bo Jiang, Yonghong Tian, and Jin Tang. Event stream-based visual object tracking: A high-resolution benchmark dataset and a novel baseline. In *CVPR*, 2024.

Qingmao Wei, Bi Zeng, Jianqi Liu, Li He, and Guotian Zeng. Litetrack: Layer pruning with asynchronous feature extraction for lightweight and efficient visual tracking. In *ICRA*, 2024.

Xing Wei, Yifan Bai, Yongchao Zheng, Dahu Shi, and Yihong Gong. Autoregressive visual tracking. In *CVPR*, 2023a.

Xing Wei, Anjia Cao, Funing Yang, and Zhiheng Ma. Sparse parameterization for epitomic dataset distillation. In *NeurIPS*, 2023b.

Qiangqiang Wu, Tianyu Yang, Ziquan Liu, Baoyuan Wu, Ying Shan, and Antoni B. Chan. Dropmae: Masked autoencoders with spatial-attention dropout for tracking tasks. In *CVPR*, 2023a.

Qiao Wu, Jiaqi Yang, Kun Sun, Chu'ai Zhang, Yanning Zhang, and Mathieu Salzmann. Mixcycle: Mixup assisted semi-supervised 3d single object tracking with cycle consistency. In *ICCV*, 2023b.

You Wu, Yongxin Li, Mengyuan Liu, Xucheng Wang, Xiangyang Yang, Hengzhou Ye, Dan Zeng, Qijun Zhao, and Shuiwang Li. Learning adaptive and view-invariant vision transformer with multi-teacher knowledge distillation for real-time uav tracking. *arXiv*, 2024.

Fei Xie, Chunyu Wang, Guangting Wang, Yue Cao, Wankou Yang, and Wenjun Zeng. Correlation-aware deep tracking. In *CVPR*, 2022.

Fei Xie, Lei Chu, Jiahao Li, Yan Lu, and Chao Ma. Videotrack: Learning to track objects via video transformer. In *CVPR*, 2023.

Jinxia Xie, Bineng Zhong, Zhiyi Mo, Shengping Zhang, Liangtao Shi, Shuxiang Song, and Rongrong Ji. Autoregressive queries for adaptive tracking with spatio-temporal transformers. In *CVPR*, 2024.

Daitao Xing, Nikolaos Evangeliou, Athanasios Tsoukalas, and Anthony Tzes. Siamese transformer pyramid networks for real-time uav tracking. In *WACV*, 2022.

Liang Xu, Zhiqing Guo, and Liejun Wang. Efficient hybrid linear self-attention based visual object tracking with lora. *Neurocomputing*, 2025.

Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 2020.

Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021a.

Bin Yan, Houwen Peng, Kan Wu, Dong Wang, Jianlong Fu, and Huchuan Lu. Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In *CVPR*, 2021b.

Dawei Yang, Jianfeng He, Yinchao Ma, Qianjin Yu, and Tianzhu Zhang. Foreground-background distribution modeling transformer for visual object tracking. In *ICCV*, 2023a.

Jinyu Yang, Shang Gao, Zhe Li, Feng Zheng, and Aleš Leonardis. Resource-efficient rgbd aerial tracking. In *CVPR*, 2023b.

Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *ECCV*, 2022.

Kaiyu Yue, Jiangfan Deng, and Feng Zhou. Matching guided distillation. In *ECCV*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv*, 2016.

Ram Zaveri, Shivang Patel, Yu Gu, and Gianfranco Doretto. Improving accuracy and generalization for efficient visual tracking. In *WACV*, 2025.

Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *CVPR*, 2022.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019.

Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *PAMI*, 2021.

Minghua Zhang, Qiuyang Zhang, Wei Song, Dongmei Huang, and Qi He. Promptvt: Prompting for efficient and accurate visual tracking. *IEEE TCSVT*, 2024.

Tianlu Zhang, Hongyuan Guo, Qiang Jiao, Qiang Zhang, and Jungong Han. Efficient rgb-t tracking via cross-modality distillation. In *CVPR*, 2023.

Haojie Zhao, Dong Wang, and Huchuan Lu. Representation learning for visual object tracking by masked appearance transfer. In *CVPR*, 2023.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Zhikai Li, Yibing Song, Kai Wang, Zhangyang Wang, and Yang You. A stitch in time saves nine: Small vlm is a precise guidance for accelerating large vlms. *arXiv*, 2024a.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. 2024b.

