Learning Spatio-Temporal Transformer for Visual Tracking

Bin Yan[1,*], Houwen Peng[2,†], Jianlong Fu[2], Dong Wang[1,†], Huchuan Lu[1]
[1]Dalian University of Technology [2]Microsoft Research Asia

Abstract

In this paper, we present a new tracking architecture with an encoder-decoder transformer as the key component. The encoder models the global spatio-temporal feature dependencies between target objects and search regions, while the decoder learns a query embedding to predict the spatial positions of the target objects. Our method casts object tracking as a direct bounding box prediction problem, without using any proposals or predefined anchors. With the encoder-decoder transformer, the prediction of objects just uses a simple fully-convolutional network, which estimates the corners of objects directly. The whole method is end-to-end, does not need any postprocessing steps such as cosine window and bounding box smoothing, thus largely simplifying existing tracking pipelines. The proposed tracker achieves state-of-the-art performance on multiple challenging short-term and long-term benchmarks, while running at real-time speed, being 6× faster than Siam R-CNN [54]. Code and models are open-sourced at https://github.com/researchmm/Stark.

1. Introduction

Visual object tracking is a fundamental yet challenging research topic in computer vision. Over the past few years, based on convolutional neural networks, object tracking has achieved remarkable progress [28, 11, 54]. However, convolution kernels are not good at modeling long-range dependencies of image contents and features, because they only process a local neighborhood, either in space or time. Current prevailing trackers, including both the offline Siamese trackers and the online learning models, are almost all built upon convolutional operations [2, 44, 3, 54]. As a consequence, these methods only perform well on modeling local relationships of image content, but being limited to capturing long-range global interactions. Such deficiency may degrade the model capacities for dealing with the scenarios where the global contextual information is important for localization, such as the objects undergoing large-scale variations or getting in and out of views frequently.

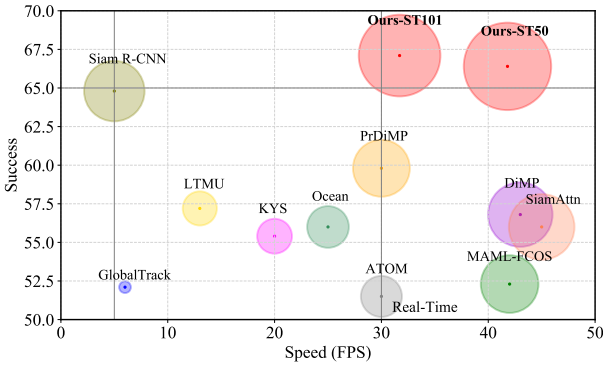

Figure 1: Comparison with state-of-the-arts on LaSOT [15]. We visualize the Success performance with respect to the Frames-Per-Seconds (*fps*) tracking speed. The circle size indicates a weighted sum of the tracker's speed (x-axis) and success score (y-axis). The larger, the better. Ours-ST101 and Ours-ST50 indicate the proposed trackers with ResNet-101 and ResNet-50 as backbones, respectively. Better viewed in color.

The problem of long-range interactions has been tackled in sequence modeling through the use of transformer [53]. Transformer has enjoyed rich success in tasks such as natural language modeling [13, 46] and speech recognition [40]. Recently, transformer has been employed in discriminative computer vision models and drawn great attention [14, 5, 41]. Inspired by the recent DEtection TRansformer (DETR) [5], we propose a new end-to-end tracking architecture with encoder-decoder transformer to boost the performance of conventional convolution models.

Both spatial and temporal information are important for object tracking. The former one contains object appearance information for target localization, while the latter one includes the state changes of objects across frames. Previous Siamese trackers [28, 59, 16, 7] only exploit the spatial information for tracking, while online methods [63, 66, 11, 3] use historical predictions for model updates. Although being successful, these methods do not explicitly model the relationship between space and time. In this work, considering the superior capacity on modeling global dependencies, we resort to transformer to integrate spatial and temporal

*Work performed when Bin Yan was an intern of MSRA.
† Corresponding authors: Houwen Peng (houwen.peng@microsoft.com), Dong Wang (wdice@dlut.edu.cn).

information for tracking, generating discriminative spatio-temporal features for object localization.

More specifically, we propose a new spatio-temporal architecture based on the encoder-decoder transformer for visual tracking. The new architecture contains three key components: an encoder, a decoder and a prediction head. The encoder accepts inputs of an initial target object, the current image, and a dynamically updated template. The self-attention modules in the encoder learn the relationship between the inputs through their feature dependencies. Since the template images are updated throughout video sequences, the encoder can capture both spatial and temporal information of the target. The decoder learns a query embedding to predict the spatial positions of the target object. A corner-based prediction head is used to estimate the bounding box of the target object in the current frame. Meanwhile, a score head is learned to control the updates of the dynamic template images.

Extensive experiments demonstrate that our method establishes new state-of-the-art performance on both short-term [20, 43] and long-term tracking benchmarks [15, 25]. For instance, our spatio-temporal transformer tracker surpasses Siam R-CNN [54] by 3.9% (AO score) and 2.3% (Success) on GOT-10K [20] and LaSOT [15], respectively. It is also worth noting that compared with previous long-term trackers [9, 54, 62], the framework of our method is much simpler. Specifically, previous methods usually consist of multiple components, such as base trackers [11, 57], target verification modules [23], and global detectors [47, 21]. In contrast, our method only has a single network learned in an end-to-end fashion. Moreover, our tracker can run at real-time speed, being $6\times$ faster than Siam R-CNN (30 v.s. 5 *fps*) on a Tesla V100 GPU, as shown in Fig. 1

Considering recent trends of over-fitting on small-scale benchmarks, we collect a new large-scale tracking benchmark called **NOTU**, integrating all sequences from NFS [24], OTB100 [58], TC128 [33], and UAV123 [42].

