SparseTT: Visual Tracking with Sparse Transformers

Zhihong Fu, **Zehua Fu**, **Qingjie Liu***, **Wenrui Cai** and **Yunhong Wang**
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
Hangzhou Innovation Institute, Beihang University
{fuzhihong, zehua_fu, qingjie.liu, wenrui_cai, yhwang}@buaa.edu.cn

Abstract

Transformers have been successfully applied to the visual tracking task and significantly promote tracking performance. The self-attention mechanism designed to model long-range dependencies is the key to the success of Transformers. However, self-attention lacks focusing on the most relevant information in the search regions, making it easy to be distracted by background. In this paper, we relieve this issue with a sparse attention mechanism by focusing the most relevant information in the search regions, which enables a much accurate tracking. Furthermore, we introduce a double-head predictor to boost the accuracy of foreground-background classification and regression of target bounding boxes, which further improve the tracking performance. Extensive experiments show that, without bells and whistles, our method significantly outperforms the state-of-the-art approaches on LaSOT, GOT-10k, TrackingNet, and UAV123, while running at 40 FPS. Notably, the training time of our method is reduced by 75% compared to that of TransT. The source code and models are available at https://github.com/fzh0917/SparseTT.

1 Introduction

Visual tracking aims to predict the future states of a target given its initial state. It is applicable broadly, such as human-computer interactions, video surveillance, and autonomous driving. Most of the existing methods address the tracking problem with sequence prediction frameworks where they estimate the current state based on the initial and the previous states. Thus, it is important to give accurate states in every time slice otherwise errors accumulate and will lead to tracking failure. Significant efforts have been devoted to improving the tracking accuracy, i.e., the accuracy of the target bounding boxes. However, challenges such as target deformation, partial occlusion, and scale variation are still huge obstacles ahead hindering them from perfect tracking. The reason may be that most of these methods adopt cross-correlation operation to measure similarities between the target template and the search region, which may trap into local optimums. Recently, TransT [Chen *et al.*, 2021] and DTT [Yu *et al.*, 2021] improve the tracking performance by replacing the correlation with Transformer [Vaswani *et al.*, 2017]. However, building trackers with Transformers will lead to a new problem: the global perspective of self-attention in Transformers causes the primary information (such as targets in search regions) under-focused, but the secondary information (such as background in search regions) over-focused, making the edge region between the foreground and background blurred, and thus degrade the tracking performance.

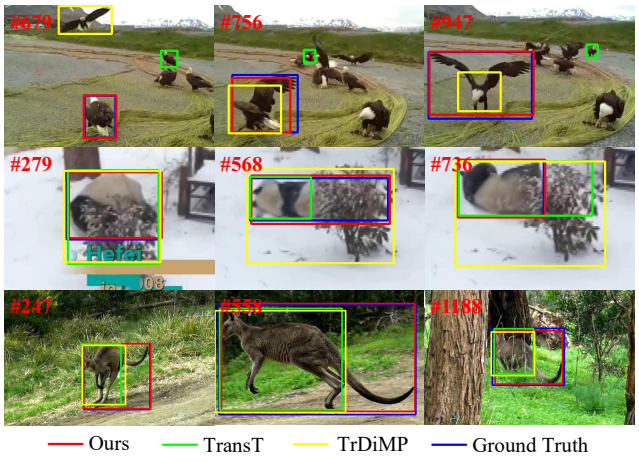

Figure 1: Visualized comparisons of our method with excellent trackers TransT [Chen *et al.*, 2021] and TrDiMP [Wang *et al.*, 2021]. Our method enables the bounding boxes of targets to be more accurate even under severe target deformation, partial occlusion, and scale variation. Zoom in for better view.

In this paper, we attack this issue by concentrating on the most relevant information of the search region, which is realized with a sparse Transformer. Different from vanilla Transformers used in previous works [Chen *et al.*, 2021; Yu *et al.*, 2021], sparse Transformer is designed to focus on primary information, enabling the targets to be more discriminative and the bounding boxes of targets to be more accurate even under severe target deformation, partial occlusion, scale variation, and so on, as shown in Fig. 1.

