Joint Feature Learning and Relation Modeling for Tracking: A One-Stream Framework

Botao Ye[1,2], Hong Chang[1,2], Bingpeng Ma[2],
Shiguang Shan[1,2], and Xilin Chen[1,2]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
botao.ye@vipl.ict.ac.cn, changhong@ict.ac.cn, bpma@ucas.ac.cn, {sgshan, xlchen}@ict.ac.cn

Abstract. The current popular two-stream, two-stage tracking framework extracts the template and the search region features separately and then performs relation modeling, thus the extracted features lack the awareness of the target and have limited target-background discriminability. To tackle the above issue, we propose a novel *one-stream tracking* (OSTrack) framework that unifies feature learning and relation modeling by bridging the template-search image pairs with bidirectional information flows. In this way, discriminative target-oriented features can be dynamically extracted by mutual guidance. Since no extra heavy relation modeling module is needed and the implementation is highly parallelized, the proposed tracker runs at a fast speed. To further improve the inference efficiency, an in-network candidate early elimination module is proposed based on the strong similarity prior calculated in the one-stream framework. As a unified framework, OSTrack achieves state-of-the-art performance on multiple benchmarks, in particular, it shows impressive results on the one-shot tracking benchmark GOT-10k, *i.e.*, achieving 73.7% AO, improving the existing best result (SwinTrack) by 4.3%. Besides, our method maintains a good performance-speed trade-off and shows faster convergence. The code and models are available at https://github.com/botaoye/OSTrack.

1 Introduction

Visual object tracking (VOT) aims at localizing an arbitrary target in each video frame, given only its initial appearance. The *continuously changing* and *arbitrary* nature of the target poses a challenge to learn a target appearance model that can effectively discriminate the specified target from the background. Current mainstream trackers typically address this problem with a common *two-stream* and *two-stage* pipeline, which means that the features of the template and the search region are separately extracted (two-stream), and the whole process is divided into two sequential steps: feature extraction and relation modeling (two-stage). Such a natural pipeline employs the strategy of "divide-and-conquer" and achieves remarkable success in terms of tracking performance.

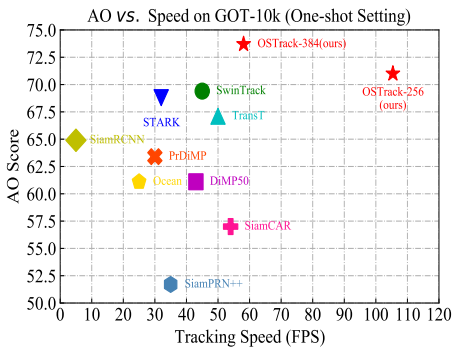

Fig. 1: A comparison of AO and speed of state-of-the-art trackers on GOT-10k under one-shot setting. Our OSTrack-384 sets a new SOTA of 73.7% AO on GOT-10k, showing impressive one-shot tracking performance. OSTrack-256 runs at 105.4 FPS while still outperforming all previous trackers.

However, the separation of feature extraction and relation modeling suffers from the following limitations. Firstly, the feature extracted by the vanilla two-stream two-stage framework is unaware of the target. In other words, the extracted feature for each image is determined after off-line training, since there is no interaction between the template and the search region. This is against with the continuously changing and arbitrary nature of the target, leading to limited target-background discriminative power. On some occasions when the category of the target object is not involved in the training dataset (*i.e.*, one-shot tracking), the above problems are particularly serious. Secondly, the two-stream, two-stage framework is vulnerable to the performance-speed dilemma. According to the computation burden of the feature fusion module, two different strategies are commonly utilized. The first type, shown in Fig 2(a), simply adopts one single operator like cross-correlation [1,23] or discriminative correlation filter [2,6], which is efficient but less effective since the simple linear operation leads to discriminative information loss [4]. The second type, shown in Fig 2(b), addresses the information loss by complicated non-linear interaction (Transformer [38]), but is less efficient due to a large number of parameters and the use of iterative refinement (*e.g.*, for each search image, STARK-S50 [43] takes 7.5 ms for the feature extraction and 14.1 ms for relation modeling on an RTX2080Ti GPU).

In this work, we set out to address the aforementioned problems via a unified *one-stream one-stage* tracking framework. The core insight of the one-stream framework is to bridge a free information flow between the template and search region at the early stage (*i.e.*, the raw image pair), thus extracting target-oriented features and avoiding the loss of discriminative information. Specifically, we concatenate the flattened template and search region and feed them into staked self-attention layers [38] (widely used Vision Transformer (ViT) [10] is chosen in our implementation), and the produced search region features can be directly used for target classification and regression without further matching. The staked self-attention operations enable iteratively feature matching between the template and the search region, thus allowing mutual guidance for target-oriented feature extraction. Therefore, both template and search region features can be extracted dynamically with strong discriminative power. Additionally, the proposed framework achieves a good balance between performance and speed

because the concatenation of the template and the search region makes the one-stream framework highly parallelizable and does not require additional heavy relational modeling networks.

Moreover, the proposed one-stream framework provides a strong prior about the similarity of the target and each part of the search region (*i.e.* candidates) as shown in Fig. 4, which means that the model can identify background regions even at the early stage. This phenomenon verifies the effectiveness of the one-stream framework and motivates us to propose an in-network *early candidate elimination* module for progressively identifying and discarding the candidates belonging to the background in a timely manner. The proposed candidate elimination module not only significantly boosts the inference speed, but also avoids the negative impact of uninformative background regions on feature matching.

Despite its simple structure, the proposed trackers achieve impressive performance and set a new state-of-the-art (SOTA) on multiple benchmarks. Moreover, it maintains adorable inference efficiency and shows faster convergence compared to SOTA Transformer based trackers. As shown in Fig. 1, our method achieves a good balance between the accuracy and inference speed.

