Joint Neural Architecture Search and Token Pruning for Efficient Visual Tracking

Yihong Chen[1] , **Shuo Wang**[1] , **Jiayao Zheng**[1] and **Yongqiang Bai**[1,*]
[1]School of Automation, Beijing Institute of Technology
{cyhoon, wwangshuo, zeyo, byfengyun}@bit.edu.cn

Abstract

Recently, transformer-based trackers have become the leading approach, surpassing traditional CNN-based trackers in accuracy. However, their high computational demands hinder their deployment on edge platforms, necessitating efficient solutions. To address this, we propose a novel Neural Architecture Search (NAS) framework designed for transformer-based trackers, named NASTrack. This framework incorporates token pruning to optimize both transformer block structures and the layer-wise token keeping ratio, striking a balance between performance and efficiency. To handle the larger search space introduced by the keeping ratio, we propose a blacklist strategy and a matching-based distillation driven by the Token Overlap Ratio (TOR). Our method discovers hundreds of high-performing trackers, with FLOPs ranging from 1G to 18G. The searched trackers consistently outperform existing efficient state-of-the-art trackers such as CompressTracker and LiteTrack under comparable computational budgets.The code and models are available at https://github.com/Cyhoon84/NASTrack.git.

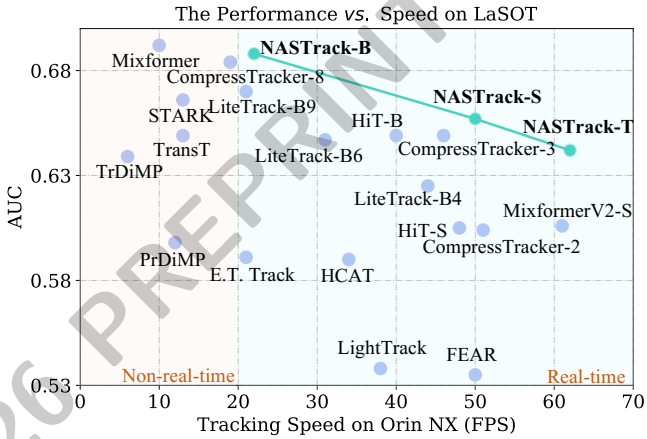

Figure 1: Performance comparisons with state-of-the-art trackers in terms of AUC and speed on LaSOT. The proposed NASTrack is superior to LiteTrack and CompressTracker family [Wei *et al.*, 2024a; Hong *et al.*, 2025], while achieving faster speed on Jetson Orin NX. Best viewed in color.

1 Introduction

Visual object tracking is a key task in computer vision that focuses on tracking a specific object throughout a video sequence based on its initial state. CNN-based [Bertinetto *et al.*, 2016; Li *et al.*, 2018] trackers once dominated the field, but the rise of transformer-based architectures [Dosovitskiy *et al.*, 2021] has redirected research focus due to their superior performance.

However, transformer-based trackers demand significant computational resources, which hinders their deployment on edge devices. Existing lightweight methods, including token pruning [Li *et al.*, 2023], CNN-transformer hybrids [Kang *et al.*, 2023] and distillation [Cui *et al.*, 2024], predominantly rely on heuristic manual designs. These methods also lack adaptability to diverse computational constraints, often requiring retraining for different platforms. Neural Architecture Search (NAS) automates the design of efficient architectures tailored to specific computational budgets, offering a promising alternative. While LightTrack [Yan *et al.*, 2021b] has successfully introduced Neural Architecture Search (NAS) to optimize CNN-based trackers, the application of NAS to transformer-based trackers remains unexplored, leaving a critical research gap in the field.

In this work, we present the first approach that integrates Neural Architecture Search (NAS) with transformer-based trackers to automatically design efficient models under varying computational constraints. Existing NAS methods [Guo *et al.*, 2020; Yu *et al.*, 2020; Chen *et al.*, 2021a; Gong *et al.*, 2022; Liu *et al.*, 2022] search for transformer architectures by optimizing model hyperparameters such as MLP ratio, number of heads and depth, but often sacrifice performance due to excessive network compression [Hou and Kung, 2022]. Our method expands the search space by incorporating token pruning, allowing NAS to jointly optimize transformer architectures and per-block keeping ratio (the proportion of tokens retained after pruning in each block).

*Corresponding author: Yongqiang Bai

This multi-dimensional search strategy mitigates architectural constraints, allowing flexible adaptation to resource limitations while identifying the optimal configuration in the search space.

Expanding the search space with the keeping ratio introduces non-trivial optimization challenges. Training a massive supernet often suffers from optimization interference among subnetworks of varying scales [Tang *et al.*, 2023]. We also find that the keeping ratio introduces sampling bias, making suboptimal pruning structures more likely to be sampled. Some pruning configurations lead to performance degradation regardless of architectural variations, such as excessive pruning in shallow layers. To mitigate this, we implement a progressive blacklist mechanism that identifies and excludes performance-collapsing pruning structures, thereby focusing the search on high-potential candidates.

To ensure subnetworks receive sufficient optimization, we propose a matching-based distillation strategy inspired by Cream of the Crop [Peng *et al.*, 2020]. Throughout the supernet training phase, we maintain an evolving "elite pool" of top-performing subnetworks to serve as potential teachers. To facilitate high-fidelity knowledge transfer, we introduce the Token Overlap Ratio (TOR)—a novel metric designed to quantify the structural and functional similarity between subnetworks based on their pruning patterns. By leveraging TOR, we dynamically pair each student subnetwork with the most compatible teacher in the elite pool, fostering collaborative convergence across the search space. Notably, this matching mechanism operates without auxiliary selection modules, thereby minimizing training overhead.