Summarily, the main contributions of this work are three-

*Contact Author

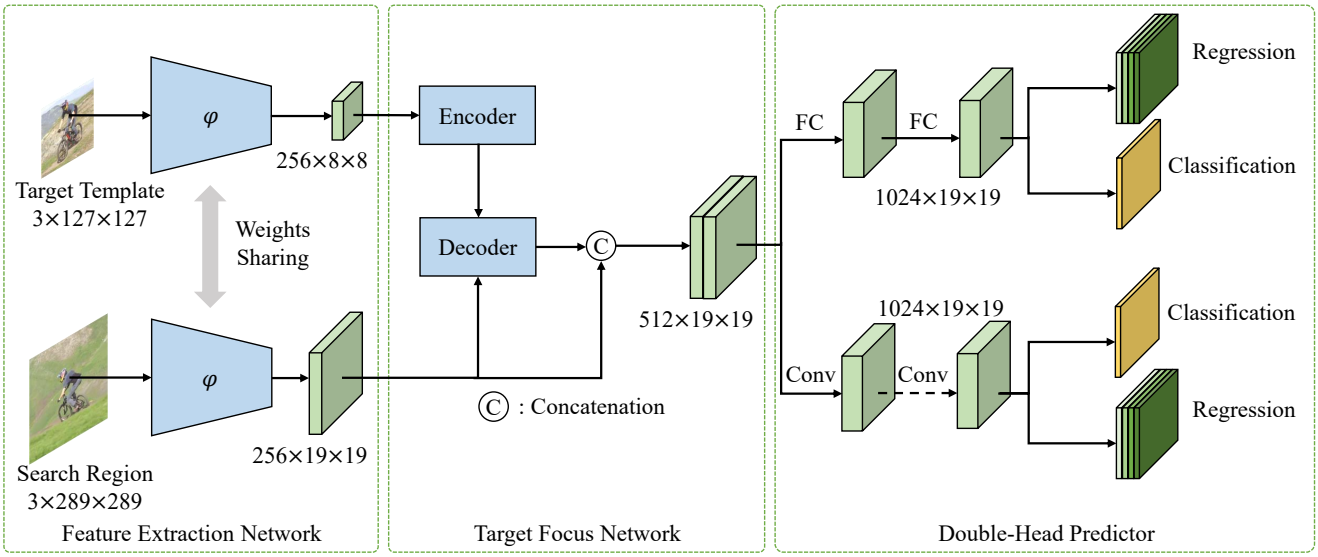

Figure 2: The architecture of our method.

fold.

- We present a target focus network that is capable of focusing on the target of interest in the search region and highlighting the features of the most relevant information for better estimating the states of the target.
- We propose a sparse Transformer based siamese tracking framework that has a strong ability to deal with target deformation, partial occlusion, scale variation, and so on.
- Extensive experiments show that our method outperforms the state-of-the-art approaches on LaSOT, GOT-10k, TrackingNet, and UAV123, while running at 40 FPS, demonstrating the superiority of our method.

2 Related Work

Siamese Trackers. In siamese visual trackers, cross-correlation, commonly used to measure the similarity between the target template and the search region, has been extensively studied for visual tracking. Such as naive cross-correlation [Bertinetto *et al.*, 2016], depth-wise cross-correlation [Li *et al.*, 2019; Xu *et al.*, 2020], pixel-wise cross-correlation [Yan *et al.*, 2021b], pixel to global matching cross-correlation [Liao *et al.*, 2020], etc. However, cross-correlation performs local linear matching processes, which may fall into local optimum easily [Chen *et al.*, 2021]. And furthermore, the cross-correlation captures relationships and thus corrupts semantic information of the inputted features, which is adverse to accurate perception of target boundaries. Most siamese trackers still have difficulties dealing with target deformation, partial occlusion, scale variation, etc.

Transformer in Visual Tracking. Recently, Transformers have been successfully applied to visual tracking field. Borrowing inspiration from DETR [Carion *et al.*, 2020], STARK [Yan *et al.*, 2021a] casts target tracking as a bounding box prediction problem and solve it with an encoder-decoder transformer, in which the encoder models the global spatio-temporal feature dependencies between targets and search regions, and the decoder learns a query embedding to predict the spatial positions of the targets. It achieves excellent performance on visual tracking. TrDiMP [Wang *et al.*, 2021] designs a siamese-like tracking pipeline where the two branches are built with CNN backbones followed by a Transformer encoder and a Transformer decoder, respectively. The Transformers here are used to enhance the target templates and the search regions. Similar to previous siamese trackers, TrDiMP applies cross-correlation to measure similarities between the target templates and the search region, which may impede the tracker from high-performance tracking. Noticing this shortcoming, TransT [Chen *et al.*, 2021] and DTT [Yu *et al.*, 2021] propose to replace cross-correlation with Transformer, thereby generating fused features instead of response scoress. Since fused features contain rich semantic information than response scores, these methods reach much accurate tracking than previous siamese trackers.

Self-attention in Transformers specializes in modeling long-rang dependencies, making it good at capturing global information, however, suffering from a lack of focusing on the most relevant information in the search regions. To further boost Transformer trackers, we alleviate the aforementioned drawback of self-attention with a sparse attention mechanism. The idea is inspired by [Zhao *et al.*, 2019]. We adapt the sparse Transformer in [Zhao *et al.*, 2019] to suit the visual tracking task and propose a new end-to-end siamese tracker with an encoder-decoder sparse Transformer. Driven by the sparse attention mechanism, the sparse Transformer focuses on the most relevant information in the search regions, thus suppressing distractive background that disturbs the tracking more efficiently.

3 Method

We propose a siamese architecture for visual tracking, which consists of a feature extraction network, a target focus network, and a double-head predictor, as shown in Fig. 2. The feature extraction network is a weight-shared backbone. The target focus network built with a sparse Transformer is used to generate target-focused features. The double-head predictor discriminates foreground from background and outputs bounding boxes of the target. Note that our method runs at a real-time speed as no online updating in the tracking phase.

3.1 Target Focus Network

The target focus network is built with sparse Transformer, and it has an encoder-decoder architecture, as shown in Fig. 3. The encoder is responsible for encoding the target template features. The decoder is responsible for decoding the search region features to generate the target-focused features.