The main contributions of this work are three-fold: (1) We propose a simple, neat, and effective one-stream, one-stage tracking framework by combining the feature extraction and relation modeling. (2) Motivated by the prior of the early acquired similarity score between the target and each part of the search region, an in-network early candidate elimination module is proposed for decreasing the inference time. (3) We perform comprehensive experiments to verify that the one-stream framework outperforms the previous SOTA two-stream trackers in terms of performance, inference speed, and convergence speed. The resulting tracker OSTrack sets a new state-of-the-art on multiple tracking benchmarks.

2 Related Work

In this section, we briefly review different tracking pipelines, as well as the adaptive inference methods related to our early candidate elimination module.

Tracking Pipelines. Based on the different computational burdens of feature extraction and relation modeling networks, we compare our method with two different two-stream two-stage archetypes in Fig. 2. Earlier Siamese trackers [1, 23, 46] and discriminative trackers [2, 6] belong to Fig. 2(a). They first extract the features of the template and the search region separately by a CNN backbone [15, 20], which shares the same structure and parameters. Then, a lightweight relation modeling network (*e.g.*, the cross-correlation layer [1, 22] in Siamese trackers and correlation filter [3, 16] in discriminative trackers) takes responsibility to fuse these features for the subsequent state estimation task. However, the template feature cannot be adjusted according to the search region feature in these methods. Such a shallow and unidirectional relation modeling strategy may be insufficient for information interaction. Recently, stacked Transformer layers [38] are introduced for better relation modeling. These methods belong to Fig. 2(b) where the relation modeling module is relatively heavy

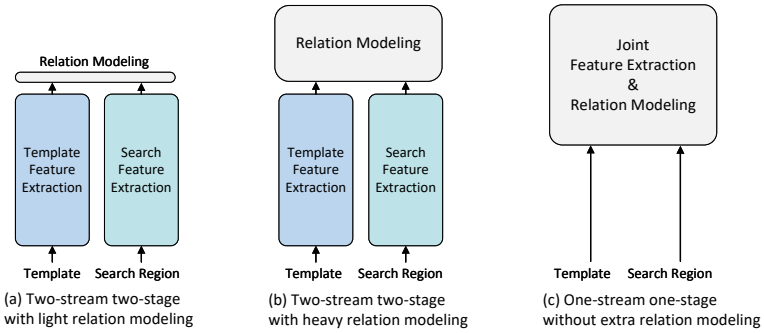

Fig. 2: Three different taxonomies of tracking pipeline. The height of each rectangular represents the relative model size.

and enables bi-directional information interaction. TransT [4] proposes to stack a series of self-attention and cross-attention layers for iterative feature fusion. STARK [43] concatenates the pre-extracted template and search region features and feeds them into multiple self-attention layers. The bi-directional heavy structure brings performance gain but inevitably slows down the inference speed. Differently, our one-stream one-stage design belongs to Fig. 2(c). For the first time, we seamlessly combine feature extraction and relation modeling into a unified pipeline. The proposed method provides free information flow between the template and search region with minor computation costs. It not only generates target-oriented features by mutual guidance but also is efficient in terms of both training and testing time.

Adaptive Inference. Our early candidate elimination module can be seen as a progressive process of adaptively discarding potential background regions based on the similarity between the target and the search region. One related topic is the adaptive inference [24,33,44] in vision transformers, which is proposed to accelerate the computation of ViT. DynamicViT [33] trains extra control gates with the Gumbel-softmax trick to discard tokens during inference. Instead of directly discarding non-informative tokens, EViT [24] fuses them to avoid potential information loss. These works are tightly coupled with the classification task and are therefore not suitable for tracking. Instead, we treat each token as a target candidate and then discard the candidates that are least similar to the target by means of a free similarity score calculated by the self-attention operation. To the best of our knowledge, this is the first work that attempts to eliminate potential background candidates within the tracking network.

3 Method

This section describes the proposed one-stream tracker (OSTrack). The input image pairs are fed into a ViT backbone for simultaneous feature extraction and relation modeling, and the resulting search region features are directly adopted

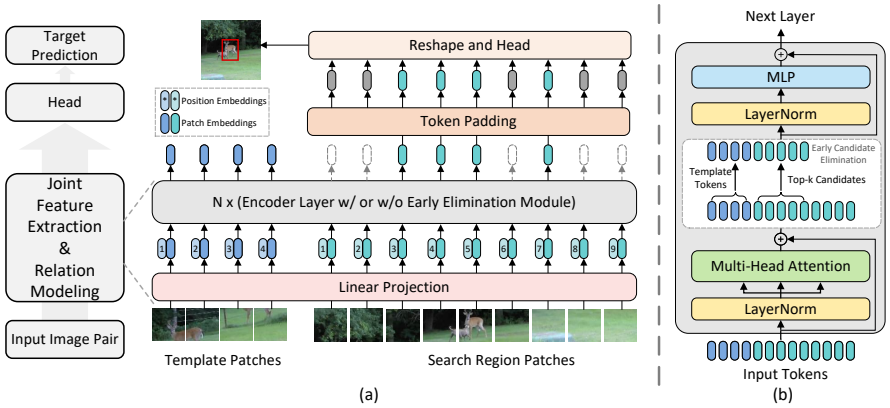

Fig. 3: **(a)** The overall framework of the proposed one-stream framework. The template and search region are split, flattened, and linear projected. Image embeddings are then concatenated and fed into Transformer encoder layers for joint feature extraction and relation modeling. **(b)** The structure of the encoder layer with early candidate elimination module, which is insert after the multi-head attention operation [38].

for subsequent target classification and regression. An overview of the model is shown in Fig. 3(a).

