

# SwinTrack: A Simple and Strong Baseline for Transformer Tracking

Liting Lin<sup>1,2,\*</sup>   Heng Fan<sup>3,\*</sup>   Yong Xu<sup>1,2</sup>   Haibin Ling<sup>4</sup>

<sup>1</sup>School of Computer Science & Engineering, South China Univ. of Tech., Guangzhou, China

<sup>2</sup>Peng Cheng Laboratory, Shenzhen, China

<sup>3</sup>Department of Computer Science and Engineering, University of North Texas, Denton, TX USA

<sup>4</sup>Department of Computer Science, Stony Brook University, Stony Brook, NY USA

l.l.t@mail.scut.edu.cn   heng.fan@unt.edu   yxu@scut.edu.cn   hling@cs.stonybrook.edu

## Abstract

Transformer has recently demonstrated clear potential in improving visual tracking algorithms. Nevertheless, existing transformer-based trackers mostly use Transformer to fuse and enhance the features generated by convolutional neural networks (CNNs). By contrast, in this paper, we propose a fully attentional-based Transformer tracking algorithm, Swin-Transformer Tracker (SwinTrack). SwinTrack uses Transformer for both feature extraction and feature fusion, allowing full interactions between the target object and the search region for tracking. To further improve performance, we investigate comprehensively different strategies for feature fusion, position encoding, and training loss. All these efforts make SwinTrack a simple yet solid baseline. In our thorough experiments, SwinTrack sets a new record with 0.702 SUC on LaSOT, surpassing STARK [44] by 3.1% while still running at 45 FPS. Besides, it achieves state-of-the-art performances with 0.476 SUC, 0.840 SUC and 0.694 AO on other challenging LaSOT<sub>ext</sub>, TrackingNet, and GOT-10k datasets. Our implementation and trained models are available at <https://github.com/LitingLin/SwinTrack>.

## 1. Introduction

Recently, Transformer has made significant progress on vision tasks. Attempts to introduce the Transformer architecture into the vision community can be broadly classified into two types: some studies regard the Transformer structure as a powerful complement to CNNs, employing a hybrid architecture which combines the attention mechanisms with convolutional networks, attempting to exploit the advantages of both; the other studies, encouraged by Transformer’s remarkable success in NLP tasks, devote to explorer a fully attentional model, believe that Transformer

\*Equal contributions.

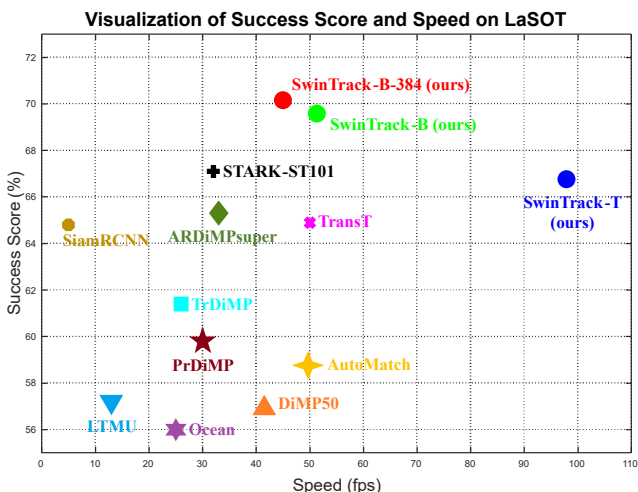


Figure 1. Comparison with state-of-the-art trackers on LaSOT [16] using success (SUC) score and speed. Our SwinTrack-T with light architecture reaches a state-of-the-art performance with 0.667 SUC score and meanwhile runs the fastest at around 100 fps. Using larger model, SwinTrack-B-384 set a breakthrough new record with 0.702 SUC score and still runs efficiently at approximately 45 fps. Best viewed in color and by zooming in.

will defeat CNN structures in the near future, and attention mechanisms will be served as the fundamental building blocks of the next generation. Several hybrid architectures, such as [10] [5], have rapidly reached state-of-the-art in a variety of tasks, indicating the great potential of the Transformer. In contrast, the fully attentional model did not go so well at the first time. The Vision Transformer [13] (ViT, the first fully attentional model in vision tasks) and many of its successors [37] were inferior to convnets in terms of performance, until the appearance of the Swin-Transformer [30].

Swin-Transformer employs a hierarchical window attention-based architecture to address two major challenges in the Transformer architecture: the variety of vi-

sual elements in scale and the high computational complexity on high-resolution images. Unlike the ViT family using a fixed-size feature map, Swin-Transformer builds the feature map by gradually merging neighbor patches from large to small. With hierarchical feature maps, traditional multi-scale prediction techniques can be used to overcome the scaling problem. Besides, Swin-Transformer introduces a non-overlapping window partition operation. Self-attention computing is limited within the window. As a result, the computational complexity is greatly reduced. Furthermore, the partition windows are shifted periodically to bridge the windows in preceding layers.

The advantages of Transformer are widely acknowledged to be due to two factors [39]: The Transformer is a sequence-to-sequence model, which makes it easier to combine multi-modal data, thus providing more flexibility in network architecture design; The capability of long-range modeling from the attention mechanism unleash the limitation of the traditional CNN-based or RNN-based model.