In summary, this work has four contributions.

- We propose a new transformer architecture dedicated to visual tracking. It is capable of capturing global feature dependencies of both spatial and temporal information in video sequences.
- The whole method is end-to-end, does not need any postprocessing steps such as cosine window and bounding box smoothing, thus largely simplifying existing tracking pipelines.
- The proposed trackers achieve state-of-the-art performance on five challenging short-term and long-term benchmarks, while running at real-time speed.
- We construct a new large-scale tracking benchmark to alleviate the over-fitting problem on previous small-scale datasets.

2. Related Work

Transformer in Language and Vision. Transformer is originally proposed by Vaswani *et al.* [53] for machine translation task, and has become a prevailing architecture in language modeling. Transformer takes a sequence as the input, scans through each element in the sequence and learns their dependencies. This feature makes transformer be intrinsically good at capturing global information in sequential data. Recently, transformer has shown their great potential in vision tasks like image classification [14], object detection [5], semantic segmentation [56], multiple object tracking [51, 41], etc. Our work is inspired by the recent work DETR [5], but has following fundamental differences. (1) The studied tasks are different. DETR is designed for object detection, while this work is for object tracking. (2) The network inputs are different. DETR takes the whole image as the input, while our input is a triplet consisting of one search region and two templates. Their features from the backbone are first flattened and concatenated then sent to the encoder. (3) The query design and training strategies are different. DETR uses 100 object queries and uses the Hungarian algorithm to match predictions with ground-truths during training. In contrast, our method only uses one query and always matches it with the ground-truth without using the Hungarian algorithm. (4) The bounding box heads are different. DETR uses a three-layer perceptron to predict boxes. Our network adopts a corner-based box head for higher-quality localization.

Moreover, TransTrack [51] and TrackFormer [41] are two most recent representative works on transformer tracking. TransTrack [51] has the following features. (1) The encoder takes the image features of both the current and the previous frame as the inputs. (2) It has two decoders, which take the learned object queries and queries from the last frame as the input respectively. With different queries, the output sequence from the encoder is transformed into detection boxes and tracking boxes respectively. (3) The predicted two groups of boxes are matched based on the IoUs using the Hungarian algorithm [27]. While Trackformer [41] has the following features. (1) It only takes the current frame features as the encoder inputs. (2) There is only one decoder, where the learned object queries and the track queries from the last frame interact with each other. (3) It associates tracks over time solely by attention operations, not relying on any additional matching such as motion or appearance modeling. In contrast, our work has the following fundamental differences with these two methods. (1) Network inputs are different. Our input is a triplet consisting of the current search region, the initial template and a dynamic template. (2) Our method captures the appearance changes of the tracked targets by updating the dynamic template, rather than updating object queries as [51, 41].

Spatio-Temporal Information Exploitation. Exploita-

tion of spatial and temporal information is a core problem in object tracking field. Existing trackers can be divided into two classes: spatial-only ones and spatio-temporal ones. Most of offline Siamese trackers [2, 29, 28, 69, 34] belong to the spatial-only ones, which consider the object tracking as a template-matching between the initial template and the current search region. To extract the relationship between the template and the search region along the spatial dimension, most trackers adopt the variants of correlation, including the naive correlation [2, 29], the depth-wise correlation [28, 69], and the point-wise correlation [34, 61]. Although achieving remarkable progress in recent years, these methods merely capture local similarity, while ignoring global information. By contrast, the self-attention mechanism in transformer can capture long-range relationship, making it suitable for pair-wise matching tasks. Compared with spatial-only trackers, spatio-temporal ones additionally exploit temporal information to improve trackers' robustness. These methods can also be divided into two classes: gradient-based and gradient-free ones. Gradient-based methods require gradient computation during inference. One of the classical works is MDNet [44], which updates domain-specific layers with gradient descent. To improve the optimization efficiency, later works [11, 3, 30, 55, 64] adopt more advanced optimization methods like Gauss-Newton method or meta-learning-based update strategies. However, many real-world devices for deploying deep learning do not support back-propagation, which restricts the application of gradient-based methods. In contrast, gradient-free methods have larger potentials in real-world applications. One class of gradient-free methods [63, 66] exploits an extra network to update the template of Siamese trackers [2, 70]. Another representative work LTMU [9] learns a meta-updater to predict whether the current state is reliable enough to be used for the update in long-term tracking. Although being effective, these methods cause the separation between space and time. In contrast, our method integrates the spatial and temporal information as a whole, simultaneously learning them with the transformer.

Tracking Pipeline and Post-processing. The tracking pipelines of previous trackers [28, 59, 69, 54] are complicated. Specifically, they first generate a large number of box proposals with confidence scores, then use various post-processing to choose the best bounding box as the tracking result. The commonly used post-processing includes cosine window, scale or aspect-ratio penalty, bounding box smoothing, tracklet-based dynamic programming, etc. Though it brings better results, post-processing causes the performance to be sensitive to hyper-parameters. There are some trackers [18, 21] attempting to simplify the tracking pipeline, but their performances still lag far behind that of state-of-the-art trackers. Recent books and surveys on