3.1 Joint Feature Extraction and Relation Modeling

We propose to combine the feature extraction and relation modeling modules and construct a free information flow between the contents of the template and the search region. The global contextual modeling capacity of self-attention [38] operation perfectly fits our goal, therefore, vanilla ViT [10] is selected as the main body of OSTrack. Adopting the existing Vision Transformer architecture also provides a bunch of publicly available pre-trained models [14, 36], freeing us from the time-consuming pre-training stage. The input of OSTrack is a pair of images, namely, the template image patch $z \in \mathbb{R}^{3\times H_z\times W_z}$ and the search region patch $x \in \mathbb{R}^{3\times H_x\times W_x}$. They are first split and flattened into sequences of patches $z_p \in \mathbb{R}^{N_z\times\left(3\cdot P^2\right)}$ and $x_p \in \mathbb{R}^{N_x\times\left(3\cdot P^2\right)}$, where $P\times P$ is the resolution of each patch, and $N_z = H_zW_z/P^2$, $N_x = H_xW_x/P^2$ are the number of patches of template and search region respectively. After that, a trainable linear projection layer with parameter $\boldsymbol{E}$ is used to project z_p and x_p into D dimension latent space as in Eq. 1 and Eq. 2, and the output of this projection is commonly called patch embeddings [10]. Learnable 1D position embeddings $\boldsymbol{P}_z$ and $\boldsymbol{P}_x$ are added to the patch embeddings of the template and search region separately to produce the final template token embeddings $\boldsymbol{H}_z^0 \in \mathbb{R}^{N_z\times D}$ and search region

token embeddings $\boldsymbol{H}_x^0 \in \mathbb{R}^{N_x \times D}$.

$$\boldsymbol{H}_z^0 = \left[\boldsymbol{z}_p^1\boldsymbol{E}; \boldsymbol{z}_p^2\boldsymbol{E}; \cdots; \boldsymbol{z}_p^{N_z}\boldsymbol{E}\right] + \boldsymbol{P}_z, \qquad \boldsymbol{E} \in \mathbb{R}^{\left(3\cdot P^2\right)\times D}, \boldsymbol{P}_z \in \mathbb{R}^{N_z \times D} \tag{1}$$

$$\boldsymbol{H}_x^0 = \left[\boldsymbol{x}_p^1\boldsymbol{E}; \boldsymbol{x}_p^2\boldsymbol{E}; \cdots; \boldsymbol{x}_p^{N_z}\boldsymbol{E}\right] + \boldsymbol{P}_x, \qquad \boldsymbol{P}_x \in \mathbb{R}^{N_x \times D} \tag{2}$$

To verify whether adding addition identity embeddings (to indicate a token belonging to the template or search region as in BERT [9]) or adopting relative positional embeddings are beneficial to the performance, we also conduct ablation studies and observe no significant improvement, thus they are omitted for simplicity (details can be found in the supplementary material).

Token sequences $\boldsymbol{H}_z^0$ and $\boldsymbol{H}_x^0$ are then concatenated as $\boldsymbol{H}_{zx}^0 = [\boldsymbol{H}_z^0; \boldsymbol{H}_x^0]$, and the resulting vector $\boldsymbol{H}_{zx}^0$ is then fed into several Transformer encoder layers [10]. Unlike the vanilla ViT [10], we insert the proposed early candidate eliminating module into some of encoder layers as shown in Fig. 3(b) for inference efficiency, and the technical details are presented in Sec. 3.2. Notably, adopting the self-attention of concatenated features makes the whole framework highly parallelized compared to the cross-attention [4]. Although template images are also fed into the ViT for each search frame, the impact on the inference speed is minor due to the highly parallel structure and the fact that the number of template tokens is small compared to the number of search region tokens.

Analysis. From the perspective of the self-attention mechanism [38], we further analyze the intrinsic reasons why the proposed framework is able to realize simultaneous feature extraction and relation modeling. The output of self-attention operation $\boldsymbol{A}$ in our approach can be written as:

$$\boldsymbol{A} = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\cdot\boldsymbol{V} = \text{Softmax}\left(\frac{[\boldsymbol{Q}_z;\boldsymbol{Q}_x][\boldsymbol{K}_z;\boldsymbol{K}_x]^\top}{\sqrt{d_k}}\right)\cdot[\boldsymbol{V}_z;\boldsymbol{V}_x], \tag{3}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are query, key and value matrices separately. The subscripts z and x denote matrix items belonging to the template and search region. The calculation of attention weights in Eq. 3 can be expanded to:

$$\begin{aligned}\text{Softmax}\left(\frac{[\boldsymbol{Q}_z;\boldsymbol{Q}_x][\boldsymbol{K}_z;\boldsymbol{K}_x]^\top}{\sqrt{d_k}}\right) &= \text{Softmax}\left(\frac{[\boldsymbol{Q}_z\boldsymbol{K}_z^\top, \boldsymbol{Q}_z\boldsymbol{K}_x^\top; \boldsymbol{Q}_x\boldsymbol{K}_z^\top, \boldsymbol{Q}_x\boldsymbol{K}_x^\top]}{\sqrt{d_k}}\right) \\ &\triangleq [\boldsymbol{W}_{zz}, \boldsymbol{W}_{zx}; \boldsymbol{W}_{xz}, \boldsymbol{W}_{xx}],\end{aligned} \tag{4}$$

where $\boldsymbol{W}_{\boldsymbol{zx}}$ is a measure of similarity between the template and the search region, and the rest are similar. The output $\boldsymbol{A}$ can be further written as:

$$\boldsymbol{A} = [\boldsymbol{W}_{zz}\boldsymbol{V}_z + \boldsymbol{W}_{zx}\boldsymbol{V}_x; \boldsymbol{W}_{xz}\boldsymbol{V}_z + \boldsymbol{W}_{xx}\boldsymbol{V}_x]. \tag{5}$$

In the right part of Eq. 5, $\boldsymbol{W}_{xz}\boldsymbol{V}_z$ is responsible for aggregating the iter-image feature (relation modeling) and $\boldsymbol{W}_{xx}\boldsymbol{V}_x$ aggregating the intra-image feature (feature extraction) based on the similarity of different image parts. Therefore, the feature extraction and relation modeling can be done with a self-attention

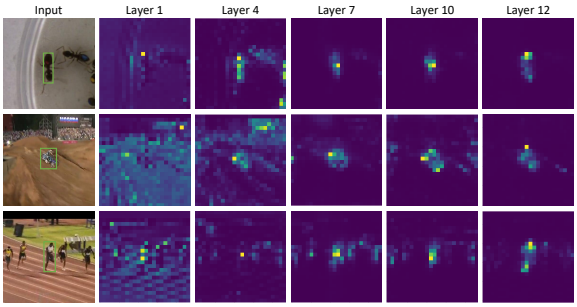

Fig. 4: Visualization of the attention weights of search region corresponding to the center part of template after different ViT layers, the green rectangles indicate target objects. It can be seen as an estimate of the similarity between the target and each position of the search region.

operation. Moreover, Eq. 5 also constructs a bi-direction information flow that allows mutual guidance of target-oriented feature extraction through the similarity learning.