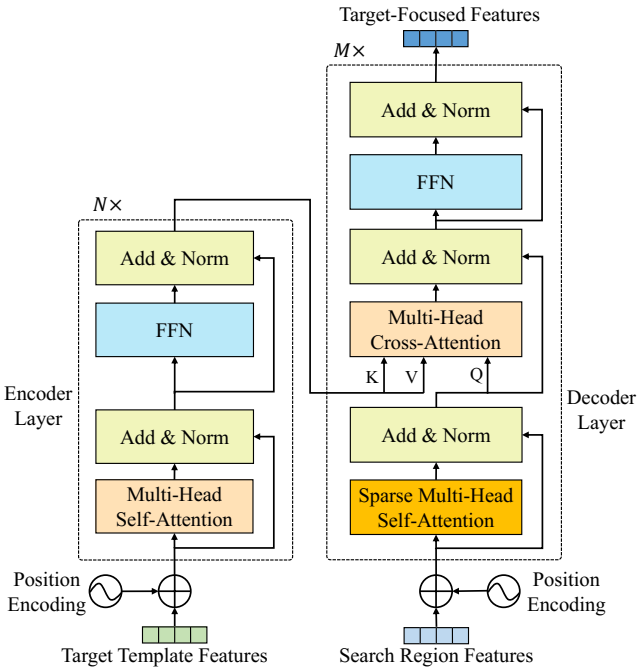

Figure 3: The architecture of target focus network.

3.2 Encoder

Encoder is an **important but not essential** component in the proposed target focus network. It is composed of N encoder layers where each encoder layer takes the outputs of its previous encoder layer as input. Note that, in order to enable the network to have the perception of spatial position information, we add a spatial position encoding to the target template features, and input the sum to the encoder. Thus, the first encoder layer takes the target template features with spatial position encoding as input. In short, it can be formally denoted as:

$$\text{encoder}(\boldsymbol{Z}) = \begin{cases} f_{enc}^{i}\left(\boldsymbol{Z} + \boldsymbol{P}_{enc}\right), & i = 1 \\ f_{enc}^{i}\left(\boldsymbol{Y}_{enc}^{i-1}\right), & 2 \leq i \leq N \end{cases} \tag{1}$$

where $\boldsymbol{Z} \in \mathbb{R}^{H_t W_t \times C}$ represents the target template features, $\boldsymbol{P}_{enc} \in \mathbb{R}^{H_t W_t \times C}$ represents the spatial position encoding, f_{enc}^{i} represents the i-th encoder layer, $\boldsymbol{Y}_{enc}^{i-1} \in \mathbb{R}^{H_t W_t \times C}$ represents the output of the $(i-1)$-th encoder layer. H_t and W_t are the height and width of the feature maps of target templates, respectively.

In each encoder layer, we use multi-head self-attention (MSA) to explicitly model the relations between all pixel pairs of target template features. Other operations are the same as the encoder layer of vanilla Transformer [Vaswani *et al.*, 2017].

3.3 Decoder

Decoder is an **essential** component in the proposed target focus network. Similar to the encoder, the decoder is composed of M decoder layers. However, different from the encoder layer, each decoder layer not only inputs the search region features with spatial position encoding or the output of its previous decoder layer, but also inputs the encoded target template features outputted by the encoder. In short, it can be formally denoted as:

$$\text{decoder}(\boldsymbol{X}, \boldsymbol{Y}_{enc}^{N}) = \begin{cases} f_{dec}^{i}\left(\boldsymbol{X} + \boldsymbol{P}_{dec}, \boldsymbol{Y}_{enc}^{N}\right), i = 1 \\ f_{dec}^{i}\left(\boldsymbol{Y}_{dec}^{i-1}, \boldsymbol{Y}_{enc}^{N}\right), 2 \leq i \leq M \end{cases} \tag{2}$$

where $\boldsymbol{X} \in \mathbb{R}^{H_s W_s \times C}$ represents the search region features, $\boldsymbol{P}_{dec} \in \mathbb{R}^{H_s W_s \times C}$ represents the spatial position encoding, $\boldsymbol{Y}_{enc}^{N} \in \mathbb{R}^{H_t W_t \times C}$ represents the encoded target template features outputted by the encoder, f_{dec}^{i} represents the i-th decoder layer, $\boldsymbol{Y}_{dec}^{i-1} \in \mathbb{R}^{H_s W_s \times C}$ represents the output of $(i-1)$-th decoder layer. H_s and W_s are height and width of the feature maps of search regions, respectively.

Different from the decoder layer of vanilla Transformer [Vaswani *et al.*, 2017], each decoder layer of the proposed sparse Transformer first calculates self-attention on $\boldsymbol{X}$ using sparse multi-head self-attention (SMSA), then calculates cross-attention between $\boldsymbol{Z}$ and $\boldsymbol{X}$ using naive multi-head cross-attention (MCA). Other operations are the same as the decoder layer of vanilla Transformer [Vaswani *et al.*, 2017]. Formally, each decoder layer of the proposed sparse Transformer can be denoted as:

$$\begin{aligned} \hat{\boldsymbol{X}} &= \text{Norm}\left(\text{SMSA}\left(\boldsymbol{Y}_{dec}^{i-1}\right) + \boldsymbol{Y}_{dec}^{i-1}\right) \\ \hat{\boldsymbol{Y}}_{dec}^{i} &= \text{Norm}\left(\text{MCA}\left(\hat{\boldsymbol{X}}, \boldsymbol{Y}_{enc}^{N}, \boldsymbol{Y}_{enc}^{N}\right) + \hat{\boldsymbol{X}}\right) \\ \boldsymbol{Y}_{dec}^{i} &= \text{Norm}\left(\text{FFN}\left(\hat{\boldsymbol{Y}}_{dec}^{i}\right) + \hat{\boldsymbol{Y}}_{dec}^{i}\right) \end{aligned} \tag{3}$$