The efficacy of TOR as a proxy for teacher selection is grounded in our empirical analysis. By evaluating a vast ensemble of subnetworks, we observe that: (1) pairs with high TOR exhibit strong semantic consistency in their activation maps, and (2) TOR remains robust to variations in model scale, focusing instead on the alignment of spatial information. Drawing on the insights from DisWOT [Dong *et al.*, 2023], which correlates feature-level similarity with distillation success, our findings suggest that TOR effectively identifies architecturally synergistic teacher-student pairs, even within an expansive and heterogeneous search space.

Extensive experiments on benchmark datasets demonstrate that the trackers searched by our method outperform state-of-the-art efficient trackers in terms of accuracy and efficiency. From Fig. 1, we can see that our NASTrack family outperforms the LiteTrack and CompressTracker family in both speed and accuracy on LaSOT [Fan *et al.*, 2019]. Furthermore, our smallest variant, NASTrack-T (1.3 GFLOPs), surpasses MixformerV2-S (4.3 GFLOPs) and HiT-S (1.1 GFLOPs) by 3.6% and 3.8% AUC respectively. In terms of speed, our largest model can still run 21 FPS on Jetson Orin NX without TensorRT acceleration.

The main contributions of this work are as follows.

(1) First Integration of NAS & Transformer Trackers: We present a novel one-shot NAS framework specifically designed for transformer-based trackers, incorporating token pruning ratios into a unified search space.

(2) Robust Supernet Training: We introduce a blacklist mechanism to prune the search space of suboptimal configurations and a TOR-guided matching distillation to ensure balanced and sufficient training across the supernet.

(3) State-of-the-Art Efficiency: NASTrack achieves superior performance-efficiency trade-offs across multiple benchmarks, providing a versatile solution for edge-device deployment.

2 Related Work

Efficient Transformer-based Tracker. The powerful feature modeling capabilities of transformers have significantly enhanced the performance of transformer-based trackers [Ye *et al.*, 2022; Yan *et al.*, 2021a; Chen *et al.*, 2021b; Cui *et al.*, 2022; Chen *et al.*, 2023; Xie *et al.*, 2024], surpassing traditional CNN-based trackers [Bertinetto *et al.*, 2016; Li *et al.*, 2018] by a large margin. However, due to the inherently slow inference speed of transformers, these high-performance trackers often struggle to be deployed on resource-constrained edge devices. To accelerate transformer-based trackers while maintaining competitive performance, researchers have explored various efficient tracking architectures. HiT [Kang *et al.*, 2023] employs a hybrid CNN-transformer architecture as the baseline, integrating it into tracking tasks. Aba-ViT [Li *et al.*, 2023] proposes an adaptive and background-aware token computation strategy that identifies redundant background tokens and discards them to decrease overall computational overhead. LiteTrack [Wei *et al.*, 2024a], MixFormerV2 [Cui *et al.*, 2024] and CompressTracker [Hong *et al.*, 2025] adopt a top-down pruning strategy to progressively reduce the number of layers in the network. Although these approaches effectively improve tracking speed, they heavily rely on human expertise. Additionally, they lack the flexibility to adjust the network structure based on varying computational constraints.

Neural Architecture Search. Neural Architecture Search (NAS) has emerged as a transformative paradigm for automating the design of efficient architectures under stringent computational budgets. Early frameworks leveraged evolutionary algorithms [Xie and Yuille, 2017; Real *et al.*, 2019] or reinforcement learning [Pham *et al.*, 2018] to navigate the vast design space, albeit at prohibitive training costs. To enhance search efficiency, weight-sharing mechanisms [Guo *et al.*, 2020; Liu *et al.*, 2019] have become the de facto standard, enabling gradient-based optimization [Liu *et al.*, 2019] or posterior evolutionary searches to identify architectures along the Pareto frontier

Recently, the scope of NAS has expanded to Vision Transformers (ViTs). A notable pioneer, Autoformer [Chen *et al.*, 2021a], introduced a specialized search space encompassing width, depth, MLP ratios, and embedding dimensions. It leverages weight entanglement to address the challenges of weight sharing in ViTs, where convergence can be problematic. Like CNN-based NAS [Peng *et al.*, 2020; Yu *et al.*, 2020; Howard *et al.*, 2019], transformer-based NAS faces the classic issue of one-shot NAS: *how to improve the correlation between models using shared weights and those retrained from scratch.* Some methods [Yu *et al.*, 2020; Tang *et al.*, 2023; Gong *et al.*, 2022] leverage the sandwich rule technique to enhance the overall performance of the su-

	Supernet$_{tiny}$	Supernet$_{small}$	Supernet$_{base}$
Layer Num	(12, 10)	(12, 10)	(12, 10, 8)
Embed Dim	(128, 192)	(320, 384)	(640, 704, 768)
Q-K-V Dim	(128, 192)	(320, 384)	(640, 704, 768)
Head Num	(2, 3)	(5, 6)	(10, 11, 12)
MLP Ratio	(3.0, 3.5, 4.0)	(3.0, 3.5, 4.0)	(3.0, 3.5, 4.0)
Keeping Ratio	(1, 0.8, 0.7, 0.6)	(1, 0.8, 0.7, 0.6)	(1, 0.8, 0.7, 0.6)
Params Range	4 - 8M	16 - 26M	55 - 93M

Table 1: The search space of NASTrack. We set up three supernets to satisfy different resource constraints.

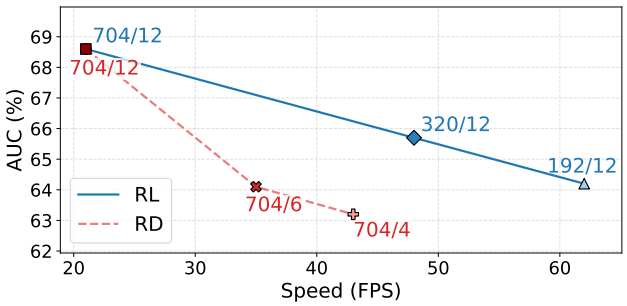

Figure 2: Comparison of Performance based on Layer and Dimension Retention Strategies on LaSOT. 'RL/RD': a model configured by retaining layers/dimensions, '192/12': dimension/number of layers.

pernet. Some approaches [Zhang *et al.*, 2023; Liu *et al.*, 2022] prioritize the discovery of Pareto-optimal structures during training.