Visual object tracking is a challenging research topic with a long history. Many issues are still not well addressed, including relocation after occlusion or being out of vision, discrimination between similar objects, etc. [7] and [44] are the most advanced trackers in the visual object tracking task. They both use a hybrid architecture, with ResNet serving as the backbone and Transformer serving as the encoder and decoder networks, as previously summarized. We believe that by fully utilizing the power of fully attentional model and the Swin-Transformer backbone, we can significantly boost up the tracker’s performance to a new level.

Through the insight of the nature of the attention mechanism and a bunch of thorough experiments, we designed a powerful yet efficient fully attentional tracker - SwinTrack. SwinTrack suppresses the SOTA [44] [32] trackers on the challenging long-term dataset LaSOT by 3.1%, while still having an FPS at 45. We also provide a lighter version of SwinTrack, which provides a SOTA performance at 97 FPS.

The key designs of SwinTrack includes:

- Swin-Transformer as the backbone network;
- Proper choices between various candidate network structures for different part of the tracker;
- Introduce untied positional encoding to provide an accurate positional encoding for concatenation-based feature fusion;
- Introduce IoU-Aware Classification Score to the classification prediction branch, to select an more accurate bounding box prediction.

We believe that SwinTrack has fully revealed the great potential of the Transformer network. We’d like to propose the SwinTrack model as a new baseline network for future research.

## 2. Related Work

### 2.1. Transformer in Vision Tasks

Transformer was first proposed by [39], applied in the task of machine text translation. Due to significantly more parallelization and promising performance, Transformer rapidly replaced the LSTM model and soon achieved complete dominance in NLP tasks.

Starting from 2020, Transformer has been vastly introduced to the vision community. DETR [5] attracted a lot of attention. By modeling the object detection as a direct set prediction problem, DETR removes most hand-crafted processes and reaches a state-of-the-art comparable performance without domain knowledge. Later, the advancing model of DETR [48] and many other transformer-based models were proposed to the image and video tasks.

The large-scale pre-trained models in NLP tasks have made a great success, such as the well-known BERT [12] and the GPT family [34]. With the attempts to replicate the success, the Vision Transformer(ViT) [13] was proposed. ViT splits the image into multiple fixed-size patches as the token, with a linear projection and a proper positional encoding. The image tokens are then fed into the standard Transformer encoder. With the success of the first applicable convolution-free network architecture and a vision of a shared pre-trained backbone network for CV and NLP tasks, a family of ViT variants was proposed [38] [28] [6] [47].

In standard ViTs, the number of tokens is fixed across the layers. To control the computation complexity and open the access to the multi-scale architecture in various vision tasks, multi-scale Vision Transformers with window-based attention were proposed, like [42] [8] [30]. Swin Transformer [30] may be the most famous one since it reached state-of-the-art in multiple tasks when it was first released.

### 2.2. Siamese Tracking

By offline learning a generic matching function from a large set of sequences, tracking is to search for a region that is the most similar to the target template. The Siamese methods formulate object tracking as a matching problem. Especially, the work of [1] introduces a fully convolutional Siamese network for tracking and shows a good balance off between accuracy and speed. In order to improve [1] in dealing with scale variation, the method of [26] incorporates the region proposal network into Siamese network and proposes the anchor-based tracker, achieving higher accuracy with faster speed. Later, numerous extensions have been presented to improve [26], including deeper backbone network [25], multi-stage architecture [16, 17], anchor-free Siamese trackers [46].

### 2.3. Transformer in Visual Tracking

Several Transformer based trackers have been proposed. [7] [41] [18] are the very first works that introduce the Transformer architecture to the visual object tracking. [7] propose the ECA and CFA modules. The modules replace the traditional correlation operation with cross attention. [41] improves the Siamese matching and DiMP based tracking frameworks by Transformer enhanced template features and search features. [44] explores the Spatio-temporal Transformer by integrating the model updating operations into a Transformer module.

## 3. Swin Transformer Tracking

### 3.1. Overview

Our tracker is based on the Siamese network architecture [4], as shown in 2. Four main components comprise our fully attentional tracker: the Swin-Transformer backbone, the attentional encoder-decoder network, positional encoding, and the head network. During tracking, the backbone network extracts the features of the template image patch and the search region image patch separately with shared weights (for simplification, we call them the template image and the search image for convenience, respectively), the encoder network fuse the feature tokens from the *template image* and the *search image* by concatenation, and enhances the concatenated tokens layer-by-layer by attention mechanism, positional encoding helps the model to distinguish the tokens from the different source and the different position, the decoder network generates the final feature map of the search image and feeds it to the head network to obtain the IoU-Aware classification response map and bounding box estimation map. We will discuss the details of each component in the following sections.

### 3.2. Transformer-based Feature Extraction

The deep convolutional neural network has significantly improved the performance of trackers. Along with the advancement of trackers, the backbone network has evolved twice: AlexNet [23] and ResNet [19]. Swin-Transformer [30], in comparison to ResNet, can give a more compact feature representation and richer semantic information to assist succeeding networks in better localizing the target objects, which we will demonstrate in the ablation study experimentally.

Our tracker follows the scheme of classic Siamese tracker [1], which requires a pair of image patches as the input, one is the template image patch  $z \in \mathbb{R}^{H_z \times W_z \times 3}$ , the other one is the search region image patch  $x \in \mathbb{R}^{H_x \times W_x \times 3}$  (for simplification, we call them the *template image* and the *search image* for convenience respectively).

