arXiv:2112.00995v2 [cs.CV] 8 Dec 2021

Success Score (%)

72

70

68

66

64

62

60

58

**Ocean**

56

**SwinTrack:** **A** **Simple** **and** **Strong** **Baseline** **for** **Transformer** **Tracking**

Liting Lin1,2,\*

Heng Fan3,\*

Yong Xu1,2

Haibin Ling4

1 School of Computer Science & Engineering, South China Univ. of Tech., Guangzhou, China

2 Peng Cheng Laboratory, Shenzhen, China

3 Department of Computer Science and Engineering, University of North Texas, Denton, TX USA

4 Department of Computer Science, Stony Brook University, Stony Brook, NY USA

l.lt@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,* *ex-* *isting* *transformer-based* *trackers* *mostly* *use* *Transformer* *tofuse* *and* *enhance* *thefeatures* *generated* *by* *convolutional* *neural* *networks* *(CNNs).* *By* *contrast,* *in* *this* *paper,* *we* *pro-* *pose* *a* *fully* *attentional-based* *Transformer* *tracking* *algo-*

*rithm,* *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* *per-* *formance,* *we* *investigate* *comprehensively* *different* *strate-* *giesforfeaturefusion,* *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]*](#_bookmark1) *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*ext*,* *TrackingNet,* *and* *GOT-10k* *datasets.* *Our* *implementation* *and* *trained* *models* *are* *available* *at* [*https://github.com/LitingLin/SwinTrack*](https://github.com/LitingLin/SwinTrack)*.*

**1.** **Introduction**

Recently, Transformer has made signiﬁcant progress on vision tasks. Attempts to introduce the Transformer archi- tecture into the vision community can be broadly classi- ﬁed into two types: some studies regard the Transformer structure as a powerful complement to CNNs, employing a hybrid architecture which combines the attention mech- anisms 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.

**Visualization** **of** **Success** **Score** **and** **Speed** **on** **LaSOT**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | **Sw** | **inTrac** | **k** **-B-384** | **(ours)** |  |  |  |
|  |  |  |  |  | **Swin** | **Track** **-** | **B** **(ours** | **)** |  |  |
|  |  |  | **STA** | **RK** **-ST** | **101** |  |  |  |  |  |
| **Siam** | **RCNN** | **ARD** | **iMPsu** | **per** | **Trans** | **T** |  |  | **SwinTr**  **(ou** | **rs)**  **ack** **-T** |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  | **T** | **rDiMP** |  |  |  |  |  |  |  |
|  |  | **PrD** | **iMP** |  | **Auto** | **Match** |  |  |  |  |
| **L** | **TMU** |  |  | **DiM** | **P50** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

0 10 20 30 40 50 60 70 80 90 100 110

Speed (fps)

Figure 1. Comparison with state-of-the-art trackers on LaSOT [[16](#_bookmark2)] 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*. Us- ing 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 atten- tion mechanisms will be served as the fundamental building blocks of the next generation. Several hybrid architectures, such as [[10](#_bookmark3)] [[5](#_bookmark4)], have rapidly reached state-of-the-art in a variety of tasks, indicating the great potential of the Trans- former. In contrast, the fully attentional model did not go so well at the ﬁrst time. The Vision Transformer [[13](#_bookmark5)] (ViT, the ﬁrst fully attentional model in vision tasks) and many of its successors [[37](#_bookmark6)] were inferior to convnets in terms of perfor- mance, until the appearance of the Swin-Transformer [[30](#_bookmark7)].

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

sual elements in scale and the high computational complex- ity on high-resolution images. Unlike the ViT family using a ﬁxed-size feature map, Swin-Transformer builds the fea- ture 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 acknowl- edged to be due to two factors [[39](#_bookmark8)]: The Transformer is a sequence-to-sequence model, which makes it easier to com- bine multi-modal data, thus providing more ﬂexibility in network architecture design; The capability of long-range modeling from the attention mechanism unleash the limita-

tion 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](#_bookmark9)] and [[44](#_bookmark1)] are the most advanced trackers in the visual object track- ing task. They both use a hybrid architecture, with ResNet serving as the backbone and Transformer serving as the en- coder 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 signiﬁ- cantly boost up the tracker’s performance to a new level.

Through the insight of the nature of the attention mech- anism and a bunch of thorough experiments, we designed a powerful yet efﬁcient fully attentional tracker - SwinTrack. SwinTrack suppresses the SOTA [[44](#_bookmark1)] [[32](#_bookmark10)] 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 clas- siﬁcation 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](#_bookmark8)], 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 com- plete dominance in NLP tasks.