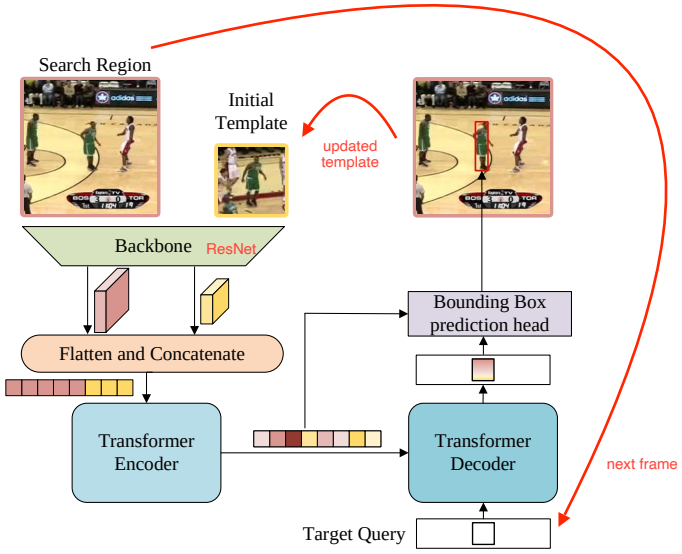

Figure 2: Framework for spatial-only tracking.

object tracking can be found in [37, 31]. This work attempts to close this gap, achieving top performance by predicting one single bounding box in each frame.

3. Method

In this section, we propose the **s**patio-**t**emporal tr**a**nsfo**r**mer network for visual trac**k**ing, called STARK. For clarity, we first introduce a simple baseline method that directly applies the original encoder-decoder transformer for tracking. The baseline method only considers spatial information and achieves impressive performance. After that, we extend the baseline to learn both spatial and temporal representations for target localization. We introduce a dynamic template and an update controller to capture the appearance changes of target objects.

3.1. A Simple Baseline Based on Transformer

We present a simple baseline framework based on visual transformer for object tracking. The network architecture is demonstrated in Fig. 2. It mainly consists of three components: a convolutional backbone, an encoder-decoder transformer, and a bounding box prediction head.

Backbone. Our method can use arbitrary convolutional networks as the backbone for feature extraction. Without loss of generality, we adopt the vanilla ResNet [17] as the backbone. More concretely, except for removing the last stage and fully-connected layers, there is no other change for the original ResNet [17]. The input of the backbone is a pair of images: a template image of the initial target object $z \in \mathbb{R}^{3\times H_z\times W_z}$ and a search region of the current frame $x \in \mathbb{R}^{3\times H_x\times W_x}$. After passing through of the backbone, the template z and the search image x are mapped to two feature maps $f_z \in \mathbb{R}^{C\times \frac{H_z}{s}\times \frac{W_z}{s}}$ and $f_x \in \mathbb{R}^{C\times \frac{H_x}{s}\times \frac{W_x}{s}}$.

Encoder. The feature maps output from the backbone require pre-processing before feeding into the encoder. To be specific, a bottleneck layer is first used to reduce the

channel number from C to d. Then the feature maps are flattened and concatenated along the spatial dimension, producing a feature sequence with length of $\frac{H_z}{s}\frac{W_z}{s} + \frac{H_x}{s}\frac{W_x}{s}$ and dimension of d, which servers as the input for the transformer encoder. The encoder consists of N encoder layers, each of which is made up of a multi-head self-attention module with a feed-forward network. Due to the permutation-invariance of the original transformer [53], we add sinusoidal positional embeddings to the input sequence. The encoder captures the feature dependencies among all elements in the sequence and reinforces the original features with global contextual information, thus allowing the model to learn discriminative features for object localization.

Decoder. The decoder takes a target query and the enhanced feature sequence from the encoder as the input. Different from DETR [5] adopting 100 object queries, we only input one single query into the decoder to predict one bounding box of the target object. Besides, since there is only one prediction, we remove the Hungarian algorithm [27] used in DETR for prediction association. Similar to the encoder, the decoder stacks M decoder layers, each of which consists of a self-attention, an encoder-decoder attention, and a feed-forward network. In the encoder-decoder attention module, the target query can attend to all positions on the template and the search region features, thus learning robust representations for the final bounding box prediction.

Head. DETR [5] adopts a three-layer perceptron to predict object box coordinates. However, as pointed by GFLoss [32], directly regressing the coordinates is equivalent to fitting a Dirac delta distribution, which fails to consider the ambiguity and uncertainty in the datasets. This representation is not flexible and not robust to challenges such as occlusion and cluttered background in object tracking. To improve the box estimation quality, we design a new prediction head through estimating the probability distribution of the box corners. As shown in Fig. 3, we first take the search region features from the encoder's output sequence, then compute the similarity between the search region features and the output embedding from the decoder. Next, the similarity scores are element-wisely multiplied with the search region features to enhance important regions and weaken the less discriminative ones. The new feature sequence is reshaped to a feature map $f \in \mathbb{R}^{d \times \frac{H_s}{s} \times \frac{W_s}{s}}$, and then fed into a simple fully-convolutional network (FCN). The FCN consists of L stacked Conv-BN-ReLU layers and outputs two probability maps $P_{tl}(x, y)$ and $P_{br}(x, y)$ for the top-left and the bottom-right corners of object bounding boxes, respectively. Finally, the predicted box coordinates $(\widehat{x_{tl}}, \widehat{y_{tl}})$ and $(\widehat{x_{br}}, \widehat{y_{br}})$ are obtained by computing the expectation of corners' probability distribution as shown in Eq. (2). Compared with DETR, our method explicitly models uncertainty in the coordinate estimation, generating more accurate and robust predictions for object tracking.

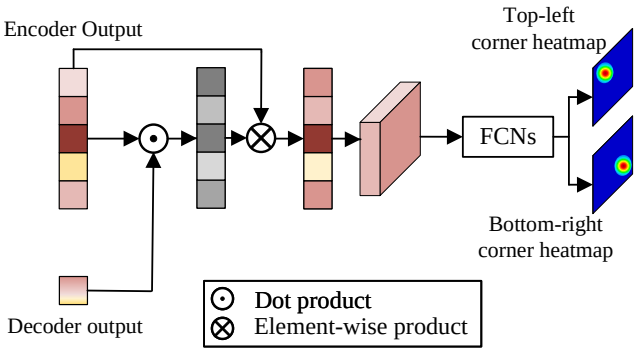

Figure 3: Architecture of the box prediction head.