However, existing methods focus on the transformer block structure, neglecting the self-attention mechanism's capacity for variable-length sequences. This often leads to excessive compression of architecture, such as layer reduction, to meet cost constraints, which compromises the model's representational capacity and degrades performance.

Token Sparsification. Since the computational complexity of Vision Transformers (ViTs) is closely related to the number of input tokens, token sparsification reduces computational cost by discarding tokens with lower information during inference. Token sparsification methods can be categorized into two types: token pruning [Kong *et al.*, 2022; Yin *et al.*, 2022] and token merging [Bolya *et al.*, 2023; Marin *et al.*, 2023]. Pruning-based methods cut tokens directly, while merging-based methods combine similar ones. Some approaches [Liang *et al.*, 2022] integrate both strategies to enhance efficiency. Despite the diversity of existing methods, most approaches apply token reduction at predefined stages. However, the optimal reduction pattern varies across models with different capacities, making it challenging to determine the best pruning locations. To address this issue, our method incorporates the keeping ratio into the NAS search space, allowing the optimal keeping ratio to be automatically determined for each transformer block. This eliminates the need for manual trial-and-error tuning, significantly reducing training overhead.

3 Method

3.1 Token Pruning Optimization via NAS

Existing methods focus on improving the selection of tokens in each layer for pruning, but they often maintain fixed keeping ratio and predetermined pruning depths. Since the optimal reduction pattern differs across models of varying capacities, it is challenging to define a universal rule for the number of tokens that can be pruned in each layer. We incorporate the keeping ratio into the search space and utilize NAS to automatically find the optimal keeping ratio for each layer. We divide the search space into three parts based on the range of model sizes, as shown in Tab.1. There are four different pruning options, each representing the percentage of tokens retained in a transformer block.

We deliberately avoided excessive reduction in the number of layers. We found that reducing head numbers and embedding dimensions provides more substantial model acceleration and performance preservation for tracking, proving more effective than layer reduction. Additionally, fewer layers diminishes the speed benefits attainable from token pruning. Fig. 2 demonstrates the superiority of dimension reduction over layer reduction in terms of balancing speed and performance.

3.2 Search Space Shrinking

Integrating keeping ratios into the search space, while flexible, often incurs significant training inefficiencies. This stems from a pronounced sampling bias: as illustrated in Fig. 4, the distribution of the cumulative keeping ratio (the product across all layers) varies drastically with network depth. In deeper architectures, the search space naturally gravitates toward small cumulative ratios. For instance, in a 12-layer network, the supernet frequently samples configurations with a product near 0.0345, retaining approximately 9 tokens (0.0345×256) by the final transformer block. For dense prediction tasks like object tracking, such excessive pruning severely undermines the model's discriminative capacity in complex scenarios.

Consequently, high-performance candidates often reside in low-probability regions of the sampling distribution, leading to a disproportionate allocation of computational resources toward suboptimal or even "broken" structures. To mitigate this, we propose a Blacklist Strategy that iteratively identifies and eliminates ineffective pruning configurations throughout the supernet training process. By dynamically pruning the search space itself, our approach steers the optimization toward high-capacity regions, ensuring that training resources are concentrated on architectures with superior representational potential.

Specifically, during supernet training, when the $\text{IoU}_{\text{train}}$ of a sampled network α falls below a predefined threshold T_{IoU}, we record its pruning configuration and evaluate its performance on the validation set to obtain IoU_{val}. If IoU_{val} is sufficiently low, the corresponding pruning configuration is added to a blacklist, and it will not be sampled in subsequent epochs.

However, network architecture also significantly impacts performance. For example, an overly simplified network structure, even with a reasonable pruning strategy, can result

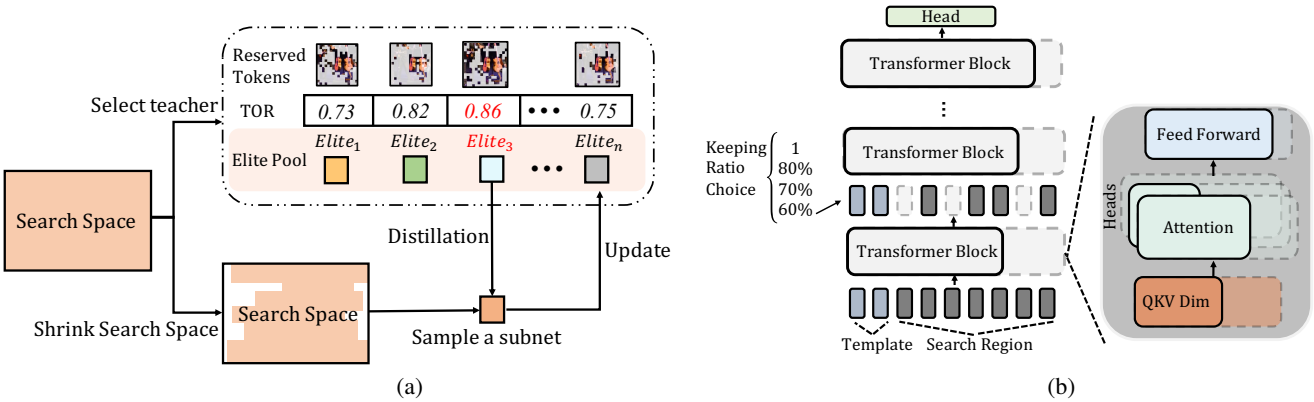

Figure 3: (a) Supernet training scheme for proposed method. We use Token Overlap Ratio to select promising networks from the elite pool to distill subnetworks in the refined search space. (b) The overall architecture of the NASTrack. In addition to selecting the embedding dimension, number of heads, MLP ratio and Q-K-V dim for each layer, we also incorporate the keeping ratio as an additional choice.