Starting from 2020, Transformer has been vastly intro- duced to the vision community. DETR [[5](#_bookmark4)] 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 perfor- mance without domain knowledge. Later, the advancing model of DETR [[48](#_bookmark11)] 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](#_bookmark12)] and the GPT family [[34](#_bookmark13)]. With the attempts to replicate the success, the Vision Transformer(ViT) [[13](#_bookmark5)] was proposed. ViT splits the image into multiple ﬁxed-size patches as the token, with a linear projection and a proper positional en- coding. The image tokens are then fed into the standard Transformer encoder. With the success of the ﬁrst applica- ble 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](#_bookmark14)] [[28](#_bookmark15)] [[6](#_bookmark16)] [[47](#_bookmark17)].

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](#_bookmark18)] [[8](#_bookmark19)] [[30](#_bookmark7)]. Swin Trans-

former [[30](#_bookmark7)] 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](#_bookmark20)] introduces a fully convolutional Siamese network for tracking and shows a good balance off between accuracy and speed. In order to improve [[1](#_bookmark20)] in dealing with scale variation, the method of [[26](#_bookmark21)] incor- porates the region proposal network into Siamese network and proposes the anchor-based tracker, achieving higher ac- curacy with faster speed. Later, numerous extensions have been presented to improve [[26](#_bookmark21)], including deeper backbone network [[25](#_bookmark22)], multi-stage architecture [[16](#_bookmark2),[17](#_bookmark23)], anchor-free Siamese trackers [[46](#_bookmark24)].

**2.3.** **Transformer** **in** **Visual** **Tracking**

Several Transformer based trackers have been proposed. [[7](#_bookmark9)] [[41](#_bookmark25)] [[18](#_bookmark26)] are the very ﬁrst works that introduce the Transformer architecture to the visual object tracking. [[7](#_bookmark9)] propose the ECA and CFA modules. The modules re- place the traditional correlation operation with cross at- tention. [[41](#_bookmark25)] improves the Siamese matching and DiMP based tracking frameworks by Transformer enhanced tem- plate features and search features. [[44](#_bookmark1)] 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 architec-

ture [[4](#_bookmark27)], as shown in[2](#_bookmark28). Four main components comprise our fully attentional tracker: the Swin-Transformer backbone, the attentional encoder-decoder network, positional encod- ing, 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 im- age 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 mecha- nism, 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 compo- nent 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 ad- vancement of trackers, the backbone network has evolved twice: AlexNet [[23](#_bookmark29)] and ResNet [[19](#_bookmark30)]. Swin-Transformer [[30](#_bookmark7)], in comparison to ResNet, can give a more compact feature representation and richer semantic information to assist succeeding networks in better localizing the target ob- jects, which we will demonstrate in the ablation study ex- perimentally.

Our tracker follows the scheme of classic Siamese

tracker [[1](#_bookmark20)], which requires a pair of image patches as the input, one is the template image patch z ∈ RH2 ×W2 ×3, the other one is the search region image patch x ∈ RHz ×Wz ×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 ∈ R   ×C, the feature tokens from the search image

as x ∈ R  ~~z~~ ~~z~~ ×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 com- putes self-attention over the union representation, FFN re- ﬁnes 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:

U 1 = Concat(z1 , x1 )

. . .

Ul′ = Ul + MSA(LN(Ul ))

(1)

Ul+1 = Ul′ + FFN(LN(Ul′ ))

. . .

zL , xL = DeConcat(UL ),

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 extrac- tion step and then compute cross-attention across feature tokens from different branches to complete the feature fu- sion step. We call this method *cross-attention-basedfusion*. 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* *fu-* *sion*, *concatenation-basedfusion* can save computation op- erations through operation combination and reduce model parameters through weight sharing. From this perspec- tive, *concatenation-based* *fusion* implicitly implements the

**Siamese** **network** **architecture**. To ensure that the atten-

tion mechanism is aware of which branch the token cur- rently being processed belongs to and its location within the branch, we must carefully design the model’s positional encoding solution.