Training and Inference. Our baseline tracker is trained in an end-to-end fashion with the combination of the ℓ_1 Loss and the generalized IoU loss [48] as in DETR. The loss function can be written as

$$L = \lambda_{iou} L_{iou}(b_i, \hat{b}_i) + \lambda_{L_1} L_1(b_i, \hat{b}_i). \tag{1}$$

where b_i and $\hat{b}_i$ represent the groundtruth and the predicted box respectively and $\lambda_{iou}, \lambda_{L_1} \in \mathbb{R}$ are hyperparameters. But unlike DETR, we do not use the classification loss and the Hungarian algorithm, thus further simplifying the training process. During inference, the template image together with its features from the backbone are initialized by the first frame and fixed in the subsequent frames. During tracking, in each frame, the network takes a search region from the current frame as the input, and returns the predicted box as the final result, without using any post-processing such as cosine window or bounding box smoothing.

$$\begin{aligned}(\widehat{x_{tl}}, \widehat{y_{tl}}) &= (\sum_{y=0}^{H}\sum_{x=0}^{W} x \cdot P_{tl}(x, y), \sum_{y=0}^{H}\sum_{x=0}^{W} y \cdot P_{tl}(x, y)), \\ (\widehat{x_{br}}, \widehat{y_{br}}) &= (\sum_{y=0}^{H}\sum_{x=0}^{W} x \cdot P_{br}(x, y), \sum_{y=0}^{H}\sum_{x=0}^{W} y \cdot P_{br}(x, y)),\end{aligned} \tag{2}$$

3.2. Spatio-Temporal Transformer Tracking

Since the appearance of a target object may change significantly as time proceeds, it is important to capture the latest state of the target for tracking. In this section, we demonstrate how to exploit spatial and temporal information simultaneously based on the previously introduced baseline. Three key differences are made, including the network inputs, an extra score head, and the training & inference strategy. We elaborate them one by one as below. The spatio-temporal architecture is shown in Fig. 4.

Input. Different from the baseline method which only uses the first and the current frames, the spatio-temporal method introduces a dynamically updated template sampled from intermediate frames as an additional input, as shown in Fig. 4. Beyond the spatial information from the initial

template, the dynamic template can captures the target appearance changes with time, providing additional temporal information. Similar to the baseline architecture in Sec. 3.1, feature maps of the triplet are flattened and concatenated then sent to the encoder. The encoder extracts discriminative spatio-temporal features by modeling the global relationships in both spatial and temporal dimensions.

Head. During tracking, there are some cases where the dynamic template should not be updated. For example, the cropped template is not reliable when the target has been completely occluded or has moved out of view, or when the tracker has drifted. For simplicity, we consider that the dynamic template could be updated as long as the search region contains the target. To automatically determine whether the current state is reliable, we add a simple score prediction head, which is a three-layer perceptron followed by a sigmoid activation. The current state is considered reliable if the score is higher than the threshold τ.

Training and Inference. As pointed out by recent works [8, 50], jointly learning of localization and classification may cause sub-optimal solutions for both tasks, and it is helpful to decouple localization and classification. Therefore, we divide the training process into two stages, regarding the localization as a primary task and the classification as a secondary task. To be specific, in the first stage, the whole network, except for the score head, is trained end-to-end only with the localization-related losses in Eq. 1. In this stage, we ensure all search images to contain the target objects and let the model learn the localization capacity. In the second stage, only the score head is optimized with binary cross-entropy loss defined as

$$L_{ce} = y_i log\left(P_i\right) + \left(1 - y_i\right) log\left(1 - P_i\right), \qquad (3)$$

where y_i is the groundtruth label and P_i is the predicted confidence , and all other parameters are frozen to avoid affecting the localization capacity. In this way, the final model learn both localization and classification capabilities after the two-stage training.

During inference, two templates and corresponding features are initialized in the first frame. Then a search region is cropped and fed into the network, generating one bounding box and a confidence score. The dynamic template is updated only when the update interval is reached and the confidence score is higher than the threshold τ. For efficiency, we set the update interval as T_u frames. The new template is cropped from the original image and then fed into the backbone for feature extraction.

4. Experiments

This section first presents the implementation details and the results of our STARK tracker on multiple benchmarks, with comparisons to state-of-the-art methods. Then, ablation studies are presented to analyze the effects of the key components in the proposed networks. We also report the results of other candidate frameworks and compare them with our method to demonstrate its superiority. Finally, visualization on attention maps of the encoder and the decoder are provided to understand how the transformer works.

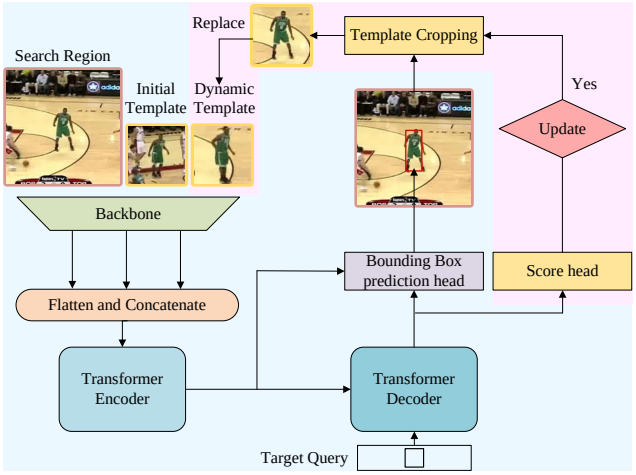

Figure 4: Framework for spatio-temporal tracking. The differences with the spatial-only architecture are in pink.