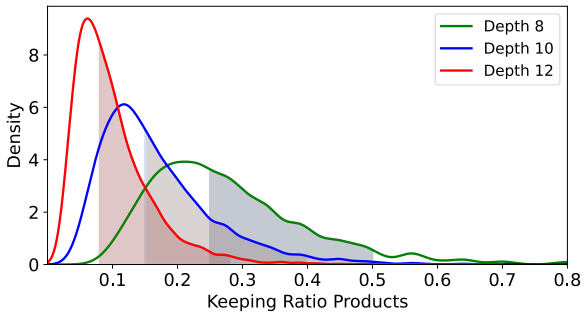

Figure 4: Kernel Density Estimation (KDE) of the keeping ratio products for networks with 8, 10, and 12 layers. The curves represent the distribution of keeping ratio products for networks with different depth. The shaded regions highlight the keeping ratio product ranges for high-performance networks.

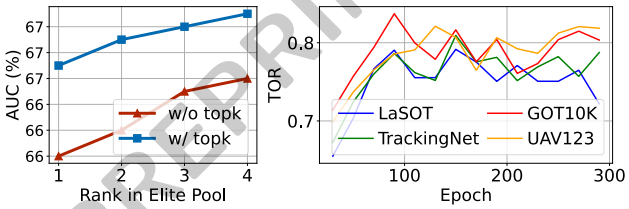

Figure 5: Left: Comparison of network performance between the trackers searched by method using all remaining tokens and elite tokens. Right: Average TOR computed for 2,000 network pairs across four datasets at different epochs. Different colors represent different datasets, showing how the average TOR evolves over training.

in low IoU_{val}, potentially leading to the erroneous removal of valid pruning structures. To decouple the effects of pruning structures and network architectures on performance, we introduce the number of parameters (Params) as an additional evaluation metric. As highlighted in many zero-shot NAS methods [Lee and Ham, 2024; Wei *et al.*, 2024b], Params and FLOPs are reliable and computationally efficient proxies for model performance. Unlike FLOPs, which depend on the number of input tokens, Params remain invariant to token counts, making them a more suitable metric for isolating the impact of pruning structures.

To systematically filter out ineffective pruning structures $\mathcal{P}_\alpha$ while preserving those that contribute to optimal model performance, we define the blacklist $\mathcal{B}$ as follows:

$$\mathcal{B}^{(t+1)} = \mathcal{B}^{(t)} \cup \{\mathcal{P}_\alpha | \text{Params}(\alpha) \geq T_{\text{Params}} \,\&\, \text{IoU}_{\text{val}}\left(\mathcal{N}(\alpha, w_\alpha)\right) < T_{\text{IoU}}\}. \quad (1)$$

The function $\mathcal{N}(\alpha, w_\alpha)$ denotes the subnetwork α inherits the weight w_α from the supernet. This criterion ensures that a pruning structure $\mathcal{P}_\alpha$ is discarded only if the network's Params remain within a reasonable range while yielding a low IoU_{val}. By applying this strategy, we mitigate the risk of discarding promising pruning structures due to biases introduced by network architecture variations.

3.3 Matching-Based Distillation

Although we have narrowed the search space, there still exist networks with significant disparities in model size within the space. This leads to interference in optimization among different networks, resulting in poor subnetworks performance. To address this issue, we assign each subnetwork in the search space a suitable teacher for distillation, thereby accelerating the convergence of the subnetworks [Peng *et al.*, 2020].

Firstly, we identify networks with high capacity and low computational cost during training and place them into an elite pool $\mathcal{E}$ as teacher candidates. This process is updated through the following formula:

$$\hat{\alpha}_k \leftarrow \{\alpha | \text{IoU}_{\text{val}}\left(\mathcal{N}(\alpha, w_\alpha)\right) \geq \text{IoU}_{\text{val}}\left(\mathcal{N}(\hat{\alpha}_k, w_{\hat{\alpha}_k})\right) \,\&\, \text{FLOPs}(\alpha) \leq \text{FLOPs}(\hat{\alpha}_k),\ \hat{\alpha}_k \in \mathcal{E}\}. \quad (2)$$

During each iteration, a newly sampled architecture α is eligible to replace an existing candidate $\hat{\alpha}_k \in \mathcal{E}$ if it demonstrates superior Pareto-dominance, characterized by higher validation performance under stricter or equivalent computational constraints.

Then, we select teacher networks from the elite pool. It is important to note that distillation tends to be less effective

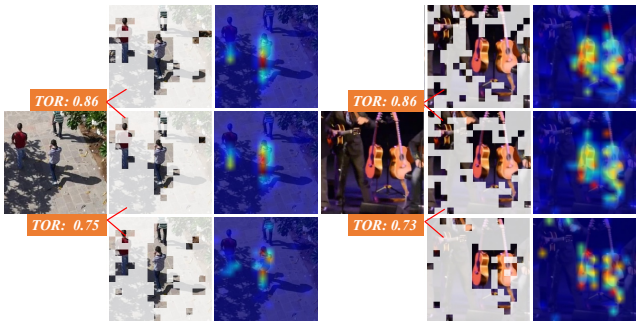

Figure 6: Visualization of the discriminative regions of backbone features and retained tokens extracted by subnetworks from search space.

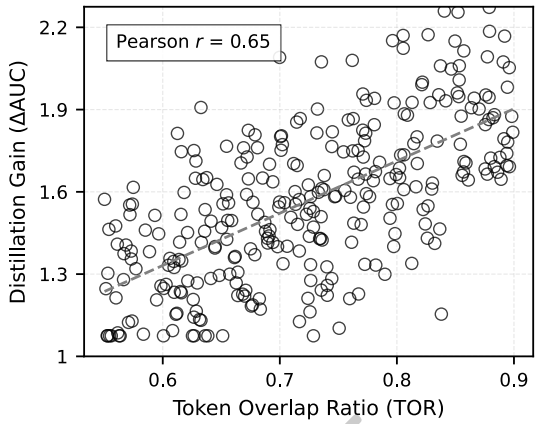

Figure 7: Relationship between TOR and distillation gain on LaSOT. Each point represents a teacher–student pair sampled from the supernet.

when there is a significant capacity gap between teacher and student models [Dong *et al.*, 2023]. Therefore, selecting a suitable teacher for each network becomes a critical step in ensuring effective knowledge transfer.