Swin Transformer

Stages 1

Swin Transformer

Stages 3









Swin Transformer

Stages 2

Swin Transformer

Stages 1

Swin Transformer

Stages 3









Swin Transformer

Stages 2

|  |
| --- |
| IoU-aware *Cls.* |

|  |
| --- |
| Box *Reg.* |

Patch Partition

|  |
| --- |
|  |

Target Template

Patch Partition

|  |
| --- |
|  |

encoder block × N

self-attention

|  |
| --- |
|  |
|  |



sharing weight

|  |  |
| --- | --- |
| |  | | --- | |  | |

**Backbone**

concatenation

C

|  |  |
| --- | --- |
| |  | | --- | |  | |

**Encoder**

|  |
| --- |
| decoder block |
| cross-attention |

**Decoder**

|  |
| --- |
|  |

Tracking Result

Search Region



Transformer-based Feature Extraction Transformer-based Feature Fusion Prediction Head

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 expen- sive 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 **x** ∈ R  ~~z~~ ×  ~~z~~ ×C of the search image by computing cross-attention over xL and Concat(zL , xL ). 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 im- age in the last layer. We can formulate the process in the decoder by:

UD = Concat(zL , xL )

xL′ = xL + MCA(LN(xL ), LN(UD )) (2)

**x** = xL′ + FFN(LN(xL′ )).

**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 ex- periment, when applying a transformer-style decoder, like the one in [[5](#_bookmark4)] to directly predict the bounding box of the target object, the model takes a much longer time to con- verge and has an inferior tracking performance. The de- coder 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 train- ing 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](#_bookmark8)]. Through a series of comparison experiments, we choose *untied* *po-* *sitional* *encoding*, which is proposed in TUPE [[21](#_bookmark31)], as the positional encoding solution of our tracker. In addition, we generalize the *untied* *positional* *encoding* to arbitrary di- mensions to ﬁt with other components in our tracker.

The original transformer [[39](#_bookmark8)] proposes a absolute posi- tional encoding method to represent the position: a ﬁxed or learnable vector pi is assigned to each position i. Starting from the basic attention module, we have:

Atten(Q, K, V) = softmax/  V、, (3)

where Q,K,V are the query vector, key vector and value vector, which are the parameters of the attention function, dk is the dimension of key . Introducing the linear projec- tion matrix and multi-head attention to the attention module ([3](#_bookmark32)), we get the multi-head variant deﬁned in [[39](#_bookmark8)]:

MultiHead(Q, K, V) = Concat(head1 , ..., headh )WO ,

(4)

where headi = Atten(QWiQ , KWiK , VWiV ), WiQ ∈

Rd←à↘-d ×dk , WiK ∈ Rd←à↘-d ×dk , WiV ∈ Rd←à↘-d ×d0 , WiO ∈ Rhd0 ×d←à↘-d and h is the number of heads. For simplicity, as in [[21](#_bookmark31)], we assume that dk = dv = dmodel , and use the single-head version of self-attention module. Denoting the input sequence as x = x1 , x2 , . . . , xn , where n is the length of sequence, xi is the i-th token in the input data. Denoting the output sequence as z = (z1 , z2 , . . . , zn ). Self-attention module can be rewritten as

|  |  |
| --- | --- |
| zi = n  (xj WV ),  j=1 j ′ =1 exp(αij ′ )  where αij = ~~d~~ (xi WQ )(xj WK )T . | (5)  (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 learn- able positional encoding [[39](#_bookmark8)] to the single-head self- attention module, we can obtain the following equation:

αbs = ~~((wi + pi )WQ )~~~~j + pj )WK )T~~ 

+ ed + ed .

The equation ([7](#_bookmark33)) is expanded into four terms: token- to-token, token-to-position, position-to-token, position-to- position. [[21](#_bookmark31)] discuss the problems exists in the equa- tion and proposes the *untied* *absolute* *positional* *encoding*, which unties the correlation between tokens and positions by removing the token-position correlation terms in equa- tion ([7](#_bookmark33)), and using an isolated pair of projection matrices UQ and UK to perform linear transformation upon posi- tional embedding vector. The following is the new formula for obtaining αij using the *untied* *absolute* *positional* *en-* *coding* in the l-th layer:

αij = ~~1e21d~~ (x W Q,l )(xWK,l )T

(8)

+ e2d (pi UQ )(pj UK )T .

where pi and pj is the positional embedding at position i and j respectively, UQ ∈ Rd×d and UK ∈ Rd×d are learn- able projection matrices for the positional embedding vec- tor. When extending to the multi-head version, the posi- tional embedding pi is shared across different heads, while UQ and UK are different for each head.