3.4 Sparse Multi-Head Self-Attention

Sparse multi-head self-attention is designed to improve the discrimination of foreground-background and to alleviate ambiguity of edge regions of foreground. Concretely, in the naive MSA, each pixel value of attention features is calculated by all pixel values of the input features, which makes the edge regions of foreground blurred. In our proposed SMSA, each pixel value of attention features is only determined by K pixel values that are most similar to it, which

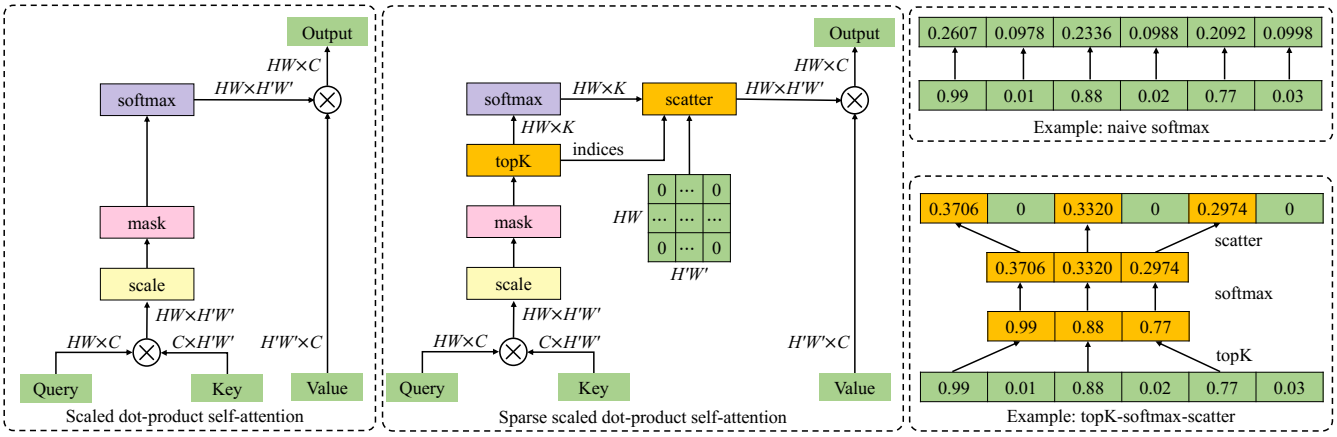

Figure 4: The left is the illustration of scaled dot-product self-attention in MSA, the middle is the illustration of the sparse scaled dot-product self-attention in SMSA, where the function `scatter` means filling given values into a 0-value matrix at given indices. The upper right and the lower right are examples of normalizing a row vector of the similarity matrix in naive scaled dot-product attention and sparse scaled dot-product attention, respectively.

makes foreground more focused and the edge regions of foreground more discriminative.

Specifically, as shown in the middle of Fig. 4, given a ***query*** $\in \mathbb{R}^{HW\times C}$, a ***key*** $\in \mathbb{R}^{C\times H'W'}$, and a ***value*** $\in \mathbb{R}^{H'W'\times C}$, we first calculate similarities of all pixel pairs between ***query*** and ***key*** and mask out unnecessary tokens in the similarity matrix. Then, different from naive scaled dot-product attention that is shown in the left of Fig. 4, we only normalize K largest elements from each row of the similarity matrix using `softmax` function. For other elements, we replace them with 0. Finally, we multiply the similarity matrix and ***value*** by matrix multiplication to get the final results.

The upper right and the lower right in Fig. 4 show examples of normalizing a row vector of the similarity matrix in naive scaled dot-product attention and sparse scaled dot-product attention, respectively. We can see that naive scaled dot-product attention amplifies relatively smaller similarity weights, which makes the output features susceptible to noises and distractive background. However, this issue can be significantly alleviated by sparse scaled dot-product attention.

3.5 Double-Head Predictor

Most existing trackers adopt fully connected network or convolutional network to classification between foreground and background and regression of target bounding boxes, without indepth analysis or design for the head networks based on the characteristics of the tasks of classification and regression. Inspired by [Wu *et al.*, 2020], we introduce a double-head predictor to improve the accuracy of classification and regression. Specifically, as shown in Fig. 2, it consists of a *fc-head* that is composed of two fully connected layers and a *conv-head* that is composed of L convolutional blocks. Unfocused tasks are added for extra supervision in training. In the inference phase, for the classification task, we fuse the classification scores outputted by the *fc-head* and the one outputted by the *conv-head*; for the regression task, we only take the predicted offsets outputted by the *conv-head*.

3.6 Training Loss

We follow [Xu *et al.*, 2020] to generate training labels of classification scores and regression offsets. In order to train the whole network end-to-end, the objective function is the weighted sum of classification loss and regression loss, as the following:

$$\begin{aligned}\mathcal{L} = \omega_{fc}\cdot\left[\lambda_{fc}\mathcal{L}_{fc}^{class} + (1-\lambda_{fc})\,\mathcal{L}_{fc}^{box}\right] \\ + \omega_{conv}\cdot\left[(1-\lambda_{conv})\,\mathcal{L}_{conv}^{class} + \lambda_{conv}\mathcal{L}_{conv}^{box}\right]\end{aligned} \tag{4}$$

where ω_{fc}, λ_{fc}, ω_{conv} and λ_{conv} are hyper-parameters. In practice, we set $\omega_{fc}=2.0$, $\lambda_{fc}=0.7$, $\omega_{conv}=2.5$, $\lambda_{conv}=0.8$. The functions $\mathcal{L}_{fc}^{class}$ and $\mathcal{L}_{conv}^{class}$ are both implemented by focal loss [Lin *et al.*, 2017], and the functions $\mathcal{L}_{fc}^{box}$ and $\mathcal{L}_{conv}^{box}$ are both implemented by IoU loss [Yu *et al.*, 2016].