4.1. Implementation Details

Our trackers are implemented using Python 3.6 and PyTorch 1.5.1. The experiments are conducted on a server with 8 16GB Tesla V100 GPUs.

Model. We report the results of three variants of STARK: STARK-S50, STARK-ST50 and STARK-ST101. STARK-S50 only exploits spatial information and takes ResNet-50 [17] as the backbone, *i.e.*, the baseline tracker presented in Sec. 3.1. STARK-ST50 and STARK-ST101 take ResNet-50 and ResNet-101 as the backbones respectively, exploiting both spatial and temporal information, *i.e.*, the spatio-temporal tracker presented in Sec. 3.2.

The backbones are initialized with the parameters pretrained on ImageNet. The BatchNorm [22] layers are frozen during training. Backbone features are pooled from the fourth stage with a stride of 16. The transformer architecture is similar to that in DETR [5] with 6 encoder layers and 6 decoder layers, which consist of multi-head attention layers (MHA) and feed-forward networks (FFN). The MHA have 8 heads, width 256, while the FFN have hidden units of 2048. Dropout ratio of 0.1 is used. The bounding box prediction head is a lightweight FCN, consisting of 5 stacked Conv-BN-ReLU layers. The classification head is a three-layer perceptron with 256 hidden units in each layer.

Training. The training data consists of the train-splits of LaSOT [15], GOT-10K [20], COCO2017 [35], and TrackingNet [43]. As required by VOT2019 challenge, we remove 1k forbidden sequences from GOT-10K training set. The sizes of search images and templates are 320×320

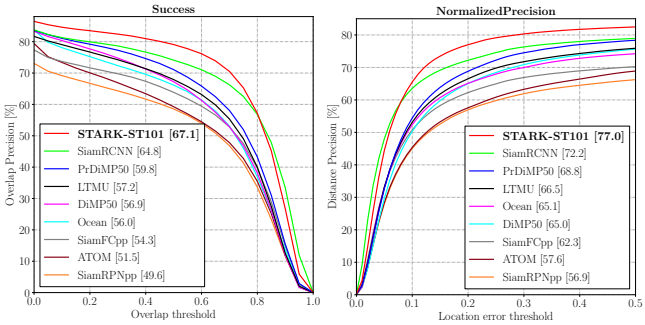

Figure 5: Comparisons on LaSOT test set [15].

pixels and 128×128 pixels respectively, corresponding to 5^2 and 2^2 times of the target box area. Data augmentations, including horizontal flip and brightness jittering, are used. The minimal training data unit for STARK-ST is one triplet, consisting of two templates and one search images. The whole training process of STARK-ST consists of two stages, which take 500 epochs for localization and 50 epochs for classification, respectively. Each epoch uses 6×10^4 triplets. The network is optimized using AdamW optimizer [36] and weight decay 10^{-4}. The loss weights λ_{L1} and λ_{iou} are set to 5 and 2 respectively. Each GPU hosts 16 triplets, hence the mini-batch size is 128 triplets. The initial learning rates of the backbone and the rest parts are 10^{-5} and 10^{-4} respectively. The learning rate drops by a factor of 10 after 400 epochs in the first stage and after 40 epochs in the second stage. The training setting of STARK-S is almost the same as that of STARK-ST, except that (1) the minimal training data unit of STARK-S is a template-search pair; (2) the training process only has the first stage.

Inference. The dynamic template update interval T_u and the confidence threshold τ are set to 200 frames and 0.5 by default. The inference pipeline only contains a forward pass and a coordinate transformation from the search region to the original image, without any extra post-processing.

4.2. Comparisons on previous benchmarks

We compare our STARK with existing state-of-the-art object trackers on three short-term benchmarks (GOT-10K, TrackingNet and VOT2020) and two long-term benchmarks (LaSOT and VOT2020-LT).

GOT-10K. GOT-10K [20] is a large-scale benchmark covering a wide range of common challenges in object tracking. GOT-10K requires trackers to only use the training set of GOT-10k for model learning. We follow this policy and retrain our models only with the GOT-10K train set. As reported in Tab. 1, with the same ResNet-50 backbone, STARK-S50 and STARK-ST50 outperform PrDiMP50 [12] by 3.8% and 4.6% AO scores, respectively. Furthermore, STARK-ST101 obtains a new state-of-the-art AO score of 68.8%, surpassing Siam R-CNN [54] by 3.9% with the same ResNet-101 backbone.

TrackingNet. TrackingNet [43] is a large-scale short-term tracking benchmark containing 511 video sequences in the test set. Tab. 2 presents that STARK-S50 and STARK-ST50 surpass PrDiMP50 [12] by 4.5% and 5.5% in AUC respectively. With a more powerful ResNet-101 backbone, STARK-ST101 achieves the best AUC of 82.0%, outperforming Siam R-CNN by 0.8%.

VOT2020. Different from previous reset-based evaluations [26], VOT2020 [25] proposes a new anchor-based evaluation protocol and uses binary segmentation masks as the groundtruth. The final metric for ranking is the Expected Average Overlap (EAO). Tab. 3 shows that STARK-S50 achieves a competitive result, which is better than DiMP [3] and UPDT [4]. After introducing temporal information, STARK-ST50 obtains an EAO of 0.308, being superior to previous bounding-box trackers. Inspired by AlphaRef [25], the winner of VOT2020 real-time challenge, we equip STARK with a refinement module in AlphaRef to generate segmentation masks. The new tracker "STARK-ST50+AR" surpasses previous SOTA trackers, like AlphaRef and OceanPlus [69], getting an EAO of 0.505.