Cream of the Crop introduces an additional training module to select teacher-student pairs. While effective, this approach increases the overall training burden. On the other hand, the metric proposed by DISWOT [Dong *et al.*, 2023] requires two feature maps of the same size for calculation. However, after token pruning, token sequences often have varying lengths, making it infeasible to directly apply this metric.

To overcome these limitations, we propose the Token Overlap Ratio (TOR), a metric that measures the structural consistency between pruned token sets of different networks.

TOR is defined on the *search tokens* retained after pruning in the last transformer block, as these tokens are most relevant to target localization. Let $\tilde{P}^i$ and $\tilde{P}^j$ denote the index sets of retained search tokens in networks i and j, respectively. TOR is computed as:

$$\text{TOR} = \frac{|\tilde{P}^i \cap \tilde{P}^j|}{|\tilde{P}^i \cup \tilde{P}^j|}. \tag{3}$$

However, we observe that TOR is biased by the number of retained tokens: networks with larger token sets tend to exhibit higher overlap, while subset relationships may underestimate semantic similarity. To mitigate this issue, we compute TOR on a fixed number of *elite tokens*. For each network i, we select the top-k_e tokens from $\tilde{P}^i$ according to their importance scores S^i:

$$P^i = \text{argsort}(S^i, \text{top-}k_e). \tag{4}$$

By enforcing equal token cardinality and prioritizing semantically important tokens, TOR provides a more reliable measure of token-level alignment between subnetworks. As shown in Fig. 5, computing TOR with elite tokens consistently improves the performance of the final searched models.

The effectiveness of TOR stems from its ability to capture semantic alignment between subnetworks at the token level. As illustrated in Fig. 6, different subnetworks attend to different regions of the same search sequence, leading to variations in the retained tokens. Network pairs with high TOR exhibit highly similar activation maps, indicating consistent focus on discriminative target regions.This qualitative observation is further supported by Fig. 7, where a positive correlation between TOR and distillation gain (ΔAUC) can be observed. It shows that higher token-level consistency generally results in more effective knowledge transfer.

In the early stages of training, TOR does not effectively reflect the alignment between networks. Fig. 5 (right) shows that TOR reaches stability in the middle stages of training (around 80 epochs) and changes little afterward. Therefore, we begin joint matching training after 80 epochs.

4 Experiments

4.1 Implementation Details

We use the sandwich rule [Yu *et al.*, 2020] for early-stage supernet training. After that, we apply Equation 2 to select networks into the elite pool. We then randomly sample a network and compute the TOR between it and those in the elite pool. The network with the highest TOR is selected as the teacher to distill it during joint training. We select the best network from the elite pool as the final searched models. Additionally, we provide an evolutionary algorithm to explore a broader set of high-performance networks. The details of the evolutionary algorithm and supernet training pipeline are provided in supplementary materials.

Supernet training. We train the model using the OSTrack [Ye *et al.*, 2022] framework, which adopts the Vanilla ViT-Base [Dosovitskiy *et al.*, 2021] model as the backbone, pre-trained with MAE [He *et al.*, 2022], to jointly perform feature extraction and relation modeling. The head is a lightweight fully convolutional network (FCN) consisting of four stacked Conv-BN-ReLU layers, designed to produce three outputs. To accommodate different computational constraints, we design three supernets, as shown in Tab.1, and optimize them using our proposed method on tracking

Methods	Source	LaSOT			LaSOT$_{ext}$		TrackingNet			TNL2K		UAV123		FLOPS	FPS		
		AUC	P$_{Norm}$	P	AUC	P	AUC	P$_{Norm}$	P	AUC	P	AUC	P	(G)	GPU	Orin NX	CPU
NASTrack-T	Ours	**64.2**	**73.1**	**67.1**	**44.0**	**48.4**	**79.9**	**85.1**	**77.0**	**51.8**	**49.6**	**66.5**	**87.2**	1.3	341	62	75
CompressTracker-2 [Hong *et al.*, 2025]	ICCV2025	60.4	68.5	61.5	40.4	43.8	78.2	83.3	74.8	48.5	45.0	62.5	82.5	6.4	330	51	33
HiT-Small [Kang *et al.*, 2023]	ICCV2023	60.5	68.3	61.5	40.4	-	77.7	81.9	73.1	-	-	63.3	-	1.1	350	48	72
MixformerV2-S [Cui *et al.*, 2024]	NeurIPS2024	60.6	69.9	60.4	43.6	46.2	75.8	81.1	70.4	48.3	43.0	65.8	86.8	4.5	331	61	46
LightTrack [Yan *et al.*, 2021b]	CVPR2021	53.8	-	53.7	-	-	72.5	77.8	69.5	-	-	-	-	0.5	157	36	52
FEAR-XS [Borsuk *et al.*, 2022]	ECCV2022	53.5	-	54.5	-	-	71.5	80.5	69.9	-	-	-	-	0.5	224	50	111
NASTrack-S	Ours	**65.7**	**74.9**	**69.9**	**44.8**	**49.8**	81.3	86.1	78.8	**53.4**	**51.8**	**67.2**	**87.2**	4.1	271	48	32
CompressTracker-3 [Hong *et al.*, 2025]	ICCV2025	64.9	74.0	68.4	44.6	49.6	**81.6**	**86.7**	**79.4**	52.6	50.9	65.4	-	8.6	253	45	26
DyTrack-Fast [Zhu *et al.*, 2025]	TNNLS	64.9	73.4	67.8	44.0	47.6	79.4	83.7	75.8	50.7	-	66.7	-	9.4	-	-	-
HiT-Base [Kang *et al.*, 2023]	ICCV2023	64.6	73.3	68.1	44.1	-	80.0	84.4	77.3	-	-	65.6	-	4.3	220	40	33
LiteTrack-B4 [Wei *et al.*, 2024a]	ICRA2024	62.5	72.1	65.7	-	-	79.9	84.9	76.6	-	-	66.4	-	6.8	242	44	24
HCAT [Chen *et al.*, 2022]	ECCV2022	59.0	-	60.5	-	-	76.6	82.6	72.9	-	-	63.6	-	5.9	235	34	36
NASTrack-B	Ours	**68.6**	**77.9**	**74.3**	**47.2**	**52.9**	**83.7**	**88.4**	**82.7**	**56.4**	**56.9**	**69.2**	**89.8**	14.8	125	21	11
CompressTracker-6 [Hong *et al.*, 2025]	ICCV2025	67.5	77.5	72.4	46.7	52.5	82.9	87.8	81.5	54.7	54.3	67.9	88.7	15.4	130	23	11
Litetrack-B9 [Wei *et al.*, 2024a]	ICRA2024	67.0	77.0	72.7	-	-	82.4	87.3	80.4	55.4	-	67.7	-	14.2	128	21	11
DyTrack-Medi [Zhu *et al.*, 2025]	TNNLS	66.5	75.5	70.4	46.3	51.1	80.9	85.5	77.8	53.8	-	68.2	-	16.3	-	-	-
STARK-ST50 [Yan *et al.*, 2021a]	ICCV2021	66.6	-	-	-	-	81.3	86.1	-	-	-	68.4	-	12.8	120	18	24
TransT [Chen *et al.*, 2021b]	CVPR2021	64.9	73.8	69.0	-	-	81.4	86.7	80.3	50.7	-	69.1	-	18.2	76	13	6
OSTrack [Ye *et al.*, 2022]	ECCV2022	69.1	78.7	75.2	47.4	53.3	83.1	87.8	82.0	54.3	-	68.3	-	21.5	98	18	7