4 Experiments

4.1 Implementation Details

Training Dataset. We use the `train` splits of TrackingNet [Muller *et al.*, 2018], LaSOT [Fan *et al.*, 2019], GOT-10k [Huang *et al.*, 2019], ILSVRC VID [Russakovsky *et al.*, 2015], ILSVRC DET [Russakovsky *et al.*, 2015] and COCO [Lin *et al.*, 2014] as the training dataset, in addition to the GOT-10k [Huang *et al.*, 2019] benchmark. We select two frames with a maximum frame index difference of 100 from each video as the target template and the search region. In order to increase the diversity of training samples, we set the range of random scaling to $\left[\frac{1}{1+\alpha}, 1+\alpha\right]$ and the range of random translation to $[-0.2\beta, 0.2\beta]$, in which $\alpha=0.3$, $\beta=\sqrt{(1.5w_t+0.5h_t)\times(1.5h_t+0.5w_t)}$ for the target template, and $\beta=\frac{t\cdot s}{\sqrt{(1.5w_s+0.5h_s)\times(1.5h_s+0.5w_s)}}$ for the search region. Here w_t and h_t are the width and height of the target in the target template, respectively; w_s and h_s are the width and height of the target in the search region, respectively; t and s are the sizes of the target template and the

search region, respectively. We set $t = 127$ and $s = 289$ in practice.

Model Settings. We use the tiny version of Swin Transformer [Liu *et al.*, 2021] (Swin-T) as the backbone φ. In the MSA, SMSA, and MCA, the number of heads is set to 8, the number of channels in the hidden layers of FFN is set to 2048, and the dropout rate is set to 0.1. The number of encoder layers N and the number of decoder layers M are set to 2, and the sparseness K in SMSA is set to 32. See Sec. 4.2 for more discussions about the hyper parameters in the proposed target focus network. In the *conv-head* of the double-head predictor, the first convolutional block is set to residual block [He *et al.*, 2016], and other $L - 1$ ones are set to bottleneck blocks [He *et al.*, 2016], where $L = 8$.

Optimization. We use AdamW optimizer to train our method for 20 epochs. In each epoch, we sample 600,000 image pairs from all training datasets. Note that we only sample 300,000 image pairs from the `train` split for the GOT-10k benchmark. The batch size is set to 32, and the learning rate and the weight decay are both set to 1×10^{-4}. After training for 10 epochs and 15 epochs, the learning rate decreases to 1×10^{-5} and 1×10^{-6}, respectively. The whole training process takes about 60 hours on 4 NVIDIA RTX 2080 Ti GPUs. Note that the training time of TransT is about 10 days (240 hours), which is $4\times$ that of our method.

4.2 Ablation Study

The Number of Encoder Layers. In our method, the encoder is used to enhance the generalization of target template, thus the number of encoder layers is important to our method. Tab. 1 lists the performance of our method using different numbers of encoder layers. Interestingly, the proposed target focus network can still bring comparable performance without the encoder. As the number increases, the performance gradually improves. However, when the number of encoder layers is greater than 2, the performance drops. We argue that excess encoder layers may lead to overfitting of model training. Therefore, we set the number of encoder layers to 2 in the remaining experiments.

N	**0**	**1**	**2**	**3**
AO	0.676	0.687	**0.693**	0.679
SR$_{0.5}$	0.770	0.783	**0.791**	0.770
SR$_{0.75}$	0.627	0.634	**0.638**	0.620

Table 1: The performance of our method on the `test` split of GOT-10k when setting the number of encoder layers to 0, 1, 2, and 3.

The Number of Decoder Layers. We then explore the best setting for the number of decoder layers M, as shown in Tab. 2. Similar to N, as the number of decoder layers increases, the performance gradually improves when M is not greater than 2. We also notice that when M equals 3, the performance decreases and the running speed slows down by large margin. We speculate that it may be caused by overfitting. Thus, M is set to 2 in the remaining experiments.

M	**1**	**2**	**3**
AO	0.672	**0.693**	0.661
SR$_{0.5}$	0.764	**0.791**	0.754
SR$_{0.75}$	0.619	**0.638**	0.610
FPS	40.2	39.9	37.7

Table 2: The performance of our method on the `test` split of GOT-10k when setting the number of decoder layers to 1, 2, and 3.

The Sparseness K in SMSA. In SMSA, the sparseness K significantly affects the activation degree of foreground. Due to the scale variation of targets, a suitable sparseness K ensures good adaptability and generalization at the same time for SMSA. Tab. 3 shows the impact of different sparseness values on the performance of our method. Note that when $K = H'W'$, SMSA becomes naive MSA. We find that SMSA always brings better performance than MSA in our method, which shows the effectiveness and superiority of SMSA. When K is 32, Our method achieves the best performance. Consequently, we set the sparseness K to 32 in our experiments.