**Relative** **positional** **bias.** According to [[36](#_bookmark35)], relative posi- tional encoding is a necessary supplement to absolute po- sitional encoding. In [[21](#_bookmark31)], a relative positional encoding is

applied by adding a relative positional bias to equation ([8](#_bookmark34)):

αij = ~~1e21d~~ (x W Q,l )(xWK,l )T

(9)

+ e2d (pi UQ )(pj UK )T + bj −i ,

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

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

α..`,m↗ ..` = ~~e~~ (x..`WQ )(xm↗ ..`WK )T

≥ ≥ ≥ ≥

+ ~~e~~ [(↗p + + . . .)` UQ][(↗p +  + . . .)` UK ]T

≥ ≥

+ bm - i, n - j, . . . .

↗ `

←←

≥

(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:

αij,mn,g,h = ~~e~~ (xj,g WQ,l )(xn,h WK,l )T + ~~e~~ [(p,g + p,g )U][(p,h + p,h )U]T (11)

+ bm −i,n −j,g,h ,

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, re- spectively.

**3.5.** **Head** **and** **Training** **Loss**

**Head.** The head network is split into two branches: classi- ﬁcation 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 receiv- ing the feature map x ∈ R(Hz ×Wz ) ×C from the decoder, predict the classiﬁcation response map rcls ∈ R(Hz ×Wz ) × 1 and bounding box regression map rreg ∈ R(Hz ×Wz ) ×4, re- spectively.

**Classiﬁcation** **loss.** In classiﬁcation branch, we employ the *IoU-aware* *classiﬁcation* *score* as the training target and the *varifocal* *loss* [[45](#_bookmark36)] 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 re- gression branch. To remove the gap between different pre- diction branches, [[45](#_bookmark36)] and [[27](#_bookmark37)] replace the classiﬁcation tar- get 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 candi- date pool. Along with the IACS, the varifocal loss was pro- posed in [[45](#_bookmark36)] to help the IACS approach outperform other IoU-aware designs. The *varifocal* *loss* has the following form:

VFL(p, q)= 

q > 0

q = 0, (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 clas- siﬁcation loss can be formulated as:

Lcls = VFL(p, IoU(b, )), (13)

where b denotes the predicted bounding box,  denotes the ground-truth bounding box.

**Regression** **loss.** For bounding box regression, we employ the generalized IoU loss [[35](#_bookmark38)]. The regression loss function can be formulated as:

✶ {q>0} [pLGIoU (bj , )].

Lreg =

(14)

j

The GIoU loss is weighted by p to emphasize the high clas- siﬁcation score samples. The training signals from the neg- ative 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](#_bookmark7)];

Template size: [112 × 112]; Search region size: [224 × 224]; C = 384; N = 4;

• **SwinTrack-B**.

Backbone: Swin Transformer-Base [[30](#_bookmark7)];

Template size: [112 × 112]; Search region size: [224 × 224]; C = 512; N = 8;

• **SwinTrack-B-384**.

Backbone: Swin Transformer-Base [[30](#_bookmark7)];

Template size: [192 × 192]; Search region size: [384 × 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 vari- ants, we use the output after the third stage of Swin Trans- former for feature extraction. Thus, the backbone stride s is

set to 16.

**Training.** We train SwinTrack using training splits of La- SOT [[15](#_bookmark39)], TrackingNet [[33](#_bookmark40)], GOT-10k [[20](#_bookmark41)] (1,000 videos are removed for fair comparisons with other trackers [[22](#_bookmark42), [44](#_bookmark1)]) and COCO 2017 [[29](#_bookmark43)]. Besides, we also report the per-

formance of SwinTrack-T and SwinTrack-B with GOT-10k

training split only to follow the protocol described in [[20](#_bookmark41)].

The model is optimized with AdamW [[31](#_bookmark44)]. 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 gra- dients 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 pro- cess, a warmup strategy is utilized. DropPath [[24](#_bookmark45)] is applied in the latter half of the optimization process.

**Inference.** We follow the common procedures for Siamese network-based tracking [[1](#_bookmark20)]. 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 track- ing 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 re- gion as inputs and output classiﬁcation map rcls and regres- sion maps rreg . To utilize positional prior in tracking, we apply hanning window penalty on rcls , and the ﬁnal classiﬁ- cation map rls is obtained via rls = (1 - γ) × rcls +γ × h, where γ is the weight parameter and h is a Hanning window with the same size as rcls . The target position is determined by the largest value in rls and scale is estimated based on the corresponding regression results in rreg .