Comparisons with Two-Stream Transformer Fusion Trackers. 1) Previous two-stream Transformer fusion trackers [4, 25] all adopt a Siamese framework, where the features of the template and search region are separately extracted first, and the Transformer layer is only adopted to fuse the extracted features. Therefore, the extracted features of these methods are not adaptive and may lose some discriminative information, which is irreparable. In contrast, OSTrack directly concatenates linearly projected template and search region images at the first stage, so feature extraction and relation modeling are seamlessly integrated and target-oriented features can be extracted through the mutual guidance of the template and the search region. 2) Previous Transformer fusion trackers only employ ImageNet [8] pre-trained backbone networks [15, 27] and leave Transformer layers randomly initialized, which degrades the convergence speed, while OSTrack benefits from pre-trained ViT models for faster convergence. 3) The one-stream framework provides the possibility of identifying and discarding useless background regions for further improving the model performance and inference speed as presented in Sec. 3.2.

3.2 Early Candidate Elimination

Each token of the search region can be regarded as a target candidate and each template token can be considered as a part of the target object. Previous trackers keep all candidates during feature extraction and relation modeling, while background regions are not identified until the final output of the network (*i.e.*, classification score map). However, our one-stream framework provides a strong prior on the similarity between the target and each candidate. As shown in Fig. 4, the attention weights of the search region highlight the foreground objects in the early stage of ViT (*e.g.*, layer 4), and then progressively focus on

the target. This property makes it possible to progressively identify and eliminate candidates belonging to the background regions inside the network. Therefore, we propose an early candidate elimination module that progressively eliminates candidates belonging to the background in the early stages of ViT to lighten the computational burden and avoid the negative impact of noisy background regions on feature learning.

Candidate Elimination. Recall that the self-attention operation in ViT can be seen as a spatial aggregation of tokens with normalized importances [38], which is measured by the dot product similarity between each token pair. Specifically, each template token $\boldsymbol{h}_z^i, 1 \leq i \leq N_z$ is calculated as:

$$\boldsymbol{h}_z^i = \text{Softmax}\left(\frac{\boldsymbol{q}_i \cdot [\boldsymbol{K}_z; \boldsymbol{K}_x]^\top}{\sqrt{d}}\right) \cdot \boldsymbol{V} = [\boldsymbol{w}_z^i; \boldsymbol{w}_x^i] \cdot \boldsymbol{V}, \tag{6}$$

where $\boldsymbol{q}_i$, $\boldsymbol{K}_z$, $\boldsymbol{K}_x$ and $\boldsymbol{V}$ denote the query vector of token $\boldsymbol{h}_z^i$, the key matrix corresponding to the template, the key matrix corresponding to the search region and the value matrix. The attention weight $\boldsymbol{w}_x^i$ determines the similarity between the template part $\boldsymbol{h}_z^i$ and all search region tokens (candidates). The $j-th$ item ($1 \leq j \leq n$, n is the number of input search region tokens) of $\boldsymbol{w}_x^i$ determines the similarity between $\boldsymbol{h}_z^i$ and the $j-th$ candidate. However, the input templates usually include background regions that introduce noise when calculating the similarity between the target and each candidate. Therefore, instead of summing up the similarity of each candidate to all template parts $\boldsymbol{w}_x^i$, $i = 1, \ldots, N_z$, we take $\boldsymbol{w}_x^\phi, \phi = \lfloor \frac{W_z}{2} \rfloor + W_z \cdot \lfloor \frac{H_z}{2} \rfloor$ ($\phi - th$ token corresponding to the center part of the original template image) as the representative similarity. This is fairly reasonable as the center template part has aggregated enough information through self-attention to represent the target. In the supplementary, we compare the effect of different template token choices. Considering that multi-head self-attention is used in ViT, there are multiple similarity scores $\boldsymbol{w}_x^\phi(m)$, where $m = 1, ..., M$ and M is the total number of attention heads [38]. We average the similarity scores of all heads by $\overline{\boldsymbol{w}}_x^\phi = \sum_{m=1}^{M} \boldsymbol{w}_x^\phi(m)/M$, which serves as the final similarity score of the target and each candidate. One candidate is more likely to be a background region if its similarity score with the target is relatively small. Therefore, we only keep the candidates corresponding to the k largest (top-k) elements in $\overline{\boldsymbol{w}}_x^\phi$ (k is a hyperparameter, and we define the token keeping ratio as $\rho = k/n$), while the remaining candidates are eliminated. The proposed candidate elimination module is inserted after the multi-head attention operation [38] in the encoder layer, which is illustrated in Fig. 3(b). In addition, the original order of all remaining candidates is recorded so that it can be recovered in the final stage.