K	**16**	**32**	**64**	**128**	**256**	$\boldsymbol{H'W'}$
AO	0.667	**0.693**	0.680	0.677	0.682	0.662
SR$_{0.5}$	0.763	**0.791**	0.777	0.771	0.780	0.754
SR$_{0.75}$	0.611	**0.638**	0.627	0.623	0.627	0.605

Table 3: The performance of our method on the `test` split of GOT-10k when setting different sparseness values for SMSA, where $H'W'$ denotes the number of columns of the similarity matrix.

4.3 Comparison with the State-of-the-art

LaSOT is a large-scale long-term dataset with high-quality annotations. Its `test` split consists of 280 sequences, the average length of which exceeds 2500 frames. We evaluate our method on the `test` split of LaSOT and compare it with other competitive methods. As shown in Tab. 4, our method achieves the best performance in terms of success, precision, and normalized precision metrics.

We also evaluate our method on the test subsets with attributes of deformation, partial occlusion, and scale variation. The results are shown in Tab. 8. As can be seen, our method performs best in the above challenging scenarios, significantly surpassing other competitive methods. These challenges bring ambiguous of determining accurate boundaries of targets thus making the trackers hard to locate and estimate target bounding boxes. However, our method copes with these challenges well.

GOT-10k contains 9335 sequences for training and 180 sequences for testing. Different from other datasets, GOT-10k only allows trackers to be trained using the `train` split. We follow this protocol to train our method and test it on the `test` split, then report the performance in Tab. 5. We see that our method surpasses the second-best tracker TransT by a significant margin, which indicates that our method is superior to other methods when annotated training data is limited.

Method	Succ.	Prec.	N. Prec.
Ours	**0.660**	**0.701**	**0.748**
TransT [Chen *et al.*, 2021]	**0.649**	**0.690**	**0.738**
TrDiMP [Wang *et al.*, 2021]	0.639	0.662	0.730
SAOT [Zhou *et al.*, 2021]	0.616	0.629	0.708
STMTrack [Fu *et al.*, 2021]	0.606	0.633	0.693
DTT [Yu *et al.*, 2021]	0.601	-	-
AutoMatch [Zhang *et al.*, 2021]	0.583	0.599	0.675
SiamRCR [Peng *et al.*, 2021]	0.575	0.599	-
LTMU [Dai *et al.*, 2020]	0.570	0.566	0.653
DiMP-50 [Bhat *et al.*, 2019]	0.565	0.563	0.646
Ocean [Zhang *et al.*, 2020]	0.560	0.566	0.651
SiamFC++ [Xu *et al.*, 2020]	0.543	0.547	0.623
SiamGAT [Guo *et al.*, 2021]	0.539	0.530	0.633

Table 4: The performance of our method and other excellent ones on the `test` split of LaSOT, where "Succ.", "Prec." and "N. Prec." represent success, precision and normalized precision, respectively. The best two results are highlighted in **red** and **blue**, respectively.

Method	AO	$SR_{0.5}$	$SR_{0.75}$
Ours	**0.693**	**0.791**	**0.638**
TransT [Chen *et al.*, 2021]	**0.671**	0.768	**0.609**
TrDiMP [Wang *et al.*, 2021]	0.671	**0.777**	0.583
AutoMatch [Zhang *et al.*, 2021]	0.652	0.766	0.543
STMTrack [Fu *et al.*, 2021]	0.642	0.737	0.575
SAOT [Zhou *et al.*, 2021]	0.640	0.749	-
KYS [Bhat *et al.*, 2020]	0.636	0.751	0.515
DTT [Yu *et al.*, 2021]	0.634	0.749	0.514
PrDiMP [Danelljan *et al.*, 2020]	0.634	0.738	0.543
SiamGAT [Guo *et al.*, 2021]	0.627	0.743	0.488
SiamRCR [Peng *et al.*, 2021]	0.624	-	-
DiMP-50 [Bhat *et al.*, 2019]	0.611	0.717	0.492

Table 5: The performance of our method and other excellent ones on the `test` split of GOT-10k. The best two results are highlighted in **red** and **blue**, respectively.

UAV123 is a low altitude aerial dataset taken by drones, including 123 sequences, with an average of 915 frames per sequence. Due to the characteristics of aerial images, many targets in this dataset have low resolution, and are prone to have fast motion and motion blur. In spite of this, our method is still able to cope with these challenges well. Thus, as shown in Tab. 6, our method surpasses other competitive methods and achieves the state-of-the-art performance on UAV123, which demonstrates the generalization and applicability of our method.

OTB2015 is a classical testing dataset in visual tracking. It contains 100 short-term tracking sequences covering 11 common challenges, such as target deformation, occlusion, scale variation, rotation, illumination variation, background clutters, and so on. We report the performance of our method on OTB2015. Although the annotations is not very accurate and it has tended to saturation over recent years, as shown in Tab. 6, however, our method still outperforms the excellent tracker TransT [Chen *et al.*, 2021] and achieves comparable performance.

TrackingNet is a large-scale dataset whose `test` split includes 511 sequences covering various object classes and tracking scenes. We report the performance of our method on the `test` split of TrackingNet. As shown in Tab. 7, our method achieves the best performance in terms of success metric.