LaSOT. LaSOT [15] is a large-scale long-term tracking benchmark, which contains 280 videos with average length of 2448 frames in the test set. STARK-S50 and STARK-ST50 achieve a gain of 6.0% and 6.6% over PrDiMP [12] respectively, using the same ResNet-50 backbone. Furthermore, STARK-ST101 obtains a success of 67.1%, which is 2.3% higher than Siam R-CNN [54], as shown in Fig. 5.

VOT2020-LT. VOT2020-LT consists of 50 long videos, in which target objects disappear and reappear frequently. Besides, trackers are required to report the confidence score of the target being present. Precision (Pr) and Recall (Re) are computed under a series of confidence thresholds. F-score, defined as $F = \frac{2PrRe}{Pr+Re}$, is used to rank different trackers. Since STARK-S does not predict this score, we do not report its result on VOT2020-LT. Tab. 4 demonstrates that STARK-ST50 and STARK-ST101 outperform all previous methods with an F-score of 70.2% and 70.1%, respectively. It is also worth noting that the framework of STARK is much simpler than that of LTMU_B, the winner of VOT2020-LT Challenge. To be specific, LTMU_B takes the combination of ATOM [11] and SiamMask [57] as the short-term tracker, MDNet [44] as the verifier, and GlobalTrack [21] as the global detector. Whereas there is only one network in STARK and the result is obtained in one forward pass without post-processing.

Speed, FLOPs and Params. As demonstrated in Tab. 6, STARK-S50 can run in real-time at more than 40 *fps*. Besides, the FLOPs and Params of STARK-S50 are $4\times$ and $2\times$ less than those of SiamRPN++. Although STARK-ST50 takes a dynamic template as the extra input and introduces an additional score head, the increases of FLOPs and Params is a little, even negligible. This shows that our

Table 1: Comparisons on GOT-10k test set [20].

	SiamFC [2]	SiamFCv2 [52]	ATOM [11]	SiamFC++ [59]	D3S [38]	DiMP50 [3]	Ocean [69]	PrDiMP50 [12]	SiamRCNN [54]	**STARK -S50**	**STARK -ST50**	**STARK -ST101**
AO(%)	34.8	37.4	55.6	59.5	59.7	61.1	61.1	63.4	64.9	**67.2**	**68.0**	**68.8**
SR0.5(%)	35.3	40.4	63.4	69.5	67.6	71.7	72.1	73.8	72.8	**76.1**	**77.7**	**78.1**
SR0.75(%)	9.8	14.4	40.2	47.9	46.2	49.2	47.3	54.3	59.7	**61.2**	**62.3**	**64.1**

Table 2: Comparisons on TrackingNet test set [43].

	DSiamRPN [70]	ATOM [11]	SiamRPN++ [28]	DiMP50 [3]	SiamAttn [65]	SiamFC++ [59]	MAML-FCOS [55]	PrDiMP50 [12]	SiamRCNN [54]	**STARK -S50**	**STARK -ST50**	**STARK -ST101**
AUC(%)	63.8	70.3	73.3	74.0	75.2	75.4	75.7	75.8	**81.2**	80.3	**81.3**	**82.0**
P_{norm}(%)	73.3	77.1	80.0	80.1	81.7	80.0	82.2	81.6	**85.4**	85.1	**86.1**	**86.9**

Table 3: Comparisons on VOT2020 [25]."+AR" means using Alpha-Refine to predict masks. The upper row summarizes trackers that only predict bounding boxes and the lower row presents trackers that report masks.

	IVT [49]	KCF [19]	SiamFC [2]	CSR-DCF [39]	ATOM [11]	DiMP [3]	UPDT [4]	DPMT	SuperDiMP [1]	**STARK -S50**	**STARK -ST50**	**STARK -ST101**
EAO(↑)	0.092	0.154	0.179	0.193	0.271	0.274	0.278	0.303	0.305	0.280	0.308	0.303
Accuracy(↑)	0.345	0.407	0.418	0.406	0.462	0.457	0.465	0.492	0.477	0.477	0.478	0.481
Robustness(↑)	0.244	0.432	0.502	0.582	0.734	0.740	0.755	0.745	0.786	0.728	0.799	0.775
	STM [45]	**SiamEM**	**SiamMask** [57]	**SiamMargin** [25]	**Ocean** [69]	**D3S** [38]	**FastOcean**	**AlphaRef** [25]	**OceanPlus** [67]	**STARK -S50+AR**	**STARK -ST50+AR**	**STARK -ST101+AR**
EAO(↑)	0.308	0.310	0.321	0.356	0.430	0.439	0.461	0.482	**0.491**	0.462	**0.505**	**0.497**
Accuracy(↑)	0.751	0.520	0.624	0.698	0.693	0.699	0.693	0.754	0.685	**0.761**	**0.759**	**0.763**
Robustness(↑)	0.574	0.743	0.648	0.640	0.754	0.769	**0.803**	0.777	**0.842**	0.749	**0.817**	0.789

method can exploit temporal information in a nearly cost-free fashion. When ResNet-101 is used as the backbone, both FLOPs and Params increase significantly but STARK-ST101 can still run at real-time speed, which is 6x faster than Siam R-CNN (5 *fps*), as shown in Fig. 1.

4.3. Comparisons on newly constructed benchmark

NOTU. In recent years, an obvious trend of over-fitting has been observed on some small-scale tracking benchmarks like OTB [58]. Performance on these datasets may not accurately reflect the tracking ability of various trackers. To address this issue, we collect a new large-scale tracking benchmark called NOTU, which contains all 401 sequences from **N**FS [24], **O**TB100 [58], **T**C-128 [33], and **U**AV-123 [42]. Tab. 5 demonstrates that the rankings of trackers on OTB100 are quite different from those on NOTU, verifying the over-fitting phenomenon we mentioned before. Besides, STARK outperforms all previous trackers consistently on NOTU, showing strong generalization ability.