Candidate Restoration. The aforementioned candidate elimination module disrupts the original order of the candidates, making it impossible to reshape the candidate sequence back into the feature map as described in Sec. 3.3, so we restore the original order of the remaining candidates and then pad the missing positions. Since the discarded candidates belong to the irrelevant background regions, they will not affect the classification and regression tasks. In other words,

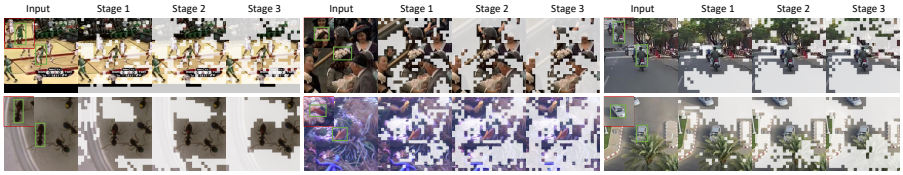

Fig. 5: Visualization of the progressive early candidate elimination process. The main body of "Input" is the search region image, and the upper left corner shows the corresponding template image. The Green rectangles indicate target objects and the masked regions represent the discarded candidates. The results show that our method can gradually identify and discard the candidates belonging to the background regions.

they just act as placeholders for the reshaping operation. Therefore, we first restore the order of the remaining candidates and then zero-pad in between them.

Visualization. To further investigate the behavior of the early candidate elimination module, we visualize the progressive process in Fig. 5. By iteratively discarding the irrelevant tokens in the search region, OSTrack not only largely lightens the computation burden but also avoids the negative impact of noisy background regions on feature learning.

3.3 Head and Loss

We first re-interpret the padded sequence of search region tokens to a 2D spatial feature map and then feed it into a fully convolutional network (FCN), which consists of L stacked Conv-BN-ReLU layers for each output. Outputs of the FCN contrain the target classification score map $\boldsymbol{P} \in [0,1]^{\frac{H_x}{P} \times \frac{W_x}{P}}$, the local offset $\boldsymbol{O} \in [0,1)^{2 \times \frac{H_x}{P} \times \frac{W_x}{P}}$ to compensate the discretization error caused by reduced resolution and the normalized bounding box size (*i.e.* width and height) $\boldsymbol{S} \in [0,1]^{2 \times \frac{H_x}{P} \times \frac{W_x}{P}}$. The position with highest classification score is considered to be target position, *i.e.*, $(x_d, y_d) = \arg\max_{(x,y)} \boldsymbol{P}_{xy}$ and the finial target bounding box is obtained as:

$$(x, y, w, h) = (x_d + \boldsymbol{O}(0, x_d, y_d), y_d + \boldsymbol{O}(1, x_d, y_d), \boldsymbol{S}(0, x_d, y_d), \boldsymbol{S}(1, x_d, y_d)). \quad (7)$$

During training, both classification and regression losses are used. We adopt the weighted focal loss [21] for classification (see the supplementary for more details). With the predicted bounding box, ℓ_1 loss and the generalized IoU loss [34] are employed for bounding box regression. Finally, the overall loss function is:

$$L_{track} = L_{cls} + \lambda_{iou} L_{iou} + \lambda_{L_1} L_1, \quad (8)$$

where $\lambda_{iou} = 2$ and $\lambda_{L_1} = 5$ are the regularization parameters in our experiments as in [43].

4 Experiments

After introducing the implementation details, this section first presents a comparison of OSTrack with other state-of-the-art methods on seven different benchmarks. Then, ablation studies are provided to analyze the impact of each component and different design choices.

4.1 Implementation Details

Our trackers are implemented in Python using PyTorch. The models are trained on 4 NVIDIA A100 GPUs and the inference speed is tested on a single NVIDIA RTX2080Ti GPU.

Model. The vanilla ViT-Base [10] model pre-trained with MAE [14] is adopted as the backbone for joint feature extraction and relation modeling. The head is a lightweight FCN, consisting of 4 stacked Conv-BN-ReLU layers for each of three outputs. The keeping ratio ρ of each candidate elimination module is set as 0.7, and a total of three candidate elimination modules are inserted at layers 4, 7, and 10 of ViT respectively, following [33]. We present two variants with different input image pair resolution for showing the scalability of OSTrack:

- **OSTrack-256.** Template: 128×128 pixels; Search region: 256×256 pixels.
- **OSTrack-384.** Template: 192×192 pixels; Search region: 384×384 pixels.

Training. The training splits of COCO [26], LaSOT [12], GOT-10k [17] (1k forbidden sequences from GOT-10k training set are removed following the convention [43]) and TrackingNet [31] are used for training. Common data augmentations including horizontal flip and brightness jittering are used in training. Each GPU holds 32 image pairs, resulting in a total batch size of 128. We train the model with AdamW optimizer [28], set the weight decay to 10^{-4}, the initial learning rate for the backbone to 4×10^{-5} and other parameters to 4×10^{-4}, respectively. The total training epochs are set to 300 with 60k image pairs per epoch and we decrease the learning rate by a factor of 10 after 240 epochs.

Inference. During inference, Hanning window penalty is adopted to utilize positional prior in tracking following the common practice [4,46]. Specifically, we simply multiply the classification map $\boldsymbol{P}$ by the Hanning window with the same size, and the box with the highest score after multiplication will be selected as the tracking result.

4.2 Comparison with State-of-the-arts

To demonstrate the effectiveness of the proposed models, we compare them with state-of-the-art (SOTA) trackers on seven different benchmarks.

GOT-10k. GOT-10k [17] test set employs the one-shot tracking rule, *i.e.*, it requires the trackers to be trained only on the GOT-10k training split, and the object classes between train and test splits are not overlapped. We follow this protocol to train our model and evaluate the results by submitting them to

Table 1: Comparison with state-of-the-arts on four large-scale benchmarks: LaSOT, LaSOT$_{\text{ext}}$, TrackingNet and GOT-10k[3]. The best two results are shown in **red** and **blue** fonts.