Table 1. Ablation studies on SwinTrack-T.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modiﬁcation | LaSOT | | | LaSOText | | | TrackingNet | | | GOT-10k | | | Speed (fps) | MACs  (G) | Params  (M) |
| NPRE  PRE  (%)  SUC  (%)  (%) | | | NPRE  PRE  (%)  SUC  (%)  (%) | | | NPRE  PRE  (%)  SUC  (%)  (%) | | | mSR75 (%)  mAO  (%)  mSR50 (%) | | |
| (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 Swin- Track. 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](#_bookmark30)] for tracking. As shown in Table [1](#_bookmark46), using ResNet-50 lead to a huge drop in each dataset.

**Feature** **fusion.** According to Table [1](#_bookmark46), compared with the *concatenation-based* *fusion*, the *cross* *attention-based* *fu-* *sion* 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 comput- ing cross attention with the pre-trained target query tokens, the model can ﬁnd the potential target objects in the fea- ture. Ideally, it can generate the bounding box of the tar- get object directly without any post-processing steps. How- ever, our empirical results in Table [1](#_bookmark46) show the tracker with a transformer-style decoder has poor performance in most datasets.

**Position** **encoding.** We compare the adopted united posi- tional encoding and the original since encoding in Trans- former. As shown in Table [1](#_bookmark46), SwinTrack-T with united po- sitional encoding achieves better accuracy with around 1% improvements over SwinTrack-T with sine encoding on dif- ferent datasets, while still runs fast in around 98*fps*.

**Loss** **function.** From Table [1](#_bookmark46), 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](#_bookmark22)], we set up an experiment to discover the impact of positional augmenta- tion during image pre-processing. The ”Weak aug.” row in Table [1](#_bookmark46) shows the dataset evaluation results of deducing random scale and random translation during the search im- age generation in the training phase. The success score eval- uated in LaSOT drops by 5.1%, in LaSOText 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](#_bookmark46), the per- formance 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](#_bookmark39)], LaSOText [[14](#_bookmark47)], TrackingNet [[33](#_bookmark40)] and GOT-10k [[20](#_bookmark41)].

**LaSOT.** LaSOT [[15](#_bookmark39)] is a large-scale benchmark containing 280 test sequences. Table [2](#_bookmark48) shows the results of SwinTrack and comparisons with state-of-the-art trackers. From Table [2](#_bookmark48), we can see that our SwinTrack-T with light architecture reaches a SOTA performance with 0.667 SUC, 0.706 PRE, and 0.758 NPRE 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 vari- ant SwinTrack-B-384 gives a breakthrough new record with 0.702 and 0.753 of the SUC score and the PRE score, re- spectively.

**LaSOT**ext **.** The recently proposed LaSOText [[14](#_bookmark47)] is an ex- tension 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 tech- nique to deal with the distractors and achieves a promis- ing 0.482 SUC score. Compared with complicated Keep- Track, SwinTrack-T is simple and neat, yet shows com- parable 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 Keep- Track. When using a larger model, SwinTrack-B shows the best performance with 0.476 SUC score and 0.582 NPRE

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 NPRE score

SiamPRN++ [[25](#_bookmark22)]

DiMP [[2](#_bookmark51)] Ocean [[46](#_bookmark24)] SiamR-CNN [[40](#_bookmark52)] TrSiam [[41](#_bookmark25)] TrDiMP [[41](#_bookmark25)] STMTrack [[18](#_bookmark26)] TransT [[7](#_bookmark9)] STARK-ST50 [[44](#_bookmark1)]

**78.1**

**78.1**

|  |  |  |  |
| --- | --- | --- | --- |
| SiamPRN++ [[25](#_bookmark22)] | 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](#_bookmark1)] | 68.8 |  | 64.1 |
| SwinTrack-T SwinTrack-B | 69.0  **69.4** | 78.0 | 62.1  **64.3** |

Table 2. Performance comparison on LaSOT [[15](#_bookmark39)].

|  |  |  |  |
| --- | --- | --- | --- |
| Tracker | SUC  (%) | PRE  (%) | NPRE  (%) |

56.9

49.6 56.9 56.0 64.8 62.4 63.9 60.6 64.9 66.4 67.1 67.1

-

53.4 56.6

-

60.0 61.4 63.3 69.0

65.0

65.1

72.2

-

-

69.3

73.8

-

-

77.0 77.2

STARK-ST101 [[44](#_bookmark1)]