Table 2: State-of-the-art comparison on TrackingNet, LaSOT, UAV123, TNL2K test benchmark. The best results are highlighted in red.

ID	Components					Metrics	
	Blacklist	Rand. Drop	MBD	Rand. Match		AUC	FLOPs
I						68.0	14.8
II		✓				68.1	14.0
III	✓					68.5	14.8
IV			✓			68.3	14.2
V	✓			✓		68.7	14.7
VI	✓		✓			**69.2**	14.8

Table 3: Component-wise analysis. MBD: Matching-Based Distillation; Rand. Drop: Randomly discarding keeping ratios; Rand. Match: Randomly sampled teacher.

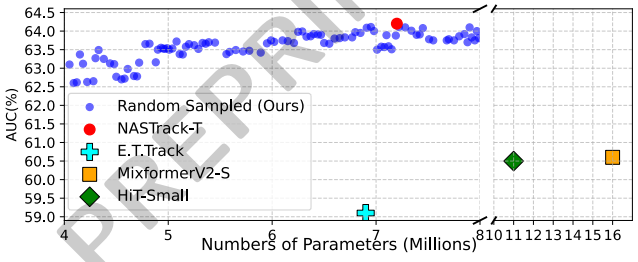

Figure 8: Performance on LaSOT of NASTrack-T and 100 sampled high-performing architectures from the supernet$_{tiny}$ with weight inherited from the supernet.

datasets for 300 epochs. The tracking datasets include TrackingNet [Müller *et al.*, 2018], GOT-10k [Huang *et al.*, 2019], LaSOT [Fan *et al.*, 2019], and COCO2017 [Lin *et al.*, 2014], with data preprocessing following OSTrack. During tracker training, we use the AdamW optimizer, setting the weight decay to 10^{-4}, the initial learning rate to 1.5×10^{-4} for the backbone and 1.5×10^{-3} for other parameters. The search and template images are resized to resolutions of 256×256 and 128×128. The tracker is trained on six NVIDIA RTX 4090 GPUs, each holding 80 image pairs, resulting in a total batch size of 480. Each epoch processes 60k image pairs, and the learning rate is reduced by a factor of 10 after 240 epochs.

4.2 Results and Comparisons

To validate the effectiveness of our algorithm, we compare it with a large number of trackers on four datasets, including LaSOT [Fan *et al.*, 2019], TrackingNet [Müller *et al.*, 2018], TNL2K [Wang *et al.*, 2021], UAV123 [Mueller *et al.*, 2016], These trackers are categorized into two groups: lightweight trackers and deep trackers. All performance evaluations are conducted on a PC equipped with an Intel(R) XEON(R) GOLD and six Nvidia RTX 4090 GPUs. For speed comparison, we utilize three different platforms: a Nvidia GeForce RTX 4070 laptop, a Intel i7-14700HX CPU and a Nvidia Jetson Orin NX 16GB. The results are shown in Tab.2.

Compared with SOTA lightweight trackers, our method outperforms existing trackers on all benchmarks. In terms of speed, our tracker achieves a significant improvement over the baseline OSTrack without sacrificing accuracy. In particular, our NASTrack-B achieves 69.2% AUC on UAV123, which is 0.9% higher than our baseline OSTrack while running 1.3× faster. Furthermore, as shown in Fig. 8, our search space contains many well-trained trackers that consistently outperform existing lightweight tracking models.

4.3 Ablation Study and Analysis

Effect of Blacklist Strategy and Matching-Based Distillation. We conduct an ablation study to evaluate the effectiveness of each component in our method. We follow the search pipeline of AutoFormer as the baseline and perform the search on OSTrack. We employ an evolutionary algorithm to search for high-performing networks in the absence of the elite pool.