We denote the feature tokens from the template image as  $z \in \mathbb{R}^{\frac{H_z}{s} \times \frac{W_z}{s} \times C}$ , the feature tokens from the search image

as  $x \in \mathbb{R}^{\frac{H_x}{s} \times \frac{W_x}{s} \times C}$ ,  $s$  is the stride of the backbone network. Since there is no dimension projection in our model,  $C$  is also the hidden dimension of the whole model.

### 3.3. Transformer-based Feature Fusion

**Encoder.** The encoder is composed of a sequence of blocks where each block contains a multi-head self-attention (MSA) module and a feed forward network (FFN). FFN contains a two-layers multi-layer perceptron (MLP), GELU activation layer is inserted after the first layer's output. Layer normalization (LN) is always performed before every module (MSA and FFN). Residual connection is applied on MSA and FFN modules.

Before the feature tokens are fed into the encoder, the tokens from the template image and the search image are concatenated along spatial dimensions to generate a union representation  $U$ . For each block, the MSA module computes self-attention over the union representation, FFN refines the feature tokens generated by MSA. When the tokens are getting out of the encoder, a de-concatenation operation is performed to recover the template image feature tokens and the search image feature tokens.

The full process can be expressed as:

$$\begin{aligned} U^1 &= \text{Concat}(z^1, x^1) \\ &\dots \\ U^l &= U^l + \text{MSA}(\text{LN}(U^l)) \\ U^{l+1} &= U^l + \text{FFN}(\text{LN}(U^l)) \\ &\dots \\ z^L, x^L &= \text{DeConcat}(U^L), \end{aligned} \tag{1}$$

where  $l$  denotes the  $l$ -th layer and  $L$  the number of blocks.

**Why concatenated attention?** To simplify the description, we call the method described above *concatenation-based fusion*. To fuse and process features from multiple branches, it is intuitive to perform self-attention on the feature tokens in each branch separately to complete the feature extraction step and then compute cross-attention across feature tokens from different branches to complete the feature fusion step. We call this method *cross-attention-based fusion*. Considering that the Transformer is a sequence-to-sequence model, the Transformer can naturally accept multi-modal data as input. In comparison to *cross-attention-based fusion*, *concatenation-based fusion* can save computation operations through operation combination and reduce model parameters through weight sharing. From this perspective, *concatenation-based fusion* implicitly implements the **Siamese network architecture**. To ensure that the attention mechanism is aware of which branch the token currently being processed belongs to and its location within the branch, we must carefully design the model's positional encoding solution.

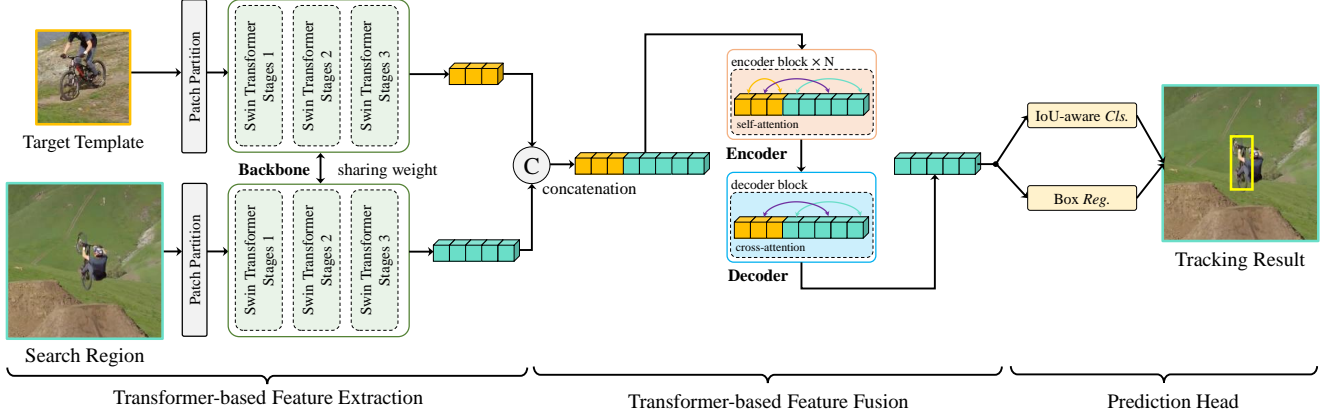


Figure 2. Architecture of SwinTrack, which consists of three parts including Swin Transformer-based feature extraction, Transformer-based feature fusion and prediction head. Our SwinTrack is a simple and neat tracking framework without complex designs such as multi-scale feature and temporal update, yet demonstrating state-of-the-art performance.

**Why not window-based self/cross-attention?** Since we select stage 3 of the Swin-Transformer as the output, the number of tokens is small enough that the FLOPs between window-based attention and full attention are pretty similar. Furthermore, some tokens may need to go through multiple-layer before computing a correlation, which is too expensive for our tracker.