Method	Source	LaSOT [12]			LaSOT$_{\text{ext}}$ [11]			TrackingNet [31]			GOT-10k* [17]		
		AUC	P$_{Norm}$	P	AUC	P$_{Norm}$	P	AUC	P$_{Norm}$	P	AO	SR$_{0.5}$	SR$_{0.75}$
SiamFC [1]	ECCVW16	33.6	42.0	33.9	23.0	31.1	26.9	57.1	66.3	53.3	34.8	35.3	9.8
MDNet [32]	CVPR16	39.7	46.0	37.3	27.9	34.9	31.8	60.6	70.5	56.5	29.9	30.3	9.9
ECO [7]	ICCV17	32.4	33.8	30.1	22.0	25.2	24.0	55.4	61.8	49.2	31.6	30.9	11.1
SiamPRN++ [22]	CVPR19	49.6	56.9	49.1	34.0	41.6	39.6	73.3	80.0	69.4	51.7	61.6	32.5
DiMP [2]	ICCV19	56.9	65.0	56.7	39.2	47.6	45.1	74.0	80.1	68.7	61.1	71.7	49.2
SiamR-CNN [39]	CVPR20	64.8	72.2	-	-	-	-	81.2	85.4	80.0	64.9	72.8	59.7
MAMLTrack [40]	CVPR20	52.3	-	-	-	-	-	75.7	82.2	72.5	-	-	-
LTMU [5]	CVPR20	57.2	-	57.2	41.4	49.9	47.3	-	-	-	-	-	-
Ocean [46]	ECCV20	56.0	65.1	56.6	-	-	-	-	-	-	61.1	72.1	47.3
TrDiMP [41]	CVPR21	63.9	-	61.4	-	-	-	78.4	83.3	73.1	67.1	77.7	58.3
TransT [4]	CVPR21	64.9	73.8	69.0	-	-	-	81.4	86.7	80.3	67.1	76.8	60.9
AutoMatch [45]	ICCV21	58.3	-	59.9	-	-	-	76.0	-	72.6	65.2	76.6	54.3
STARK [43]	ICCV21	67.1	77.0	-	-	-	-	82.0	86.9	-	68.8	78.1	64.1
KeepTrack [29]	ICCV21	67.1	77.2	70.2	**48.2**	-	-	-	-	-	-	-	-
SwinTrack-B [25]	arXiv21	**69.6**	78.6	74.1	47.6	**58.2**	**54.1**	82.5	87.0	80.4	69.4	78.0	64.3
OSTrack-256	Ours	69.1	**78.7**	**75.2**	47.4	57.3	53.3	**83.1**	**87.8**	**82.0**	**71.0**	**80.4**	**68.2**
OSTrack-384	Ours	**71.1**	**81.1**	**77.6**	**50.5**	**61.3**	**57.6**	**83.9**	**88.5**	**83.2**	**73.7**	**83.2**	**70.8**

Table 2: Comparison with state-of-the-arts on three benchmarks: NFS [19], UAV123 [30] and TNL2K [42]. AUC(%) scores are reported. The best two results are shown in **red** and **blue** fonts.

	SiamFC [1]	RT-MDNet [18]	ECO [7]	Ocean [46]	ATOM [6]	DiMP50 [2]	STMTrack [13]	TransT [4]	STARK [43]	OSTrack -256	OSTrack -384
NFS	37.7	43.3	52.2	49.4	58.3	61.8	-	65.3	**66.2**	64.7	**66.5**
UAV123	46.8	52.8	53.5	57.4	63.2	64.3	64.7	68.1	68.2	**68.3**	**70.7**
TNL2K	29.5	-	32.6	38.4	40.1	44.7	-	50.7	-	**54.3**	**55.9**

the official evaluation server. As reported in Tab. 1, OSTrack-384 and OSTrack-256 outperform SwinTrack-B [25] by 1.6% and 4.3% in AO. The SR$_{0.75}$ score of OSTrack-384 reaches 70.8%, outperforming SwinTrack-B by 6.5%, which verifies the capability of our trackers in both accurate target-background discrimination and bounding box regression. Moreover, the high performance on this one-shot tracking benchmark demonstrates that our one-stream tracking framework can extract more discriminative features for unseen classes by mutual guidance.

LaSOT. LaSOT [12] is a challenging large-scale long-term tracking benchmark, which contains 280 videos for testing. We compare the result of the OSTrack with previous SOTA trackers in Tab. 1. The results show that the proposed tracker with smaller input resolution, *i.e.*, OSTrack-256, already obtains comparable performance with SwinTrack-B [25]. Besides, OSTrack-256 runs at a fast inference speed of 105.4 FPS, being 2x faster than SwinTrack-B (52 FPS), which indicates that OSTrack achieves an excellent balance between accuracy and inference speed. By increasing the input resolution, OSTrack-384 further improves the AUC on LaSOT to 71.1% and sets a new state-of-the-art.

[3] We add the symbol $*$ to GOT-10k if the corresponding models are trained following the one-shot protocol, otherwise they are trained with all training data.

Table 3: The effect of our proposed early candidate elimination module on the inference speed, MACs and tracking performance on LaSOT, GOT-10k and TrackingNet benchmarks, and w/o and w/ denote the models with or without early candidate elimination module separately.

Input	FPS		MACs (G)		LaSOT AUC (%)		TrackingNet AUC (%)		GOT-10k* AO (%)	
Resolution	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
256x256	93.1	105.4(+13.2%)	29.0	21.5(-25.9%)	68.7	69.1(+0.4)	82.9	83.1(+0.2)	71.0	71.0(+0.0)
384x384	41.4	58.1(+40.3%)	65.3	48.3(-26.0%)	71.0	71.1(+0.1)	83.5	83.9(+0.4)	73.5	73.7(+0.2)

TrackingNet. The TrackingNet [31] benchmark contains 511 sequences for testing, which covers diverse target classes. Tab. 1 shows that OSTrack-256 and OSTrack-384 surpass SwinTrack-B [25] by 0.6% and 1.4% in AUC separately. Moreover, both models are faster than SwinTrack-B.

LaSOT$_{\text{ext}}$. LaSOT$_{\text{ext}}$ [11] is a recently released extension of LaSOT, which consists of 150 extra videos from 15 object classes. Tab 1 presents the results. Previous SOTA tracker KeepTrack [29] designs a complex association network and runs at 18.3 FPS. In contrast, our simple one-stream tracker OSTrack-256 shows slightly lower performance but runs at 105.4 FPS. OSTrack-384 sets a new state-of-the-art AUC score of 50.5% while runs in 58.1 FPS, which is 2.3% higher in AUC score and 3x faster in speed.

