

Exploring Lightweight Hierarchical Vision Transformers for Efficient Visual Tracking

Ben Kang^{1,*} Xin Chen^{1,*} Dong Wang^{1,†} Houwen Peng^{2,†} Huchuan Lu¹

¹ School of Information and Communication Engineering, Dalian University of Technology

² Microsoft Research

Abstract

Transformer-based visual trackers have demonstrated significant progress owing to their superior modeling capabilities. However, existing trackers are hampered by low speed, limiting their applicability on devices with limited computational power. To alleviate this problem, we propose HiT, a new family of efficient tracking models that can run at high speed on different devices while retaining high performance. The central idea of HiT is the Bridge Module, which bridges the gap between modern lightweight transformers and the tracking framework. The Bridge Module incorporates the high-level information of deep features into the shallow large-resolution features. In this way, it produces better features for the tracking head. We also propose a novel dual-image position encoding technique that simultaneously encodes the position information of both the search region and template images. The HiT model achieves promising speed with competitive performance. For instance, it runs at 61 frames per second (fps) on the Nvidia Jetson AGX edge device. Furthermore, HiT attains 64.6% AUC on the LaSOT benchmark, surpassing all previous efficient trackers. Code and models are available at <https://github.com/kangben258/HiT>.

1. Introduction

Visual object tracking is a fundamental task in computer vision, which aims to track an arbitrary object given its initial state in a video sequence. In recent years, with the development of deep neural networks [25, 20, 39, 42], tracking has made significant progress. In particular, the utilization of transformers [42] has played a pivotal role in the development of several high-performance trackers [8, 50, 43, 53, 10, 52, 7]. Unfortunately, much of the research [27, 2, 8] has concentrated solely on achieving high performance without considering tracking speed.

* Equal contribution.

† Corresponding authors: Dong Wang (wdice@dlut.edu.cn), Houwen Peng (houwen.peng@microsoft.com).

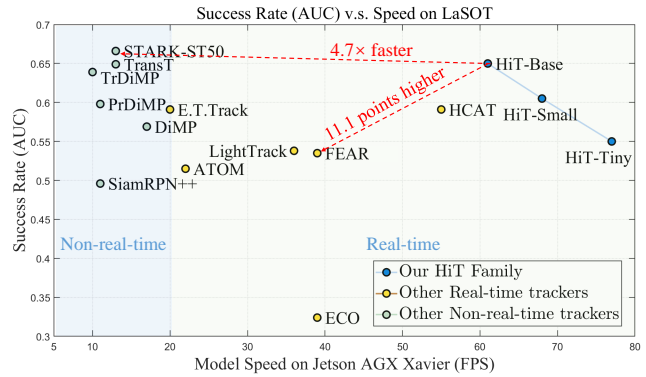


Figure 1: Comparison of our HiT with other trackers on LaSOT in terms of speed (horizontal axis) on the edge AI platform of Nvidia Jetson AGX Xavier and success rate (AUC) (vertical axis). Following the VOT real-time setting [23], we set the real-time line at 20 fps. Our HiT achieves the best real-time result, surpassing other efficient trackers.

While these trackers may achieve real-time speed on powerful GPUs, they lack competitiveness and advantages on resource-limited devices. For instance, TransT [8], which is a high-performance tracker, only achieves a speed of 5 frames per second (fps) on the Intel Core i9-9900K CPU and 13 fps on the Nvidia Jetson AGX. Consequently, a high-performance tracker with fast speed is critical.

The one-stream structure has gained popularity in tracking applications [52, 5, 48, 10]. This structure performs feature extraction and feature fusion jointly, leveraging the capabilities of the backbone network [14] that has been pre-trained for image classification. In our work, we also adopt the one-stream architecture, leveraging a pre-trained lightweight transformer backbone network. However, there exists a substantial gap between the tracking field and the image classification field. In the image classification field, lightweight networks [18, 33, 47] frequently incorporate a hierarchical architecture with high-stride downsampling to decrease computational expenses. However, large-stride downsampling often leads to a loss of critical information,

which is crucial for accurate tracking. This naturally raises the question of how to reconcile the need for detailed information in tracking with the large-stride downsampling in the hierarchical backbone network.