**Decoder.** The decoder consists of a multi-head cross-attention ( $MCA$ ) module and a feed forward network (FFN). The decoder takes the outputs from the encoder as input, generating the final feature map  $\mathbf{x} \in \mathbb{R}^{\frac{H_x}{s} \times \frac{W_x}{s} \times C}$  of the search image by computing cross-attention over  $\mathbf{x}_L$  and  $\text{Concat}(\mathbf{z}_L, \mathbf{x}_L)$ . The decoder is very similar to a layer in the encoder, except that the correlation from the template image tokens to the search image tokens is dropped, since we do not need to update the features from the template image in the last layer. We can formulate the process in the decoder by:

$$\begin{aligned} U^D &= \text{Concat}(\mathbf{z}^L, \mathbf{x}^L) \\ \mathbf{x}^{L'} &= \mathbf{x}^L + MCA(\text{LN}(\mathbf{x}^L), \text{LN}(U^D)) \\ \mathbf{x} &= \mathbf{x}^{L'} + \text{FFN}(\text{LN}(\mathbf{x}^{L'})). \end{aligned} \quad (2)$$

**Why not an end-to-end architecture?** Many Transformer-based models have an end-to-end architecture, which means that the model predicts the task’s objective directly, without any post-processing steps. However, in our tests, an end-to-end model is still not applicable for our task. In our experiment, when applying a transformer-style decoder, like the one in [5] to directly predict the bounding box of the target object, the model takes a much longer time to converge and has an inferior tracking performance. The decoder we’ve chosen can help to improve the performance in three folds: By predicting a response map, we can offload

the candidate selection task to the manually designed post-processing step. By dense prediction, we can feed a richer supervision signal to the model, which can fasten the training process. And also, we can use more domain knowledge to help improve the tracking performance, like applying a Hanning penalty window on the response map to introduce the smooth movement assumption.

**Why not a target query-based decoder?** We also find that the traditional transformer decoder is hard to recover 2D positional information in our experiment.

### 3.4. Positional Encoding

Transformer requires a positional encoding to identify the position of the current processing token [39]. Through a series of comparison experiments, we choose *untied positional encoding*, which is proposed in TUPE [21], as the positional encoding solution of our tracker. In addition, we generalize the *untied positional encoding* to arbitrary dimensions to fit with other components in our tracker.

The original transformer [39] proposes a absolute positional encoding method to represent the position: a fixed or learnable vector  $p_i$  is assigned to each position  $i$ . Starting from the basic attention module, we have:

$$\text{Atten}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}V\right), \quad (3)$$

where  $Q, K, V$  are the *query* vector, *key* vector and *value* vector, which are the parameters of the attention function,  $d_k$  is the dimension of *key*. Introducing the linear projection matrix and multi-head attention to the attention module (3), we get the multi-head variant defined in [39]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O, \quad (4)$$

where  $\text{head}_i = \text{Atten}(QW_i^Q, KW_i^K, VW_i^V)$ ,  $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ,  $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ,  $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ ,  $W_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$  and  $h$  is the number of heads. For simplicity, as in [21], we assume that  $d_k = d_v = d_{\text{model}}$ , and use the single-head version of self-attention module. Denoting the input sequence as  $x = x_1, x_2, \dots, x_n$ , where  $n$  is the length of sequence,  $x_i$  is the  $i$ -th token in the input data. Denoting the output sequence as  $z = (z_1, z_2, \dots, z_n)$ . Self-attention module can be rewritten as

$$z_i = \sum_{j=1}^n \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^n \exp(\alpha_{ij'})} (x_j W^V), \quad (5)$$

$$\text{where } \alpha_{ij} = \frac{1}{\sqrt{d}} (x_i W^Q)(x_j W^K)^T. \quad (6)$$

Obviously, the self-attention module is permutation-invariance. Thus it can not “understand” the order of input tokens.

**Untied absolute positional encoding.** By adding a learnable positional encoding [39] to the single-head self-attention module, we can obtain the following equation:

$$\begin{aligned} \alpha_{ij}^{\text{Abs}} &= \frac{((w_i + p_i)W^Q)((w_j + p_j)W^K)^T}{\sqrt{d}} \\ &= \frac{(w_i W^Q)(w_j W^K)^T}{\sqrt{d}} + \frac{(w_i W^Q)(p_j W^K)^T}{\sqrt{d}} \\ &\quad + \frac{(p_i W^Q)(w_j W^K)^T}{\sqrt{d}} + \frac{(p_i W^Q)(p_j W^K)^T}{\sqrt{d}}. \end{aligned} \quad (7)$$

The equation (7) is expanded into four terms: token-to-token, token-to-position, position-to-token, position-to-position. [21] discuss the problems exists in the equation and proposes the *untied absolute positional encoding*, which unties the correlation between tokens and positions by removing the token-position correlation terms in equation (7), and using an isolated pair of projection matrices  $U^Q$  and  $U^K$  to perform linear transformation upon positional embedding vector. The following is the new formula for obtaining  $\alpha_{ij}$  using the *untied absolute positional encoding* in the  $l$ -th layer:

$$\begin{aligned} \alpha_{ij} &= \frac{1}{\sqrt{2d}} (x_i^l W^{Q,l})(x_j^l W^{K,l})^T \\ &\quad + \frac{1}{\sqrt{2d}} (p_i U^Q)(p_j U^K)^T. \end{aligned} \quad (8)$$

where  $p_i$  and  $p_j$  is the positional embedding at position  $i$  and  $j$  respectively,  $U^Q \in \mathbb{R}^{d \times d}$  and  $U^K \in \mathbb{R}^{d \times d}$  are learnable projection matrices for the positional embedding vector. When extending to the multi-head version, the positional embedding  $p_i$  is shared across different heads, while  $U^Q$  and  $U^K$  are different for each head.