NFS, UAV123 and TNL2K. We also evaluate our tracker on three additional benchmarks: NFS [19], UAV123 [30] and TNL2K [42] includes 100, 123, and 700 video sequences, separately. The results in Tab. 2 show that OSTrack-384 achieves the best performance on all three benchmarks, demonstrating the strong generalizability of OSTrack.

4.3 Ablation Study and Analysis

The Effect of Early Candidate Elimination Module Tab. 1 shows that increasing the input resolution of the input image pairs can bring significant performance gain. However, the quadratic complexity with respect to the input resolution makes simply increasing the input resolution unaffordable in inference time. The proposed early candidate elimination module addresses the above problem well. We present the effect of the early candidate elimination module from the aspects of inference speed (FPS), multiply-accumulate computations (MACs), and tracking performance on multiple benchmarks in Tab. 3. The effect on different input search region resolutions is also presented. Tab. 3 shows that the early candidate elimination module can significantly decrease the calculation and increase the inference speed, while slightly boosting the performance in most cases. This demonstrates that the proposed module alleviates the negative impact brought by the noisy background regions on feature learning. For example, adding the early candidate elimination module in OSTrack-256 decreases the MACs by 25.9% and increases the tracking speed by 13.2%, and the LaSOT AUC is increased by 0.4%. Furthermore, larger input resolution benefits more from this module, *e.g.*, OSTrack-384 shows a 40.3% increase in speed.

Table 4: The effect of different pre-training methods. All the models are trained without the early candidate elimination module.

Trackers	LaSOT			TrackingNet			GOT-10k		
	Success	P_{Norm}	P	Success	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$
No pre-training	60.4	70.0	62.8	77.5	83.0	73.8	62.7	72.8	53.7
ImageNet-1k	66.1	75.8	70.6	82.0	86.7	80.1	69.7	79.0	65.6
ImageNet-21k	66.9	76.3	71.2	82.4	86.9	80.1	70.2	80.7	65.4
MAE	**68.7**	**78.1**	**74.6**	**82.9**	**87.5**	**81.6**	**73.6**	**83.0**	**71.7**

Different Pre-training Methods. While previous Transformer fusion trackers [4,25,43] random initialize the weights of Transformer layers, our joint feature learning and relation modeling module can directly benefit from the pre-trained weights. We further investigate the effect of different pre-training methods on the tracking performance by comparing four different pre-training strategies: no pre-training; ImageNet-1k [8] pre-trained model provided by [37]; ImageNet-21k [35] pre-trained model provided by [36]; unsupervised pre-training model MAE [14]. As the results in Tab. 4 show, pre-training is necessary for the model weights initialization. Interestingly, we also observe that the unsupervised pre-training method MAE brings better tracking performance than the supervised pre-training ones using ImageNet. We hope this can inspire the community for designing better pre-training strategies tailored for the tracking task.

Aligned Comparison with SOTA Two-stream Trackers. One may wonder whether the performance gain is brought by the proposed one-stream structure or purely by the superiority of ViT. We thus compare our method with two SOTA two-stream Transformer fusion trackers [25, 43] by eliminating the influencing factors of backbone and head structure. To be specific, we align two previous SOTA two-stream trackers (STRAK-S [43] and SwinTrack [25]) with ours for fair comparison as follows: replacing their backbones with the same pre-trained ViT and setting the same input resolution, head structure, and training objective as OSTrack-256. The remaining experimental settings are kept the same as in the original paper. As shown in Tab. 5, our re-implemented two-stream trackers show comparable or stronger performance compared to the initially published performance, but still lag behind OSTrack, which demonstrates the effectiveness of our one-stream structure. We also observe that OSTrack significantly outperforms the previous two-stream trackers on the one-shot benchmark GOT-10k, which further proves the advantage of our one-stream framework in the challenging scenario. Actually, the discriminative power of features extracted by the two-stream framework is limited since the object classes in the testing set are completely different from the training set. Whereas, by iterative interaction between the features of the template and search region, OSTrack can extract more discriminative features through mutual guidance. Different from the two-stream SOTA trackers, OSTrack neglects the extra heavy relation modeling module while still keeping the high parallelism of joint feature extraction and relation modeling module. Therefore, when the same backbone network is adopted, the proposed one-stream framework is much faster than STARK (40.2

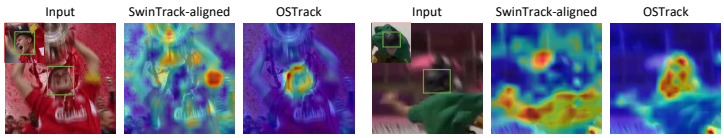

Fig. 6: Visualization of discriminative regions (*i.e.*, activation maps) of backbone features extracted by OSTrack and two-stream tracker (SwinTrack-aligned).

Table 5: Comparison with re-implemented previous SOTA trackers aligned with OSTrack. Here OSTrack is trained without the early candidate elimination module for fair comparison and "aligned" denotes that the backbone, head, loss and input resolution are kept the same as OSTrack.

Trackers	LaSOT			TrackingNet			GOT-10k*			FPS	Traing Pairs($\times 10^6$)
	Success	P_{Norm}	P	Success	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$		
STARK-aligned	67.6	76.3	72.8	82.6	87.4	81.5	68.8	78.4	65.6	52.9	30
SwinTrack-aligned	68.0	77.6	73.9	**82.9**	**87.6**	**81.6**	69.5	79.2	65.0	67.5	39.3
OSTrack	**68.7**	**78.1**	**74.6**	**82.9**	87.5	**81.6**	**71.0**	**80.3**	**68.2**	**93.1**	**18**

FPS faster) and SwinTrack (25.6 FPS faster). Besides, OSTrack requires fewer training image pairs to converge.