4.4. Component-wise Analysis

In this section, we choose STARK-ST50 as the base model and evaluate the effects of different components in it on LaSOT [15]. For simplicity, encoder, decoder, positional encoding, corner prediction, and score head are abbreviated as enc, dec, pos, corner, and score respectively. As shown in Tab. 7 #1, when the encoder is removed, the success drops significantly by 5.3%. This illustrates that the deep interaction among features from the template and the search region plays a key role. The performance drops by 1.9% when the decoder is removed as shown in #2. This drop is less than that of removing the encoder, showing that the importance of the decoder is less than the encoder. When the positional encoding is removed, the performance only drops by 0.2% as shown in #3. Thus we conclude that the positional encoding is not a key component in our method. We also try to replace the corner head with a three-layer perceptron as in DETR [5]. #4 shows that the performance of STARK with an MLP as the box head is 2.7% lower than that of the proposed corner head. It demonstrates that the boxes predicted by the corner head are more accurate. As shown in #5, when removing the score head, the performance drops to 64.5%, which is lower than that of STARK-S50 without using temporal information. This demonstrates that improper uses of temporal information may hurt the performance and it is important to filter out unreliable templates.

4.5. Comparison with Other Frameworks

In this section, we choose the STARK-ST50 as our base model and compare it with other possible candidate frameworks. These frameworks include generating queries from the template, using the Hungarian algorithm, updating queries as in TrackFormer [41], and jointly learning localization and classification. Due to the space limitation, the figures of the detailed architectures are presented in the *supplementary material*.

Template images serve as the queries. Queries and templates have similar functions in transformer tracking. For example, both of them are expected to encode information about the target objects. From this view, a natural idea is to use template images to serve as the queries of the decoder. Specifically, first, the template and the search region features are separately fed to a weight-shared encoder then the queries generated from the template features are used to interact with the search region features in the decoder. As shown in Tab. 8, the performance of this framework is

Table 4: Comparisons on VOT-LT2020 benchmark [25].

	SPLT [62]	ltMDNet	SiamDW_LT [68]	RLT_DiMP	CLGS	Megtrack	LTMU_B [9]	LT_DSE	**STARK-ST50**	**STARK-ST101**
F-score(%)	56.5	57.4	65.6	67.0	67.4	68.7	69.1	**69.5**	**70.2**	**70.1**
Pr(%)	58.7	64.9	67.8	65.7	**73.9**	70.3	70.1	**71.5**	**71.0**	70.2
Re(%)	54.4	51.4	63.5	68.4	61.9	67.1	68.1	**67.7**	**69.5**	**70.1**

Table 5: Success score (%) comparisons on the collected large-scale benchmark NOTU and its subsets [24, 58, 33, 42].

	SiamFC [2]	RT-MDNet [23]	ECO [10]	Ocean [69]	LightTrack [60]	SiamRPN++ [28]	ATOM [11]	DiMP50 [3]	TransT [6]	**STARK-S50**	**STARK-ST50**	**STARK-ST101**
NOTU	47.2	52.9	56.7	56.7	57.4	59.8	61.5	63.4	**65.0**	64.9	**66.0**	**66.1**
NFS	37.7	43.3	52.2	49.4	49.3	57.1	58.3	61.8	**65.3**	64.3	**65.2**	**66.2**
OTB100	58.3	65.0	66.6	68.4	65.4	**68.7**	66.3	68.4	**69.5**	68.3	**68.5**	68.1
TC128	48.9	56.3	58.9	55.7	55.0	57.7	59.9	**61.2**	59.6	60.0	**62.6**	**63.1**
UAV123	46.8	52.8	53.5	57.4	62.6	59.3	63.2	64.3	68.1	**68.4**	**69.1**	**68.2**

Table 6: Comparison about the speed, FLOPs and Params.

Trackers	Speed(*fps*)	FLOPs(G)	Params(M)
STARK-S50	42.2	12.1	28.1
STARK-ST50	41.8	12.8	28.2
STARK-ST101	31.7	20.4	47.2
SiamRPN++	35.0	48.9	54.0

Table 7: Ablation for important components. Blank denotes the component is used by default, while ✗ represents the component is removed. Performance is evaluated on LaSOT.

#	Enc	Dec	Pos	Corner	Score	Success
1	✗					61.1
2		✗				64.5
3			✗			66.2
4				✗		63.7
5					✗	64.5
6						66.4

Table 8: Comparison between STARK and other candidate frameworks. Performance is evaluated on LaSOT.

	Template query	Hungarian	Update query	Loc-Cls Joint	Ours
Success	61.2	63.7	64.8	62.5	66.4

61.2%, which is 5.2% lower than that of our design. We conjecture that the underlying reason is that compared with our method, this design lacks the information flow from the template to the search region, thus weakening the discriminative power of the search region features.

Using the Hungarian algorithm [5]. We also try to use K queries, predicting K boxes with confidence scores. K is equal to 10 in the experiments. The groundtruth is dynamically matched with these queries during the training using the Hungarian algorithm. We observe that this training strategy leads to the "Matthew effect". There are only one or two queries having the ability to predict high-quality boxes. If they are not selected during inference, the predicted box may become unreliable. As shown in Tab. 8, this strategy performs inferior to our method by 2.7%.

Updating the query embedding. Different from STARK exploiting temporal information by introducing an extra dynamic template, TrackFormer [41] encodes temporal information by updating the query embedding. Following this idea, we extend the STARK-S50 to a new temporal tracker by updating the target query. Tab. 8 shows that this design achieves a success of 64.8%, which is 1.6% lower than that of STARK-ST50. The underlying reason might be that the extra information brought by an updatable query embedding is much less than that by an extra template.