**Relative positional bias.** According to [36], relative positional encoding is a necessary supplement to absolute positional encoding. In [21], a relative positional encoding is

applied by adding a relative positional bias to equation (8):

$$\begin{aligned} \alpha_{ij} &= \frac{1}{\sqrt{2d}} (x_i^l W^{Q,l})(x_j^l W^{K,l})^T \\ &\quad + \frac{1}{\sqrt{2d}} (p_i U^Q)(p_j U^K)^T + b_{j-i}, \end{aligned} \quad (9)$$

where for each  $j - i$ ,  $b_{j-i}$  is a learnable scalar. The *relative positional bias* is also shared across layers. When extending to the multi-head version,  $b_{j-i}$  is different for each head.

**Generalize to multiple dimensions.** Before working with our tracker’s encoder and decoder network, we need to extend the *untied positional encoding* to a multidimensional version. One straightforward method is allocating a positional embedding matrix for every dimension and summing up all embedding vectors from different dimensions at the corresponding index to represent the final embedding vector. Together with *relative positional bias*, for an  $n$ -dimensional case, we have:

$$\begin{aligned} \alpha_{\underbrace{ij \dots mn}_{n}} &= \frac{1}{\sqrt{2d}} (\underbrace{x_{ij \dots mn}}_n W^Q)(\underbrace{x_{mn \dots ij}}_n W^K)^T \\ &\quad + \frac{1}{\sqrt{2d}} [\underbrace{(p_i^1 + p_j^2 + \dots)}_n U^Q][\underbrace{(p_m^1 + p_n^2 + \dots)}_n U^K]^T \\ &\quad + \underbrace{b_{m-i, n-j, \dots}}_n. \end{aligned} \quad (10)$$

**Generalize to concatenation-based fusion.** In order to work with *concatenation-based fusion*, the *untied absolute positional encoding* is also concatenated to match the real position, the indexing tuple of *relative positional bias* now appends with a pair of indices to reflect the origination of *query* and *key* involved currently.

Taking  $l$ -th layer in the encoder as the example:

$$\begin{aligned} \alpha_{ij, mn, g, h} &= \frac{1}{\sqrt{2d}} (x_{ij, g}^l W^{Q, l})(x_{mn, h}^l W^{K, l})^T \\ &\quad + \frac{1}{\sqrt{2d}} [(p_{i, g}^1 + p_{j, g}^2) U_g^Q][(p_{m, h}^1 + p_{n, h}^2) U_h^K]^T \\ &\quad + b_{m-i, n-j, g, h}, \end{aligned} \quad (11)$$

where  $g$  and  $h$  are the index of the origination of *query* and *key* respectively, for instance, 1 for the tokens from the template image, 2 for the tokens from the search image. The form in the decoder is similar, except that  $g$  is fixed. In our implementation, the parameters of *untied positional encoding* are shared inside the encoder and the decoder, respectively.

### 3.5. Head and Training Loss

**Head.** The head network is split into two branches: classification branch and bounding box regression branch. Each

branch is a three-layer perceptron. One is in charge of foreground-background classification. The other one is in charge of bounding box regression. They are both receiving the feature map  $x \in \mathbb{R}^{(H_x \times W_x) \times C}$  from the decoder, predict the classification response map  $r_{cls} \in \mathbb{R}^{(H_x \times W_x) \times 1}$  and bounding box regression map  $r_{reg} \in \mathbb{R}^{(H_x \times W_x) \times 4}$ , respectively.

**Classification loss.** In classification branch, we employ the *IoU-aware classification score* as the training target and the *varifocal loss* [45] as the training loss function.

IoU-aware design has been very popular recently, but most works task IoU prediction branch as an auxiliary branch to assist classification branch or bounding box regression branch. To remove the gap between different prediction branches, [45] and [27] replace the classification target from ground-truth value, i.e., 1 for positive samples, 0 for negative samples, to the IoU between the predicted bounding box and the ground-truth one, which is named the *IoU-aware classification score* (IACS). IACS can help the model select a more accurate bounding box from the candidate pool. Along with the IACS, the varifocal loss was proposed in [45] to help the IACS approach outperform other IoU-aware designs. The *varifocal loss* has the following form:

$$\text{VFL}(p, q) = \begin{cases} -q(q \log(p) + (1-q) \log(1-p)) & q > 0 \\ -\alpha p^\gamma \log(1-p) & q = 0, \end{cases} \quad (12)$$

where  $p$  is the predicted IACS and  $q$  is the target score. For positive samples, i.e., the foreground points,  $q$  is the IoU between the predicted bounding box and the ground-truth bounding box. For negative samples,  $q$  is 0. Then the classification loss can be formulated as:

$$\mathbb{L}_{cls} = \text{VFL}(p, \text{IoU}(b, \hat{b})), \quad (13)$$

where  $b$  denotes the predicted bounding box,  $\hat{b}$  denotes the ground-truth bounding box.

**Regression loss.** For bounding box regression, we employ the generalized IoU loss [35]. The regression loss function can be formulated as:

$$\mathbb{L}_{reg} = \sum_j \mathbb{I}_{\{q>0\}} [p \mathbb{L}_{\text{GIoU}}(b_j, \hat{b})]. \quad (14)$$

The GIoU loss is weighted by  $p$  to emphasize the high classification score samples. The training signals from the negative samples are ignored.