Tab.3 shows that applying either strategy individually enhances performance, while the combination of both achieves the most improvement. To further validate our method, we compare the performance of each component under four conditions: (II) Randomly discarding keeping ratio configurations. (III) Discarding keeping ratio configurations based on IoU and model size. (V) Randomly select networks from the elite pool. (VI) Selecting a teacher network based on TOR. The results show that (III) outperforms (II) and (VI) surpasses

Method	TrackingNet (%)	LaSOT (%)	UAV (%)	FLOPs (G)	Param (M)
Autoformer†	80.1	64.2	65.9	13.1	43.1
Autoformer	81.0	64.0	66.6	12.8	67.0
Sandwich Rule	82.1	66.5	68.3	12.8	61.1
Cream of the Crop	82.3	66.6	68.2	12.5	64.3
Ours	**82.8**	**67.8**	**68.7**	**12.9**	**68.1**

Table 4: Performance comparison (AUC) of different NAS methods under the same computational constraints. †: Search on vanilla search space without pruning.

Method	GFLOPs				
	1.3	4.1	8.2	10.4	14.8
Token Pruning (Fixed Arch)	-	-	78.1	81.0	82.8
Standard NAS (w/o Token Pruning)	77.4	80.1	81.2	81.8	83.2
Decoupled (NAS → Grid Search)	79.1	81.1	82.1	82.2	83.5
Ours (Joint NAS + Ratio Search)	**79.9**	**81.3**	**81.9**	**82.5**	**83.7**

Table 5: Ablation study results on TrackingNet. We show the AUC of best searched models for each case. Fixed Arch: OSTrack_{256}.

Method	Ratio Strategy	AUC (%)	FLOPs (G)	PT*	Cost (h)
NASTrack-B†	Grid Search	68.2	1.3	Yes	120+
	Learnable Threshold	68.7	1.4	Yes	45
NASTrack	**Joint Search**	**68.6**	**1.3**	**No**	**20**

*PT: Requires additional post-training/fine-tuning after search.

Table 6: Comparison of different ratio selection methods on LaSOT. All methods are constrained to ∼1.3 GFLOPs. Cost is measured in GPU hours on Nvidia RTX 4090. NASTrack-B†: NASTrack-B without token pruning.

Model	Method	AUC (%)	Params (M)	FLOPs (G)	Speed (fps)
NASTrack-B†	**Ours**	**68.6**	**74.1**	**14.3**	**125**
	EViT [Liang *et al.*, 2022]	68.7	74.1	15.4	100
	PPT [Wu *et al.*, 2023]	68.4	74.1	15.4	105

Table 7: Comparison of our searched keeping ratios versus manual pruning schemes on LaSOT.

(V), demonstrating the effectiveness of our proposed methods in both pruning strategy optimization and teacher network selection.

The Efficacy of Multi-Dimensional Search. We compare the results with and without searching for the optimal keeping ratio. As shown in Tab. 4, the network discovered through our search space achieves better performance under the same computational budget. These results demonstrate that jointly pruning in both the token and structural dimensions effectively compresses the tracker while preserving competitive performance.

To further validate the effectiveness of our method in search spaces that incorporate the keeping ratio, we compare it against other NAS approaches, our approach identifies better-performing networks.

Superiority over Ratio Selection Strategies. We compare our joint search with established ratio selection methods under a 1.3 GFLOPs constraint in Tab. 6. Traditional grid-based search on a fixed backbone suffers from exponential search costs and requires extensive retraining. Learnable threshold methods, while reducing search overhead, remain sensitive to hyperparameters and necessitate additional post-training to stabilize performance. In contrast, by treating layer-wise keeping ratios as searchable architectural variables, our joint strategy discovers optimal architecture-pruning pairs simultaneously. This one-shot approach directly satisfies the FLOPs constraint without post-training, demonstrating superior efficiency and reliability over decoupled or implicit selection methods.

Synergy Between Architecture Search and Token Pruning. We investigate the necessity of co-designing architecture and token pruning by comparing our joint search against a decoupled pipeline in Tab. 5. While the decoupled approach improves over single-factor optimization, it consistently lags behind our joint search. This performance gap reveals a fundamental coupling: the optimal pruning ratio is highly dependent on the specific architecture, and vice versa. Such non-trivial interactions—where token reduction directly alters the effective receptive field and information flow—cannot be captured when architecture and pruning are optimized in isolation. These results demonstrate that joint search is not a simple extension of prior NAS frameworks, but a crucial design choice to fully exploit architecture-pruning synergy for superior trade-offs.

Comparison with Token Pruning Methods. We believe that an optimized keeping rate structure can achieve performance on par with carefully designed pruning strategies. As shown in Tab.7, our searched pruning configurations achieve performance comparable to manually designed pruning strategies under the same network architectures.

5 Conclusion

In this work, we propose NASTrack, the first method to leverage NAS for designing efficient transformer-based trackers. This framework incorporates token pruning to optimize both transformer block structures and layer-wise keeping ratio. The multi-dimensional compression enables speedup without over-compressing the model. We also find that optimizing per-layer token keeping ratios alone can achieve performance comparable to existing token pruning methods. Meanwhile, We design two training strategies to manage the enlarged search space caused by the keeping ratio. First, we blacklist suboptimal pruning structures to shrink search space. Second, we introduce a novel metric, TOR, to select the optimal teacher-student model pairs, accelerating network convergence. Extensive experiments demonstrate the proposed algorithm can enhance supernet training. The subnetworks searched through this framework achieve state-of-the-art performance on multiple benchmarks, outperforming existing efficient trackers.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62273048).

References

[Bertinetto *et al.*, 2016] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, pages 850–865, 2016.

[Bolya *et al.*, 2023] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023.

[Borsuk *et al.*, 2022] Vasyl Borsuk, Roman Vei, Orest Kupyn, Tetiana Martyniuk, Igor Krashenyi, and Jiři Matas. Fear: Fast, efficient, accurate and robust visual tracker. In *ECCV*, pages 644–663, 2022.