-

70.2

KeepTrack [[32](#_bookmark10)]

SwinTrack-B

SwinTrack-T 66.7 70.6 75.8

69.6 74.1 **78.6** **70.2** **75.3** 78.4

SwinTrack-B-384

Table 3. Performance comparison on LaSOText [[14](#_bookmark47)].

|  |  |  |  |
| --- | --- | --- | --- |
| 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](#_bookmark10)] | **48.2** | - | 58.0 |

|  |  |
| --- | --- |
| SwinTrack-T 46.9 SwinTrack-B 47.6 SwinTrack-B-384 47.5 | 52.9 57.6  **54.1** **58.2**  53.3 57.7 |

Table 4. Performance comparison on TrackingNet [[33](#_bookmark40)].

|  |  |  |  |
| --- | --- | --- | --- |
| 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](#_bookmark1)] | 82.0 | - | 86.9 |
| SwinTrack-T  85.5 87.0  80.8 82.5  77.9 80.4  **84.0** **83.2** **88.2**  SwinTrack-B  SwinTrack-B-384 | | | |

is better than KeepTrack.

**TrackingNet.** We evaluate our trackers on the test set of TrackingNet [[33](#_bookmark40)]. The results are shown in Table [4](#_bookmark49). From Table [4](#_bookmark49), we observe that our SwinTrack-T achieves compa- rable result of 0.808 SUC score. When using larger model

Table 5. Performance comparison on GOT-10k [[20](#_bookmark41)].

|  |  |  |  |
| --- | --- | --- | --- |
| Tracker | mAO  (%) | mSR50 (%) | mSR75 (%) |

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](#_bookmark1)] | 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 per- formance 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](#_bookmark41)] provides 180 sequences for test- ing and it requires trackers to be trained using GOT-10k train split only. The results and comparisons are displayed in Table [4](#_bookmark49). From Table [4](#_bookmark49), we see that SwinTrack-B achieves the best mAO of 0.694, outperforming other Transformer- based counterparts including START-ST101 (0.688 mAO), TransT (0.671 mAO), TrDiMP (0.671 mAP) and TrSiam (0.660 mAO).

**Efﬁciency** **comparison.** In addition to accuracy compari- son, we also report the comparisons of SwinTrack with oth- ers trackers on efﬁciency and complexity. As shown in Ta- ble [6](#_bookmark50), 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 solu- tion for any combination of attention operation, and incor- porating 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 fur- ther 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 track- ing. In *ICCV*, 2019.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu

Timofte. Know your surroundings: Exploiting scene infor- mation for object tracking. In *ECCV*, 2020.

[4] Jane Bromley, James W Bentz, Lon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Sckinger, 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, Haib-

ing 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. Proba-

bilistic regression for visual tracking. In *CVPR*, 2020.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina

Toutanova. Bert: Pre-training of deep bidirectional trans- formers for language understanding. *arXiv*, 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov,

Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Syl- vain Gelly, et al. An image is worth 16x16 words: Trans- formers 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 ob- ject tracking. In *CVPR*, 2019.

[16] Heng Fan and Haibin Ling. Siamese cascaded region pro-

posal 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* *Confer-* *ence* *on* *Learning* *Representations*, 2021.

[22] Matej Kristan, Jiri Matas, Ale Leonardis, Tomas Vojir, Ro-

man Pﬂugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka ehovin. 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 net- works. *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 pro- posal network. In *CVPR*, 2018.

[27] 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.

[28] Yawei Li, Kai Zhang, Jiezhang 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 Dollr, 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. *In-* *ternational* *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 Al-

subaihi, 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 Rezatoﬁghi, Nathan Tsoi, JunYoung Gwak, Amir

Sadeghian, Ian Reid, and Silvio Savarese. Generalized in-

tersection 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 at-

tention. In *ICML*, 2021.

[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco

Massa, Alexandre Sablayrolles, and Herv Jgou. Training

data-efﬁcient image transformers & distillation through at-

tention. In *ICML*, 2021.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszko-

reit, 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 Bas-

tian 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. Pyra-

mid vision transformer: A versatile backbone for dense pre-

diction 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 vi-

sual tracking. In *ICCV*, 2021.

[45] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko

Snderhauf. Varifocalnet: An iou-aware dense object de-

tector. 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, Xi-

aochen 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.