## 4. Experiments

### 4.1. Implementation

**Model.** We show three variants of SwinTrack with different configurations as follows:

- **SwinTrack-T.**

Backbone: Swin Transformer-Tiny [30];

Template size:  $[112 \times 112]$ ; Search region size:  $[224 \times 224]$ ;  $C = 384$ ;  $N = 4$ ;

- **SwinTrack-B.**

Backbone: Swin Transformer-Base [30];

Template size:  $[112 \times 112]$ ; Search region size:  $[224 \times 224]$ ;  $C = 512$ ;  $N = 8$ ;

- **SwinTrack-B-384.**

Backbone: Swin Transformer-Base [30];

Template size:  $[192 \times 192]$ ; Search region size:  $[384 \times 384]$ ;  $C = 512$ ;  $N = 8$ ;

where  $C$  and  $N$  represent the channel number of the hidden layers in the first stage of Swin Transformer and the number of encoder blocks in feature fusion, respectively. In all variants, we use the output after the third stage of Swin Transformer for feature extraction. Thus, the backbone stride  $s$  is set to 16.

**Training.** We train SwinTrack using training splits of LaSOT [15], TrackingNet [33], GOT-10k [20] (1,000 videos are removed for fair comparisons with other trackers [22, 44]) and COCO 2017 [29]. Besides, we also report the performance of SwinTrack-T and SwinTrack-B with GOT-10k training split only to follow the protocol described in [20].

The model is optimized with AdamW [31]. The learning rate of the backbone is set to  $5e-5$ , and the weight decay is  $1e-4$ . We adopt gradient clipping to prevent very large gradients from misleading the optimization process. We train the network on 8 NVIDIA V100 GPUs for 300 epochs with 131,072 samples per epoch. The learning rate is dropped by a factor of 10 after 210 epochs. To stabilize training process, a warmup strategy is utilized. DropPath [24] is applied in the latter half of the optimization process.

**Inference.** We follow the common procedures for Siamese network-based tracking [1]. The template image is cropped from the first frame of the video sequence. The target object is in the center of the image with a background area factor of 2. The search region is cropped from the current tracking frame, and the image center is the target center position predicted in previous frame. The background area factor for the search region is 4.

Our SwinTrack takes the template image and search region as inputs and output classification map  $r_{cls}$  and regression maps  $r_{reg}$ . To utilize positional prior in tracking, we apply hanning window penalty on  $r_{cls}$ , and the final classification map  $r'_{cls}$  is obtained via  $r'_{cls} = (1-\gamma) \times r_{cls} + \gamma \times h$ , where  $\gamma$  is the weight parameter and  $h$  is a Hanning window with the same size as  $r_{cls}$ . The target position is determined by the largest value in  $r'_{cls}$  and scale is estimated based on the corresponding regression results in  $r_{reg}$ .

Table 1. Ablation studies on SwinTrack-T.

Modification	LaSOT			LaSOT <sub>ext</sub>			TrackingNet			GOT-10k			Speed (fps)	MACs (G)	Params (M)
	SUC (%)	PRE (%)	NPPE (%)	SUC (%)	PRE (%)	NPPE (%)	SUC (%)	PRE (%)	NPPE (%)	mAO (%)	mSR <sub>50</sub> (%)	mSR <sub>75</sub> (%)			
(SwinTrack-T)	66.7	70.6	75.8	46.9	52.9	57.6	80.8	77.9	85.5	70.9	81.2	64.9	98	6.4	22.7
ResNet-50	64.2	67.4	72.9	41.8	46.3	51.3	79.5	77.1	84.2	68.2	77.7	61.2	121	21.4	20.0
Cross Fusion	66.6	69.9	75.6	45.4	50.8	55.8	80.2	77.7	85.3	69.3	79.4	64.2	72	7.0	34.6
Target query	66.6	70.1	75.5	43.2	46.4	52.5	79.6	76.9	84.6	69.0	78.9	64.3	91	5.9	25.3
Sine enc.	65.7	68.8	74.4	45.0	50.0	55.4	80.0	77.3	85.2	70.0	80.0	64.4	103	6.2	21.6
BCE loss	66.2	69.5	75.8	46.7	52.5	57.1	79.4	76.9	84.8	68.2	78.3	63.1	98	6.4	22.7
Weak aug.	61.6	63.4	68.4	38.7	40.5	45.8	78.6	75.7	83.2	67.9	76.8	62.5	98	6.4	22.7
No hann.	65.7	69.4	74.6	46.0	51.5	56.6	80.0	77.3	85.0	69.6	79.3	65.1	98	6.4	22.7

## 4.2. Ablations Study and Analysis

We conduct ablations to study different factors in SwinTrack. To save training time, we perform ablation studies on SwinTrack-T.

**Comparison with ResNet backbone.** We compare our Transformer-based backbone with commonly adopted ResNet [19] for tracking. As shown in Table 1, using ResNet-50 lead to a huge drop in each dataset.

**Feature fusion.** According to Table 1, compared with the *concatenation-based fusion*, the *cross attention-based fusion* not only perform worse than the *concatenation-based fusion*, but also has a larger number of parameters.