Wangbo Zhao, Jiasheng Tang, Yizeng Han, Yibing Song, Kai Wang, Gao Huang, Fan Wang, and Yang You. Dynamic tuning towards parameter and inference efficiency for vit adaptation. 2024c.

Yaozong Zheng, Bineng Zhong, Qihua Liang, Zhiyi Mo, Shengping Zhang, and Xianxian Li. Odtrack: Online dense temporal token learning for visual tracking. In *AAAI*, 2024.

Yaozong Zheng, Bineng Zhong, Qihua Liang, Ning Li, and Shuxiang Song. Decoupled spatio-temporal consistency learning for self-supervised tracking. In *AAAI*, 2025.

Li Zhou, Zikun Zhou, Kaige Mao, and Zhenyu He. Joint visual grounding and tracking with natural language specification. In *CVPR*, 2023a.

Xinyu Zhou, Pinxue Guo, Lingyi Hong, Jinglun Li, Wei Zhang, Weifeng Ge, and Wenqiang Zhang. Reading relevant feature from global representation memory for visual object tracking. In *NeurIPS*, 2023b.

Jiawen Zhu, Huayi Tang, Xin Chen, Xinying Wang, Dong Wang, and Huchuan Lu. Two-stream beats one-stream: Asymmetric siamese network for efficient visual tracking. In *AAAI*, 2025.

APPENDIX

A TRAINING DETAILS

In the supplementary material, we have supplemented the experiments section. We have sequentially presented the detailed training processes of Frame-level Pretraining, Task-Specific Self-Distillation Training, and Inter-frame Autoregressive Sparsification Training.

Frame-level Pretraining. To fairly compare with mainstream trackers, we introduce the COCO2017 Lin et al. (2014) dataset, which is commonly used in template matching training paradigms, and apply frame-level pretraining to the AR(0) model. Similar to DiMP Bhat et al. (2019) and STARK Yan et al. (2021a), the AR(0) model uses the same data augmentations as OSTrack, including horizontal flip and brightness jittering. The model is optimized using AdamW with a weight decay of 1×10^{-4}. The learning rate for the backbone is set to 4×10^{-4}, and 4×10^{-3} for other parameters. The AR(0) model is trained for 500 epochs, with 76,800 template-search frame pairs sampled per epoch. The learning rate is reduced by 10% at the 400th epoch.

Task-Specific Self-Distillation Training. This phase uses the same datasets and data augmentations as the AR(0) phase but introduces additional KL divergence loss ($\mathcal{L}_{\mathrm{KL}}$) and trajectory sequence loss ($\mathcal{L}_{\mathrm{traj}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{SIoU}}$) for each layer. The model is optimized using AdamW with a weight decay of 1×10^{-4}. The learning rate for the backbone is set to 4×10^{-5}, and 4×10^{-4} for other parameters. The distillation model is trained for 300 epochs, with 76,800 template-search frame pairs randomly sampled per epoch. This process generates multiple versions of the distilled model, producing models at different layers from a single distillation.

Inter-frame Autoregressive Sparsification Training. Unlike traditional per-frame template matching, this method trains FARTrack directly on continuous video sequences without applying data augmentations. The model is optimized using AdamW with a weight decay of 5×10^{-2}. The learning rate for the backbone is set to 4×10^{-7} and 4×10^{-6} for other parameters. The training process consists of 20 epochs, with 1,000 video slices randomly sampled from continuous video per epoch. Due to GPU memory limitations, each slice contains 32 frames.

B TEMPLATE QUANTITY

In this section, an ablation experiment is conducted on the number of templates. The number of templates not only affects the operation efficiency of the model but also impacts the tracking accuracy. Therefore, it is essential to explore this aspect.

Table 8: Impact of Templates on Efficiency and Performance.