[Chen *et al.*, 2021a] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, pages 12250–12260, 2021.

[Chen *et al.*, 2021b] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, pages 8122–8131, 2021.

[Chen *et al.*, 2022] Xin Chen, Ben Kang, Dong Wang, Dongdong Li, and Huchuan Lu. Efficient visual tracking vianbsp;hierarchical cross-attention transformer. In *ECCVW*, page 461–477, 2022.

[Chen *et al.*, 2023] Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *CVPR*, pages 14572–14581, 2023.

[Cui *et al.*, 2022] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *CVPR*, pages 13598–13608, 2022.

[Cui *et al.*, 2024] Yutao Cui, Tianhui Song, Gangshan Wu, and Limin Wang. Mixformerv2: Efficient fully transformer tracking. In *Advances in Neural Information Processing Systems*, pages 58736–58751, 2024.

[Dong *et al.*, 2023] Peijie Dong, Lujun Li, and Zimian Wei. Diswot: Student architecture search for distillation without training. In *CVPR*, pages 11898–11908, 2023.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[Fan *et al.*, 2019] 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*, pages 5369–5378, 2019.

[Gong *et al.*, 2022] Chengyue Gong, Dilin Wang, Meng Li, Xinlei Chen, Zhicheng Yan, Yuandong Tian, Vikas Chandra, et al. Nasvit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *ICLR*, 2022.

[Guo *et al.*, 2020] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pages 544–560, 2020.

[He *et al.*, 2022] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022.

[Hong *et al.*, 2025] Lingyi Hong, Jinglun Li, Xinyu Zhou, Shilin Yan, Pinxue Guo, Kaixun Jiang, Zhaoyu Chen, Shuyong Gao, Wei Zhang, Hong Lu, et al. General compression framework for efficient transformer object tracking, 2025.

[Hou and Kung, 2022] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional vision transformer compression via dependency guided gaussian process search. In *CVPRW*, pages 3668–3677, 2022.

[Howard *et al.*, 2019] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, pages 1314–1324, 2019.

[Huang *et al.*, 2019] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 43:1562–1577, 2019.

[Kang *et al.*, 2023] Ben Kang, Xin Chen, Dong Wang, Houwen Peng, and Huchuan Lu. Exploring lightweight hierarchical vision transformers for efficient visual tracking. In *ICCV*, pages 9612–9621, 2023.

[Kong *et al.*, 2022] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *ECCV*, pages 620–640, 2022.

[Lee and Ham, 2024] Junghyup Lee and Bumsub Ham. Aznas: Assembling zero-cost proxies for network architecture search. In *CVPR*, pages 5893–5903, 2024.

[Li *et al.*, 2018] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018.

[Li *et al.*, 2023] Shuiwang Li, Yangxiang Yang, Dan Zeng, and Xucheng Wang. Adaptive and background-aware vision transformer for real-time uav tracking. In *ICCV*, pages 13943–13954, 2023.

[Liang *et al.*, 2022] 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. In *ICLR*, 2022.

[Lin *et al.*, 2014] 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*, pages 740–755, 2014.

[Liu *et al.*, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[Liu *et al.*, 2022] Jing Liu, Jianfei Cai, and Bohan Zhuang. Focusformer: Focusing on what we need via architecture sampler, 2022.

[Marin *et al.*, 2023] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers for image classification. In *WACV*, pages 12–21, 2023.

[Mueller *et al.*, 2016] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, pages 445–461, 2016.

[Müller *et al.*, 2018] Matthias Müller, Adel Bibi, Salman Giancola, Silvioand Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, pages 310–327, 2018.

[Peng *et al.*, 2020] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. In *Advances in Neural Information Processing Systems*, 2020.

[Pham *et al.*, 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, pages 4095–4104, 2018.

[Real *et al.*, 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, pages 4780–4789, 2019.

[Tang *et al.*, 2023] Chen Tang, Li Lyna Zhang, Huiqiang Jiang, Jiahang Xu, Ting Cao, Quanlu Zhang, Yuqing Yang, Zhi Wang, and Mao Yang. Elasticvit: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices. In *ICCV*, pages 5806–5817, 2023.

[Wang *et al.*, 2021] Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *CVPR*, pages 13763–13773, 2021.

[Wei *et al.*, 2024a] 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*, pages 4968–4975, 2024.

[Wei *et al.*, 2024b] Zimian Wei, Peijie Dong, Zheng Hui, Anggeng Li, Lujun Li, Menglong Lu, Hengyue Pan, and Dongsheng Li. Auto-prox: Training-free vision transformer architecture search via automatic proxy discovery. In *AAAI*, pages 15814–15822, 2024.

[Wu *et al.*, 2023] Xinjian Wu, Fanhu Zeng, Xiudong Wang, and Xinghao Chen. Ppt: Token pruning and pooling for efficient vision transformers, 2023.

[Xie and Yuille, 2017] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, pages 1379–1388, 2017.

[Xie *et al.*, 2024] 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*, pages 19300–19309, 2024.

[Yan *et al.*, 2021a] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, pages 10428–10437, 2021.

[Yan *et al.*, 2021b] 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*, pages 15175–15184, 2021.

[Ye *et al.*, 2022] 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*, pages 341–357, 2022.

[Yin *et al.*, 2022] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. Adavit: Adaptive tokens for efficient vision transformer. In *CVPR*, pages 10799–10808, 2022.

[Yu *et al.*, 2020] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, pages 702–717, 2020.

[Zhang *et al.*, 2023] Mingyang Zhang, Xinyi Yu, Haodong Zhao, and Linlin Ou. Shiftnas: Improving one-shot nas via probability shift. In *ICCV*, pages 5919–5928, 2023.

[Zhu *et al.*, 2025] Jiawen Zhu, Xin Chen, Haiwen Diao, Shuai Li, Jun-Yan He, Chenyang Li, Bin Luo, Dong Wang, and Huchuan Lu. Exploring Dynamic Transformer for Efficient Object Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2025.