**Decoder.** We employ a transformer-style decoder, which is introduced in DETR, to our SwinTrack. By computing cross attention with the pre-trained target query tokens, the model can find the potential target objects in the feature. Ideally, it can generate the bounding box of the target object directly without any post-processing steps. However, our empirical results in Table 1 show the tracker with a transformer-style decoder has poor performance in most datasets.

**Position encoding.** We compare the adopted united positional encoding and the original since encoding in Transformer. As shown in Table 1, SwinTrack-T with united positional encoding achieves better accuracy with around 1% improvements over SwinTrack-T with sine encoding on different datasets, while still runs fast in around 98 *fps*.

**Loss function.** From Table 1, we observe that SwinTrack-T with varifocal loss significantly outperforms the one with binary entropy loss (BCS) without loss of efficiency.

**Positional Augmentations.** Inspired by [25], we set up an experiment to discover the impact of positional augmentation during image pre-processing. The "Weak aug." row in Table 1 shows the dataset evaluation results of deducing random scale and random translation during the search image generation in the training phase. The success score evaluated in LaSOT drops by 5.1%, in LaSOT<sub>ext</sub> even 8.2%, compared with our fine-tuned hyper-parameters.

**Post processing.** By removing the hanning penalty win-

dow in the post-processing, as shown in Table 1, the performance is significantly dropped. This suggests that even with a strong backbone network, hanning penalty window is still functional.

## 4.3. State-of-the-art Comparison

We compare our SwinTrack with state-of-the-art trackers on four benchmarks including LaSOT [15], LaSOT<sub>ext</sub> [14], TrackingNet [33] and GOT-10k [20].

**LaSOT.** LaSOT [15] is a large-scale benchmark containing 280 test sequences. Table 2 shows the results of SwinTrack and comparisons with state-of-the-art trackers. From Table 2, we can see that our SwinTrack-T with light architecture reaches a SOTA performance with 0.667 SUC, 0.706 PRE, and 0.758 NPPE scores, which is competitive compared with other Transformer-based trackers, including STARK-ST101 (0.671 SUC score) and TransT (0.649 SUC and 0.60 PRE scores), and other trackers which utilize complicated designs for tracking, like KeepTrack (0.671 SUC and 0.702 PRE scores) and SiamR-CNN (0.648 SUC score). When using larger backbone and input size, our strongest variant SwinTrack-B-384 gives a breakthrough new record with 0.702 and 0.753 of the SUC score and the PRE score, respectively.

**LaSOT<sub>ext</sub>.** The recently proposed LaSOT<sub>ext</sub> [14] is an extension of LaSOT by adding 150 extra videos from 15 new categories. These new sequences are challenging because there are many similar distractors that cause difficulties for tracking. KeepTrack designs a complex association technique to deal with the distractors and achieves a promising 0.482 SUC score. Compared with complicated KeepTrack, SwinTrack-T is simple and neat, yet shows comparable performance with 0.469 SUC score. In addition, due to complicated design, KeepTrack runs at less than 20 *fps*, while SwinTrack-T runs in 98 *fps*, 5× faster than KeepTrack. When using a larger model, SwinTrack-B shows the best performance with 0.476 SUC score and 0.582 NPPE score in the variants. The SUC score of our SwinTrack-B is still lower than KeepTrack, mainly due to the lack of the utilization of temporal information, while the NPPE score

Table 2. Performance comparison on LaSOT [15].

Tracker	SUC (%)	PRE (%)	NPRE (%)
SiamPRN++ [25]	49.6	-	56.9
DiMP [2]	56.9	53.4	65.0
Ocean [46]	56.0	56.6	65.1
SiamR-CNN [40]	64.8	-	72.2
TrSiam [41]	62.4	60.0	-
TrDiMP [41]	63.9	61.4	-
STMTrack [18]	60.6	63.3	69.3
TransT [7]	64.9	69.0	73.8
STARK-ST50 [44]	66.4	-	-
STARK-ST101 [44]	67.1	-	77.0
KeepTrack [32]	67.1	70.2	77.2
SwinTrack-T	66.7	70.6	75.8
SwinTrack-B	69.6	74.1	<b>78.6</b>
SwinTrack-B-384	<b>70.2</b>	<b>75.3</b>	78.4

Table 3. Performance comparison on LaSOT<sub>ext</sub> [14].

Tracker	SUC (%)	PRE (%)	NPRE (%)
C-RPN [16]	27.5	32.0	34.4
DiMP [2]	39.2	45.1	47.5
LTMU [9]	41.4	47.3	49.9
SuperDiMP [11]	43.7	-	52.7
KeepTrack [32]	<b>48.2</b>	-	58.0
SwinTrack-T	46.9	52.9	57.6
SwinTrack-B	47.6	<b>54.1</b>	<b>58.2</b>
SwinTrack-B-384	47.5	53.3	57.7

Table 4. Performance comparison on TrackingNet [33].