Template Count	MACs	Params	CPU	GPU	AO	SR_{50}	SR_{75}
1	1.70G	6.82M	64	141	66.4	77.0	58.0
3	2.17G	6.82M	55	139	68.1	78.6	60.9
5	**2.65G**	**6.82M**	**53**	**135**	**70.6**	**81.0**	**63.8**
7	3.13G	6.82M	48	124	69.6	80.3	62.5
9	3.61G	6.82M	45	115	70.0	80.7	62.3

As shown in Table 8, when the number of templates gradually increases from 1 to 5, the AO gradually rises from 66.4% to 70.6%, and the MACs increases synchronously by 35.8% (from 1.70G to 2.65G). During this stage, adding every two templates can increase the AO by approximately 2.1%, which verifies that the multi-template mechanism can effectively aggregate the appearance changes of the target and achieve an accurate representation of the target's dynamic appearance. However, when the number of templates exceeds 5 (*e.g.*, 7-9), the AO instead drops to the range of 69.6%-70.0%. This indicates that redundant templates lead to the dispersion of attention weights and interfere with the extraction of key features.

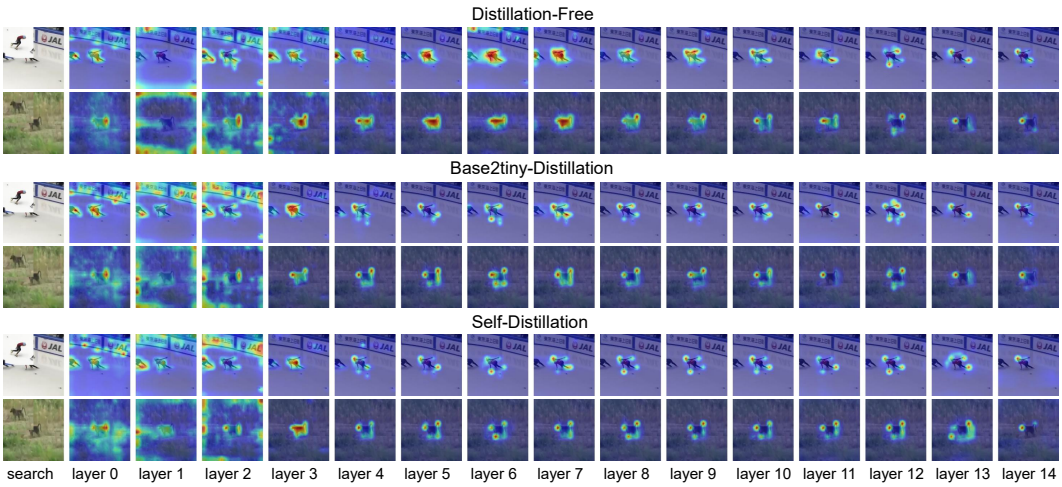


Figure 8: **Layer-wise cross-attention visualization Comparison.** search: Search region and template. layer 0-14: Trajectory sequences to search cross-attention maps at each layer.

Finally, we choose to use 5 templates because it achieves the optimal efficiency at the critical point of the accuracy peak.

C Template Update Strategy

In this section, an ablation experiment is conducted on the template update strategy. Different test benchmarks have an obvious correlation with different template update strategies. Therefore, it is necessary to experiment with different update strategies.

Table 9: Template Update Strategy Comparison.

Update Strategy	GOT - 10k			LaSOT		
	AO	SR_{50}	SR_{75}	AUC	P_{Norm}	P
Linear	70.9	81.6	63.4	62.5	70.2	65.3
Exponential Decay	**70.6**	**81.0**	**63.8**	**63.2**	**71.6**	**66.7**
Logarithmic Increasing	69.6	80.1	62.8	61.6	68.7	65.4
Equal-interval	69.5	79.4	63.1	62.1	69.3	65.5

As shown in Table 9, linear sampling achieves relatively mediocre values among all the update strategies. However, since it always samples all templates linearly, it is not sensitive to the length of the target sequence, and thus it has good performance on different datasets. Exponential decay sampling samples more frames closer to the current frame. This is more friendly to long sequences such as the LaSOT benchmark and also performs well in short video sequences like GOT-10k. Logarithmic increasing sampling samples more frames that are farther from the current frame, which makes it perform poorly in long video sequences such as LaSOT. Equal-interval sampling performs poorly both in GOT-10k and LaSOT. This is because equal-interval sampling may fail to sample the initial static template. And if there is a long period of tracking failure, such as the target disappearing or being occluded, it is very likely that all templates will become invalid, resulting in a decrease in accuracy.

D Inter-layer Attention Visualization Comparison

In this section, we conduct ablation experiments on the inter-layer attention of different distillation methods. Given the evident correlation between the distribution patterns of inter-layer attention and various distillation approaches, experimental validation is therefore deemed necessary.