Discriminative Region Visualization. To better illustrate the effectiveness of the proposed one-stream tracker, we visualize the discriminative regions of the backbone features extracted by OSTrack and a SOTA two-stream tracker (SwinTrack-aligned) in Fig. 6. As can be observed, due to the lack of target awareness, features extracted by the backbone of SwinTrack-aligned show limited target-background discriminative power and may lose some important target information (*e.g.*, head and helmet in Fig. 6), which is irreparable. In contrast, OSTrack can extract discriminative target-oriented features, since the proposed early fusion mechanism enables relation modeling between the template and search region at the first stage.

5 Conclusion

This work proposes a simple, neat, and high-performance one-stream tracking framework based on Vision Transformer, which breaks out of the Siamese-like pipeline. The proposed tracker combines the feature extraction and relation modeling tasks, and shows a good balance between performance and inference speed. In addition, we further propose an early candidate elimination module that progressively discards search region tokens belonging to background regions, which significantly boosts the tracking inference speed. Extensive experiments show that the proposed one-stream trackers perform much better than previous methods on multiple benchmarks, especially under the one-shot protocol. We expect this work can attract more attention to the one-stream tracking framework.

Acknowledgments. This work is partially supported by Natural Science Foundation of China (NSFC): 61976203 and 61876171. Thanks Zhipeng Zhang for his helpful suggestions.

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV. pp. 850–865 (2016) 2, 3, 11
2. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: ICCV. pp. 6182–6191 (2019) 2, 3, 11
3. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR. pp. 2544–2550 (2010) 3
4. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: CVPR. pp. 8126–8135 (2021) 2, 4, 6, 7, 10, 11, 13
5. Dai, K., Zhang, Y., Wang, D., Li, J., Lu, H., Yang, X.: High-performance long-term tracking with meta-updater. In: CVPR. pp. 6298–6307 (2020) 11
6. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: CVPR. pp. 4660–4669 (2019) 2, 3, 11
7. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: Efficient convolution operators for tracking. In: CVPR. pp. 6638–6646 (2017) 11
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009) 7, 13
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL. pp. 4171–4186 (2019) 6
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021) 2, 5, 6, 10
11. Fan, H., Bai, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Huang, M., Liu, J., Xu, Y., et al.: Lasot: A high-quality large-scale single object tracking benchmark. IJCV **129**(2), 439–461 (2021) 11, 12
12. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: CVPR. pp. 5374–5383 (2019) 10, 11
13. Fu, Z., Liu, Q., Fu, Z., Wang, Y.: Stmtrack: Template-free visual tracking with space-time memory networks. In: CVPR. pp. 13774–13783 (2021) 11
14. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021) 5, 10, 13
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 3, 7
16. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI **37**(3), 583–596 (2014) 3
17. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. TPAMI **43**(5), 1562–1577 (2019) 10, 11
18. Jung, I., Son, J., Baek, M., Han, B.: Real-time mdnet. In: ECCV. pp. 83–98 (2018) 11
19. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV. pp. 1125–1134 (2017) 11, 12
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NeurIPS **25** (2012) 3

21. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: ECCV. pp. 734–750 (2018) 9
22. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: SiamRPN++: Evolution of siamese visual tracking with very deep networks. In: CVPR. pp. 4282–4291 (2019) 3, 11
23. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR. pp. 8971–8980 (2018) 2, 3
24. Liang, Y., GE, C., Tong, Z., Song, Y., Wang, J., Xie, P.: EViT: Expediting vision transformers via token reorganizations. In: ICLR (2022) 4
25. Lin, L., Fan, H., Xu, Y., Ling, H.: Swintrack: A simple and strong baseline for transformer tracking. arXiv preprint arXiv:2112.00995 (2021) 7, 11, 12, 13
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014) 10
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 10012–10022 (2021) 7
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018) 10
29. Mayer, C., Danelljan, M., Paudel, D.P., Van Gool, L.: Learning target candidate association to keep track of what not to track. In: ICCV. pp. 13444–13454 (2021) 11, 12
30. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: ECCV. pp. 445–461 (2016) 11, 12
31. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV. pp. 300–317 (2018) 10, 11, 12
32. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. pp. 4293–4302 (2016) 11
33. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. NeurIPS **34** (2021) 4, 10
34. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: CVPR. pp. 658–666 (2019) 9
35. Ridnik, T., Ben-Baruch, E., Noy, A., Zelnik-Manor, L.: ImageNet-21k pretraining for the masses. arXiv preprint arXiv:2104.10972 (2021) 13
36. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021) 5, 13
37. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. pp. 10347–10357 (2021) 13
38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS. pp. 5998–6008 (2017) 2, 3, 5, 6, 8
39. Voigtlaender, P., Luiten, J., Torr, P.H., Leibe, B.: Siam r-cnn: Visual tracking by re-detection. In: CVPR. pp. 6578–6588 (2020) 11
40. Wang, G., Luo, C., Sun, X., Xiong, Z., Zeng, W.: Tracking by instance detection: A meta-learning approach. In: CVPR. pp. 6288–6297 (2020) 11
41. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: CVPR. pp. 1571–1580 (2021) 11

42. Wang, X., Shu, X., Zhang, Z., Jiang, B., Wang, Y., Tian, Y., Wu, F.: Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In: CVPR. pp. 13763–13773 (2021) 11, 12
43. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: ICCV. pp. 10448–10457 (2021) 2, 4, 9, 10, 11, 13
44. Yin, H., Vahdat, A., Alvarez, J., Mallya, A., Kautz, J., Molchanov, P.: Adavit: Adaptive tokens for efficient vision transformer. arXiv preprint arXiv:2112.07658 (2021) 4
45. Zhang, Z., Liu, Y., Wang, X., Li, B., Hu, W.: Learn to match: Automatic matching network design for visual tracking. In: ICCV. pp. 13339–13348 (2021) 11
46. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: ECCV. pp. 771–787 (2020) 3, 10, 11