Tracker	SUC (%)	PRE (%)	NPRE (%)
PrDiMP [11]	75.8	70.4	81.6
SiamFC++ [43]	75.4	70.5	80.0
KYS [3]	74.0	68.8	80.0
SiamR-CNN [40]	81.2	80.0	85.4
TrSiam [41]	78.1	72.7	82.9
TrDiMP [41]	78.4	73.1	83.3
STMTrack [18]	80.3	76.7	85.1
TransT [7]	81.4	80.3	86.7
STARK-ST50 [44]	81.3	-	86.1
STARK-ST101 [44]	82.0	-	86.9
SwinTrack-T	80.8	77.9	85.5
SwinTrack-B	82.5	80.4	87.0
SwinTrack-B-384	<b>84.0</b>	<b>83.2</b>	<b>88.2</b>

is better than KeepTrack.

**TrackingNet.** We evaluate our trackers on the test set of TrackingNet [33]. The results are shown in Table 4. From Table 4, we observe that our SwinTrack-T achieves comparable result of 0.808 SUC score. When using larger model

Table 5. Performance comparison on GOT-10k [20].

Tracker	mAO (%)	mSR <sub>50</sub> (%)	mSR <sub>75</sub> (%)
SiamPRN++ [25]	51.7	61.6	32.5
DiMP [2]	61.1	71.7	49.2
Ocean [46]	61.1	72.1	47.3
SiamR-CNN [40]	64.9	72.8	59.7
TrSiam [41]	66.0	76.6	57.1
TrDiMP [41]	67.1	77.7	58.3
STMTrack [18]	64.2	73.7	57.5
TransT [7]	67.1	76.8	60.9
STARK-ST50 [44]	68.0	77.7	62.3
STARK-ST101 [44]	68.8	<b>78.1</b>	64.1
SwinTrack-T	69.0	<b>78.1</b>	62.1
SwinTrack-B	<b>69.4</b>	78.0	<b>64.3</b>

Table 6. Comparison on average running speed and # parameters.

Tracker	Speed (fps)	Params (M)
SiamRPN++ [25]	35	54
TrSiam [41]	35	-
TrDiMP [41]	26	-
TransT [7]	50	-
STARK-ST50 [44]	42	24
STARK-ST101 [44]	32	42
SwinTrack-T	98	23
SwinTrack-B	52	91
SwinTrack-B-384	45	91

and input size, our SwinTrack-B-384 obtains the best performance with 0.840 SUC score, better than STARK-ST101 with 0.820 SUC score and TransT with 0.813 SUC score.

**GOT-10k.** GOT-10k [20] provides 180 sequences for testing and it requires trackers to be trained using GOT-10k train split only. The results and comparisons are displayed in Table 4. From Table 4, we see that SwinTrack-B achieves the best mAO of 0.694, outperforming other Transformer-based counterparts including STARK-ST101 (0.688 mAO), TransT (0.671 mAO), TrDiMP (0.671 mAP) and TrSiam (0.660 mAO).

**Efficiency comparison.** In addition to accuracy comparison, we also report the comparisons of SwinTrack with others trackers on efficiency and complexity. As shown in Table 6, our SwinTrack-T with a small model runs the fastest with a speed of 98 fps. Especially, compared with existing state-of-the-art STARK-ST101 and STARK-ST50 with 32 fps and 42 fps, SwinTrack-T is 3× and 2× faster. Despite using a larger model, our SwinTrack-B-384 is still faster than STARK-ST101 and STARK-ST50.

## 5. Conclusion

In this paper, we propose a strong baseline SwinTrack for Transformer tracker. SwinTrack consists of a Swin-



Transformer-based backbone network, a concatenation-based fusion encoder, a general positional encoding solution for any combination of attention operation, and incorporating with some popular training tricks. By achieving state-of-the-art results on multiple challenging benchmarks, we expect SwinTrack can serve as a solid baseline for further research on Transformer-based tracking.

## References

- [1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.
- [2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019.
- [3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *ECCV*, 2020.
- [4] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.
- [5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- [6] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv*, 2021.
- [7] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021.
- [8] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv*, 2021.
- [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.
- [10] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *arXiv*, 2021.
- [11] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, 2020.
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018.
- [13] 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. In *ICLR*, 2021.
- [14] Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Mingzhen Huang, Juehuan Liu, Yong Xu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *IJCV*, 129(2):439–461, 2021.
- [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.
- [16] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019.
- [17] Heng Fan and Haibin Ling. Cract: Cascaded regression-align-classification for robust visual tracking. In *IROS*, 2021.
- [18] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. Stmtrack: Template-free visual tracking with space-time memory networks. In *CVPR*, 2021.
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [20] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *PAMI*, 43(5):1562–1577, 2021.
- [21] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021.
- [22] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *PAMI*, 38(11):2137–2155, Nov 2016.
- [23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.
- [24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2016.
- [25] B Li, W Wu, Q Wang, F Zhang, J Xing, and J SiamRPN+ Yan. Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.
- [26] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.
- [27] Xiang Li, Wenhui 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.
- [28] Yawei Li, Kai Zhang, Jiezhong Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv*, 2021.
- [29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [30] 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. *International Conference on Computer Vision (ICCV)*, 2021.
- [31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [32] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *ICCV*, 2021.

- [33] 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.
- [34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
- [35] Hamid Rezaatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. 2019.
- [36] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv*, 2018.
- [37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.
- [38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [40] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, 2020.
- [41] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, 2021.
- [42] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv*, 2021.
- [43] 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.
- [44] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021.
- [45] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sünderhauf. Varifocalnet: An iou-aware dense object detector. In *CVPR*, 2021.
- [46] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020.
- [47] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv*, 2021.
- [48] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.