For the layer-wise cross-attention visualized in Figure 8, the inter-layer attention distributions under the distillation-free setting exhibit diversity with distinct differences across various layers. Compared with self-distillation, the inter-layer attention distributions in the base2tiny distillation setting are more scattered. In contrast, the deep-layer attention distributions after trajectory self-distillation demonstrate consistency — a characteristic that confirms the effectiveness of the self-distillation mechanism in transferring deep-layer knowledge to middle layers (*e.g.*, layers 5–14). Thus, compared with other configurations, self-distillation enables more coherent inter-layer feature alignment.

E Additional Tracker Comparisons

We have supplemented the performance and complexity comparisons with the latest state-of-the-art (SOTA) methods published in 2024 and 2025. (*) denotes results on GOT-10k obtained following the official one-shot protocol. The speeds of both GPU and CPU were tested on the same hardware device.

Table 10: State-of-the-art comparison on GOT-10k Huang et al. (2019), TrackingNet Muller et al. (2018), LaSOT Fan et al. (2019) and NFS Kiani Galoogahi et al. (2017). Best in **bold**, second best underlined. (*) denotes results on GOT-10k obtained following the official one-shot protocol.

Methods	GPU	CPU	MACs	Params	GOT-10k			TrackingNet			LaSOT			NFS
	FPS	FPS	G	M	AO(%)	SR_{50}(%)	SR_{75}(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)
LiteTrack-B4(*) Wei et al. (2024)	195	29	6.78	26.18	65.2	74.7	57.7	79.9	84.9	76.6	62.5	72.1	65.7	63.4
PromptVT Zhang et al. (2024)	104	30	2.90	3.00	68.2	79.3	61.8	78	83.5	74.4	63.7	73.8	66.8	-
SMAT Gopal & Amer (2024b)	135	40	-	3.78	64.5	74.7	57.8	78.6	84.2	75.6	61.7	71.1	64.6	62.0
ECTTrack Xu et al. (2025)	104	46	-	-	65.6	75	60.7	78.8	84.6	76.5	62.4	71.5	66.3	61.1
CompressTracker-OSTrack-2 Hong et al. (2025)	207	48	6.38	21.24	-	-	-	78.2	83.3	74.8	60.4	68.5	61.5	-
SSTrack(*) Zheng et al. (2025)	62	18	32.55	92.12	67.1	76.6	59.1	80.1	86.7	78.9	**64.8**	**75.2**	**69.7**	-
AsymTrack-B Zhu et al. (2025)	135	32	1.81	3.36	67.7	76.6	61.4	80.0	84.5	77.4	64.7	73.0	67.8	64.4
$FARTrack_{pico}$ w/o VastT	-	-	1.08	2.81	61.5	71.8	50.2	76.1	80.9	70.5	58.4	66.6	58.9	-
$FARTrack_{nano}$ w/o VastT	-	-	1.78	4.59	68.9	79.5	60.9	80.0	85.0	76.1	61.1	68.5	63.4	-
$FARTrack_{tiny}$ w/o VastT	-	-	2.65	6.82	69.6	80.4	62.8	80.6	85.4	77.2	63.5	71.6	66.5	-
$FARTrack_{pico}$	**343**	**121**	1.08	2.81	62.8	72.6	50.9	75.6	81.3	70.5	58.6	67.1	59.6	62.0
$FARTrack_{nano}$	210	77	1.78	4.59	69.9	81.2	61.4	79.1	84.5	75.6	61.3	69.7	64.1	65.1
$FARTrack_{tiny}$	135	53	2.65	6.82	**70.6**	**81.0**	**63.8**	**80.7**	**85.6**	**77.5**	63.2	71.6	66.7	**66.9**

As can be seen from the Table 10, our FARTrack achieves state-of-the-art (SOTA) performance on the NFS/GOT-10k/TrackingNet benchmark and competitive accuracy on the LaSOT benchmarks. On the LaSOT benchmark, the AUC of SSTrack is 1.6% higher than that of our tiny model (64.8% > 63.2%). This is attributed to the fact that its parameter count is is much larger than that of ours ($\mathbf{92.12M} > \mathbf{6.82M}$), thus endowing SSTrack with stronger robustness in long-term temporal tracking. However, under the condition of comparable parameter counts, our model should deliver higher accuracy.