To tackle this problem, we introduce the Bridge Module, which integrates features from different levels of the hierarchical backbone. The Bridge Module fuses deep semantic information with shallow detail information, mitigating the information loss resulting from large-stride downsampling. By combining the proposed Bridge Module with lightweight hierarchical backbone LeViT [18], we develop HiT, a new family of efficient tracking models. Moreover, we proposed a novel relative position encoding technique, called dual-image position encoding, to improve the position information. This method encodes the position information of the template and search region jointly, enhancing the interaction between them.

Our extensive experiments validate the effectiveness and efficiency of HiT. Specifically, compared to the high-speed tracker FEAR [28], HiT-Base achieves an 11.1% higher AUC score on the LaSOT benchmark while being 1.6 times faster than FEAR on Nvidia Jetson AGX Xavier. In comparison to the high-performance tracker STARK-ST50 [50], HiT-Base delivers similar performance while being 4.7 times faster on AGX, representing a significant improvement over previous real-time trackers. Our main contributions are summarized as follows:

- We propose the Bridge Module, which incorporates the high-level information of deep features into the shallow large-resolution features, thereby mitigating the information loss caused by the large-stride downsampling. This approach enables the use of large-stride downsampling hierarchical backbones for tracking purposes.
- To improve position accuracy, we introduce a dual-image position encoding approach that jointly encodes position information from both the template and the search region.
- Building upon these components, we introduce HiT, a family of efficient tracking models. HiT exhibits promising performance while maintaining exceptionally fast processing speeds. Empirical evaluations demonstrate that HiT outperforms state-of-the-art efficient tracking algorithms.

2. Related Work

Visual Tracking. Siamese-based methods [1, 40, 27, 44, 26, 49, 19, 9, 55] are popular in tracking. The Siamese-based framework typically employs two backbone networks with shared parameters to extract the features of the template and the search region images, uses a correlation-based network for feature interaction, and finally uses head networks for prediction. TransT [8], TMT [43], and their

follow-up works [50, 30, 32, 38, 17] further improve tracking performance by introducing the transformer [42] for the feature interaction. Recently, a one-stream framework establishes new state-of-the-art performance in tracking, such as MixFormer [10], SBT [48], SimTrack [5], and OS-Track [52]. The one-stream framework jointly performs feature extraction and feature fusion with the backbone network. This framework is simple yet effective by leveraging the capabilities of the backbone network that has been pre-trained for image classification. However, these methods are developed for powerful GPUs, and their speeds on edge devices are slow, limiting their applicability. In this work, we also adopt the one-stream framework and we focus on making this framework more efficient.

Efficient Tracking Network. Practical applications require efficient trackers that can achieve high performance and fast speed on edge devices. Early methods ECO [11] and ATOM [12] achieve real-time speed on edge devices, but the performance is inferior compared with current state-of-the-art trackers. Recently, some efficient trackers have been developed. LightTrack [51] uses NAS to search networks, which entails a low computational amount and relatively high performance. FEAR [4] obtains a family of efficient and accurate trackers by employing a dual-template representation and a pixel-wise fusion block. However, there is still a large performance gap between these efficient trackers and the popular high-performance trackers [8, 50]. In this work, the proposed HiT not only runs at high speed on edge devices but also achieves competitive results compared with high-performance trackers. For example, compared with TransT [8], our method performs only 0.3% lower (in AUC) on LaSOT but 4.7 times faster on AGX.