Jointly learning of localization and classification. As mentioned in Sec 3.2, localization is regarded as the primary task and is trained in the first stage. While classification is trained in the second stage as the secondary task. We also make an experiment to jointly learn localization and classification in one stage. As shown in Tab. 8, this strategy leads to a sub-optimal result, which is 3.9% lower than that of STARK. Two potential reasons are: (1) Optimization of the score head interferes with the training of the box head, leading to inaccurate box predictions. (2) Training of these two tasks requires different data. To be specific, the localization task expects that all search regions contain tracked targets to provide strong supervision. By contrast, the classification task expects a balanced distribution, half of search regions containing the targets, while the remaining half not.

5. Conclusion

This paper proposes a new transformer-based tracking framework, which can capture the long-range dependency in both spatial and temporal dimensions. Besides, the proposed STARK tracker gets rid of hyper-parameters sensitive post-processing, leading to a simple inference pipeline. Extensive experiments show that the STARK trackers perform much better than previous methods on both existing short-term and long-term benchmarks and newly constructed NOTU benchmark, while running in real-time. We expect this work can attract more attention on transformer architecture for visual tracking.

Acknowledgement.

We would like to thank the reviewers for their insightful comments. Lu and Wang are supported in part by National Natural Science Foundation of China under grant No. U1903215, 61725202, 61829102, 62022021 and Dalian Innovation leader's support Plan under Grant No. 2018RD07.

References

[1] https://github.com/visionml/pytracking. 7

[2] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H S Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, 2016. 1, 3, 7, 8

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 1, 3, 6, 7, 8

[4] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 6, 7

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2, 4, 5, 7, 8

[6] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 8

[7] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *CVPR*, 2020. 1

[8] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *ECCV*, 2018. 5

[9] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *CVPR*, 2020. 2, 3, 8

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017. 8

[11] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *CVPR*, 2019. 1, 2, 3, 6, 7, 8

[12] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, 2020. 6, 7

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, 2019. 1

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2

[15] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019. 1, 2, 5, 6, 7

[16] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese fully convolutional classification and regression for visual tracking. In *CVPR*, 2020. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 5

[18] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 3

[19] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. In *ICVS*, 2008. 7

[20] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2019. 2, 5, 6, 7

[21] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Globaltrack: A simple and strong baseline for long-term tracking. In *AAAI*, 2020. 2, 3, 6

[22] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[23] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time MDNet. In *ECCV*, 2018. 2, 8

[24] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017. 2, 7, 8

[25] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking vot2020 challenge results. In *ECCVW*, 2020. 2, 6, 7, 8

[26] Matej Kristan, Jiri Matas, Ales Leonardis, et al. The seventh visual object tracking VOT2019 challenge results. In *ICCVW*, 2019. 6

[27] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 2, 4

[28] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 1, 3, 7, 8

[29] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 3

[30] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. GradNet: Gradient-guided network for visual object tracking. In *ICCV*, 2019. 3

[31] Peixia Li, Dong Wang, Lijun Wang, and Huchuan Lu. Deep visual tracking: Review and experimental comparison. *PR*, 2018. 3

[32] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In *NeurIPS*, 2020. 4

[33] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. *TIP*, 2015. 2, 7, 8

[34] Bingyan Liao, Chenye Wang, Yayun Wang, Yaonong Wang, and Jun Yin. PG-Net: Pixel to global matching network for visual tracking. 3

[35] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[37] Huchuan Lu and Dong Wang. *Online visual tracking*. Springer, 2019. 3

[38] Alan Lukezic, Jiri Matas, and Matej Kristan. D3S-a discriminative single shot segmentation tracker. In *CVPR*, 2020. 7

[39] Alan Lukezic, Tomas Vojir, Luka ˇCehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017. 7

[40] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Rwth asr systems for librispeech: Hybrid vs attention–w/o data augmentation. *arXiv preprint arXiv:1905.03072*, 2019. 1

[41] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021. 1, 2, 7, 8

[42] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 2, 7, 8

[43] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 2, 5, 6, 7

[44] Hyeonseob Nam and Bohyung Han. Learning multi–domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 3, 6

[45] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019. 7

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R–CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[48] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 4

[49] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *ijcv*, 2008. 7

[50] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *CVPR*, 2020. 5

[51] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. TransTrack: Multiple-object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020. 2

[52] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 7

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 2, 4

[54] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by re-detection. In *CVPR*, 2020. 1, 2, 3, 6, 7

[55] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *CVPR*, 2020. 3, 7

[56] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020. 2

[57] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 2, 6, 7

[58] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang. Object tracking benchmark. *TPAMI*, 2015. 2, 7, 8

[59] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 2020. 1, 3, 7

[60] 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 *ICCV*, 2021. 8

[61] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. *arXiv preprint arXiv:2012.06815*, 2020. 3

[62] Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. 'Skimming-Perusal' Tracking: A framework for real-time and robust long-term tracking. In *ICCV*, 2019. 2, 8

[63] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018. 1, 3

[64] Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B Chan. ROAM: Recurrently optimizing tracking model. In *CVPR*, 2020. 3

[65] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *CVPR*, 2020. 7

[66] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *ICCV*, 2019. 1, 3

[67] Zhipeng Zhang, Bing Li, Weiming Hu, and Houweng Peng. Towards accurate pixel-wise object tracking by attention retrieval. *arXiv preprint arXiv:2008.02745*, 2020. 7

[68] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, 2019. 8

[69] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020. 3, 6, 7, 8

[70] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 3, 7