F Analysis of Independent Performance Gains of Each Module

To disentangle the gains of each Module, we supplement comparative experiments with ARTrack and ARTrackV2 (ARTrack, ARTrackV2, and FARTrack all adopt the ViT-Tiny architecture with 6 Encoder layers.), and the results are shown in Table 11.

Table 11: Performance Comparison with ARTrack and ARTrackV2.

Methods	Self-distillation	Sparsification	MACs	Params	GPU	CPU	GOT-10k		
			(G)	(M)	(FPS)	(FPS)	AO(%)	SR_{50}(%)	SR_{75}(%)
ARTrack_tiny(6)			2.39	13.56	68	34	59.9	69.6	50.4
ARTrackV2_tiny(6)			2.10	8.31	110	49	60.6	70.9	45.7
FARTrack_pico(6)		✓	1.08	2.81	343	121	60.6	69.5	46.3
FARTrack_pico(6)	✓		1.25	2.81	266	101	61.8	71.5	50.0
FARTrack_pico(6)	✓	✓	1.08	2.81	343	121	62.8	72.6	50.9

The training settings and datasets of ARTrack and ARTrackV2 are consistent with those of FARTrack. The table demonstrates the effectiveness of the self-distillation and sparsification modules: self-distillation (enabling middle layers to learn from deep-layer knowledge) improves AO by 1.2%

compared to ARTrackV2_tiny(6); sparsification (retaining useful foreground tokens while removing redundant background tokens) boosts AO by 1.0% (from 61.8 to 62.8).

G Comparative Analysis of Distillation vs. Scratch-Trained Models with Different Depths

We have supplemented comparative experiments between 10-layer or 6-layer models trained from scratch and our distilled versions to clarify whether distillation provides benefits beyond simply using fewer layers (see Table 12).

Table 12: Performance Comparison of Different FARTrack Variants and Their Corresponding Scratch-Trained Models.

Methods	GOT-10k			TrackingNet			LaSOT		
	AO(%)	SR_{50}(%)	SR_{75}(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)	P_{Norm}(%)	P(%)
FARTrack_tiny	70.6	81.0	63.8	80.7	85.6	77.5	63.2	71.6	66.7
FARTrack_nano(10)	69.9	81.2	61.4	79.1	84.5	75.6	61.3	69.7	64.1
FARTrack_scratch(10)	67.1	77.4	59.9	77.9	83.1	73.6	60.8	69.1	63.3
FARTrack_pico(6)	62.8	72.6	50.9	75.6	81.3	70.5	58.6	67.1	59.6
FARTrack_scratch(6)	60.6	69.5	46.3	73.3	78.4	67.1	56.8	64.1	55.7

The results confirm that distillation offers advantages far beyond simply using fewer layers:

(i) **10-layer or 6-layer models trained from scratch:** Suffer severe performance degradation (e.g., FARTrack_scratch(10) exhibits a 3.5% drop in AO compared to FARTrack_tiny). This is because such models lack the learning capacity of deep networks and fail to capture complex visual/temporal features required for tracking.

(ii) **Our distilled versions:** Transfer deep-layer knowledge to middle layers via self-distillation, retaining higher performance (FARTrack_nano(10) only sees a 0.7% AO drop compared to FARTrack_tiny) while achieving lower training costs than 10-layer or 6-layer models trained from scratch.

H Training Cost Comparison: Cross-Layer vs. Task-Specific Self-Distillation

We have supplemented the training cost comparison between cross-layer distillation (Deep-to-Shallow) Cui et al. (2023) and our task-specific self-distillation, with details as follows.

Both methods were trained on 8 NVIDIA RTX A6000 GPUs:

(i) Deep-to-Shallow distillation: Stage 1 (15-to-10 layers) takes about 40 hours; Stage 2 (10-to-6 layers) takes about 38 hours (two-stage sequential training required).

(ii) Task-specific self-distillation: Takes about 36 hours total, with one-time training enabling direct utilization of multiple model versions.

This comparison fully demonstrates the training efficiency advantage of our proposed method.

I The Use of Large Language Models (LLMs)

We only used LLMs minimally to aid or polish writing. For instance, when describing a concept, we might leverage LLMs to ensure terminological precision, enhance logical coherence, or optimize the academic tone of the exposition.