Method	UAV123	OTB2015
Ours	**0.704**	0.704
TransT [Chen *et al.*, 2021]	**0.691**	0.694
PrDiMP [Danelljan *et al.*, 2020]	0.680	0.696
TrDiMP [Wang *et al.*, 2021]	0.675	**0.711**
DiMP-50 [Bhat *et al.*, 2019]	0.654	0.684
STMTrack [Fu *et al.*, 2021]	0.647	**0.719**

Table 6: The performance of our method and other excellent ones on UAV123 and OTB2015. The best two results are highlighted in **red** and **blue**, respectively.

Method	Succ.	Prec.	N. Prec.
Ours	**81.7**	**79.5**	**86.6**
TransT [Chen *et al.*, 2021]	**81.4**	**80.3**	**86.7**
STMTrack [Fu *et al.*, 2021]	80.3	76.7	85.1
DTT [Yu *et al.*, 2021]	79.6	78.9	85.0
TrDiMP [Wang *et al.*, 2021]	78.4	73.1	83.3
SiamRCR [Peng *et al.*, 2021]	76.4	71.6	81.8
AutoMatch [Zhang *et al.*, 2021]	76.0	72.6	-
PrDiMP [Danelljan *et al.*, 2020]	75.8	70.4	81.6
SiamFC++ [Xu *et al.*, 2020]	75.4	70.5	80.0
DiMP-50 [Bhat *et al.*, 2019]	74.0	68.7	80.1

Table 7: The performance of our method and other excellent ones on the `test` split of TrackingNet, where "Succ.", "Prec." and "N. Prec." represent success, precision and normalized precision, respectively. The best two results are highlighted in **red** and **blue**, respectively.

4.4 Qualitative Comparison of SMSA and MSA

To intuitively explore how SMSA works, we visualize some self-attention maps of search regions in Fig. 5, in which the 1-st column and the 4-th column are the search regions, the 2-nd column and the 5-th column are the attention maps generated by SMSA and naive MSA, respectively. For better visualization, we combine the 1-st column and the 2-nd column in the 3-rd column and combine the 4-th column and the 5-th column in the 6-th column. We can see that, compared with MSA, SMSA pays more attention to primary information.

5 Conclusions

In this work, we boost Transformer based visual tracking with a novel sparse Transformer tracker. The sparse self-attention mechanism in Transformer relieves the issue of concentration on the global context and thus negligence of the most relevant information faced by the vanilla self-attention mechanism, thereby highlighting potential targets in the search regions. In addition, a double-head predictor is introduced to improve the accuracy of classification and regression. Experiments show that our method can significantly outperform the state-of-the-art approaches on multiple datasets while running at a

Method	Deformation		Partial Occlusion		Scale Variation		Rotation		Viewpoint Change	
	Succ.	Prec.	Succ.	Prec.	Succ.	Prec.	Succ.	Prec.	Succ.	Prec.
Ours	**0.685**	**0.693**	**0.634**	**0.665**	**0.660**	**0.700**	**0.666**	**0.704**	**0.673**	**0.713**
TransT [Chen *et al.*, 2021]	**0.670**	**0.674**	**0.620**	**0.650**	**0.646**	**0.687**	**0.643**	**0.687**	0.617	**0.654**
TrDiMP [Wang *et al.*, 2021]	0.646	0.615	0.609	0.619	0.634	0.655	0.624	0.641	**0.622**	0.639
STMTrack [Fu *et al.*, 2021]	0.640	0.624	0.571	0.582	0.606	0.631	0.601	0.631	0.582	0.626
SAOT [Zhou *et al.*, 2021]	0.617	0.580	0.584	0.586	0.611	0.623	0.596	0.606	0.541	0.554
AutoMatch [Zhang *et al.*, 2021]	0.601	0.565	0.553	0.557	0.581	0.596	0.572	0.584	0.567	0.591
Ocean [Zhang *et al.*, 2020]	0.600	0.557	0.523	0.514	0.557	0.560	0.546	0.543	0.521	0.518
DiMP-50 [Bhat *et al.*, 2019]	0.574	0.506	0.537	0.516	0.560	0.554	0.549	0.533	0.553	0.568
SiamFC++ [Xu *et al.*, 2020]	0.574	0.532	0.509	0.497	0.544	0.546	0.548	0.549	0.514	0.538
SiamGAT [Guo *et al.*, 2021]	0.571	0.509	0.512	0.485	0.540	0.530	0.538	0.527	0.500	0.498
LTMU [Dai *et al.*, 2020]	0.560	0.494	0.530	0.511	0.565	0.558	0.543	0.528	0.587	0.599

Table 8: The success performance of our method and other excellent ones on the test subsets of LaSOT with attributes of deformation, partial occlusion, scale variation, rotation, and viewpoint change, where "Succ." and "Prec." represent success and precision, respectively. The best two results are highlighted in **red** and **blue**, respectively.

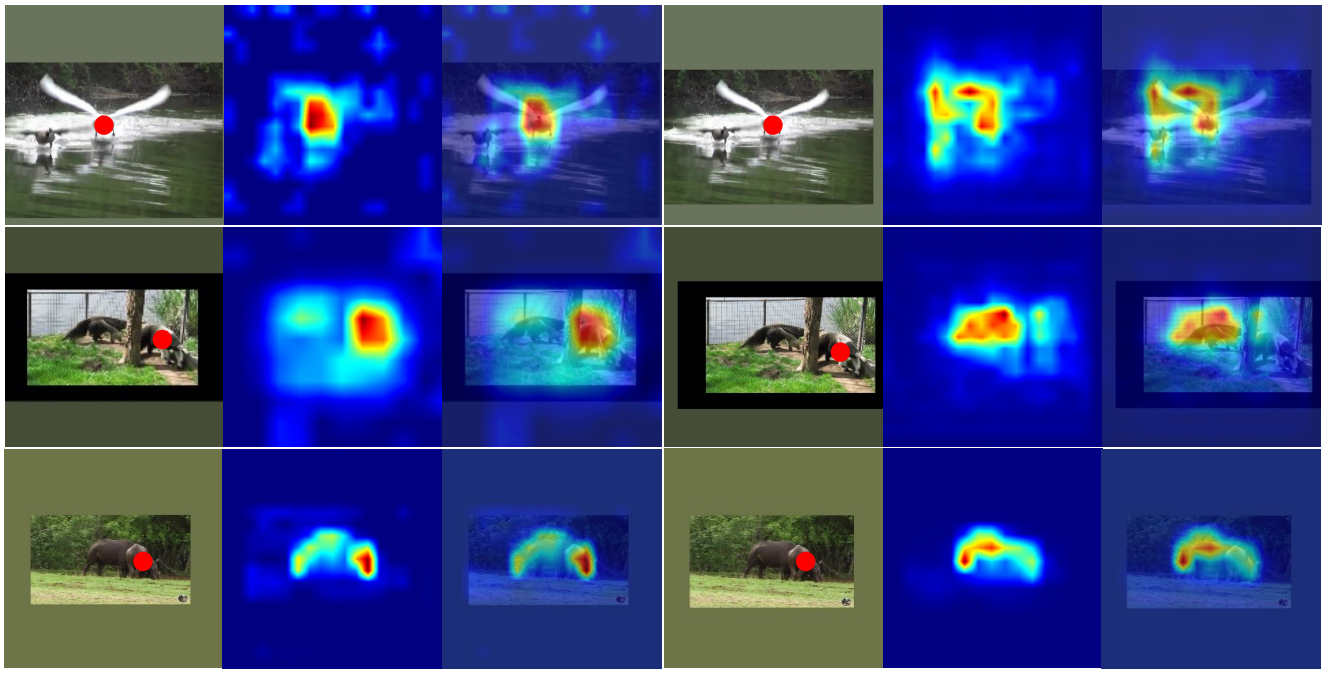

Figure 5: Visualization results of the attention maps of the search regions.

real-time speed, which demonstrates the superiority and applicability of our method. Besides, the training time of our method is only 25% of TransT. Overall, it is a new excellent baseline for further researches.

Acknowledgments

The work was supported by the National Key Research and Development Program of China under Grant 2018YFB1701600, National Natural Science Foundation of China under Grant U20B2069 and 62176017.

References

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

[Bhat *et al.*, 2019] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6182–6191, 2019.

[Bhat *et al.*, 2020] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *ECCV*, pages 205–221, 2020.

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey

Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020.

[Chen *et al.*, 2021] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, pages 8126–8135, 2021.

[Dai *et al.*, 2020] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *CVPR*, pages 6298–6307, 2020.

[Danelljan *et al.*, 2020] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020.

[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 5374–5383, 2019.

[Fu *et al.*, 2021] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. Stmtrack: Template-free visual tracking with space-time memory networks. In *CVPR*, pages 13774–13783, 2021.

[Guo *et al.*, 2021] Dongyan Guo, Yanyan Shao, Ying Cui, Zhenhua Wang, Liyan Zhang, and Chunhua Shen. Graph attention tracking. In *CVPR*, pages 9543–9552, 2021.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[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*, 2019.

[Li *et al.*, 2019] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, pages 4282–4291, 2019.

[Liao *et al.*, 2020] Bingyan Liao, Chenye Wang, Yayun Wang, Yaonong Wang, and Jun Yin. Pg-net: Pixel to global matching network for visual tracking. In *ECCV*, 2020.

[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.

[Lin *et al.*, 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[Muller *et al.*, 2018] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, pages 300–317, 2018.

[Peng *et al.*, 2021] Jinlong Peng, Zhengkai Jiang, Yueyang Gu, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, and Weiyao Lin. Siamrcr: Reciprocal classification and regression for visual object tracking. In *IJCAI*, pages 952–958, 2021.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2021] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, pages 1571–1580, 2021.

[Wu *et al.*, 2020] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. In *CVPR*, pages 10186–10195, 2020.

[Xu *et al.*, 2020] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020.

[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 10448–10457, 2021.

[Yan *et al.*, 2021b] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5289–5298, 2021.

[Yu *et al.*, 2016] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *ACM MM*, pages 516–520, 2016.

[Yu *et al.*, 2021] Bin Yu, Ming Tang, Linyu Zheng, Guibo Zhu, Jinqiao Wang, Hao Feng, Xuetao Feng, and Hanqing Lu. High-performance discriminative tracking with transformers. In *ICCV*, pages 9856–9865, 2021.

[Zhang *et al.*, 2020] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020.

[Zhang *et al.*, 2021] Zhipeng Zhang, Yihao Liu, Xiao Wang, Bing Li, and Weiming Hu. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, pages 13339–13348, 2021.

[Zhao *et al.*, 2019] Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*, 2019.

[Zhou *et al.*, 2021] Zikun Zhou, Wenjie Pei, Xin Li, Hongpeng Wang, Feng Zheng, and Zhenyu He. Saliency-associated object tracking. In *ICCV*, pages 9866–9875, 2021.