Vision Transformer. ViT [14] introduces the transformer to image classification and has achieved impressive performance. After that, a large number of vision transformer networks [41, 54, 46, 45, 30] are developed. Transformers are popular for their superior modeling capabilities but are limited in speed. Therefore, many lightweight vision transformers [33, 18, 47] have emerged, greatly accelerating the speed of transformer-based networks. Different from the classic vision transformer, these lightweight transformers employ a hierarchical architecture with high-stride downsampling to decrease computational expenses. In this work, we focus on leveraging the lightweight hierarchical vision transformer with the one-stream tracking framework. By default, we employ LeViT [18] as the backbone network. However, our method has the following fundamental differences from LeViT. (1) The overall architectures are different. LeViT makes predictions on the final heavily-downsampled features. Our HiT employs a Bridge Module to fuse features of different stages, and the predictions are made on the fused large-resolution features. We also modify the transformer module so that it can handle the search

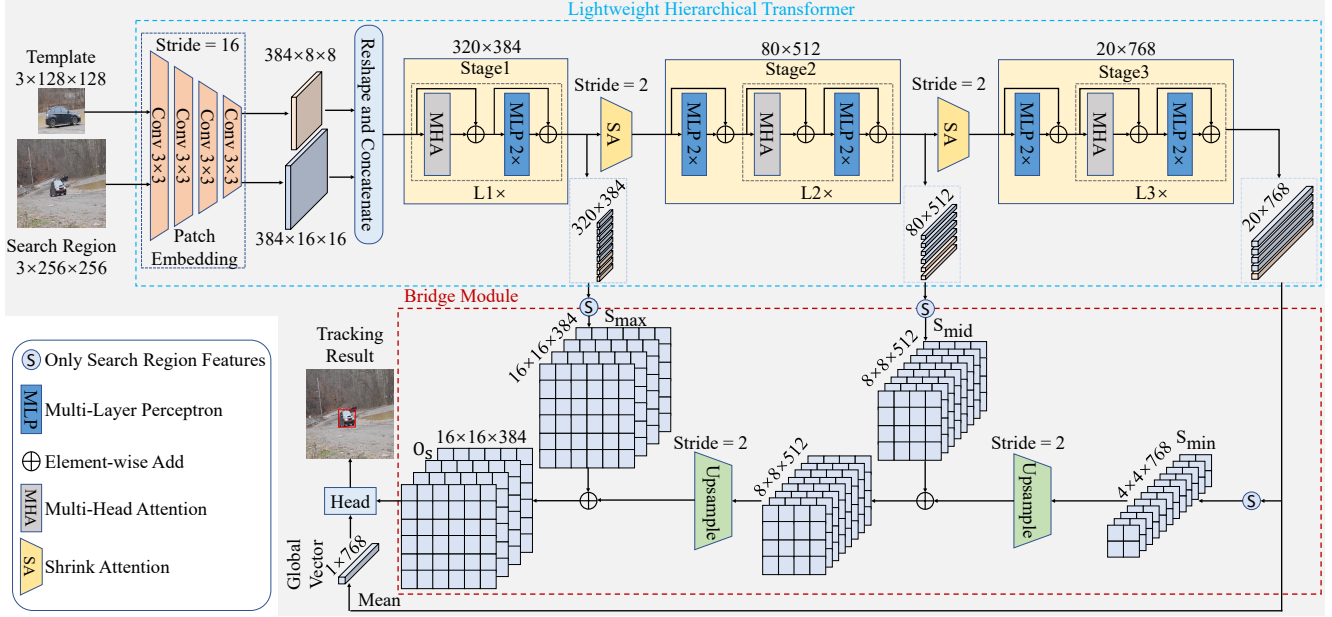


Figure 2: Architecture of the proposed HiT framework. The HiT framework contains three components: a lightweight hierarchical vision transformer for feature extraction and fusion, a Bridge Module that combines multi-stage features, and a prediction head.

region and template simultaneously. (2) The tasks are different. LeViT is designed for image classification, focusing more on high-level semantic information. Our framework is for tracking, where detailed information is also crucial. (3) The position encodings are different. LeViT encodes the position information for a single image. We develop the dual-image position encoding to encode the position information of the template and search region jointly to enhance the level of detail.

3. Method

This section presents the HiT method in detail. First, we briefly overview our HiT framework. Then, we depict the model architecture, including the lightweight hierarchical vision transformer with our dual-image position encoding, the proposed Bridge Module, and the head network. Finally, we introduce the training and inference pipelines.

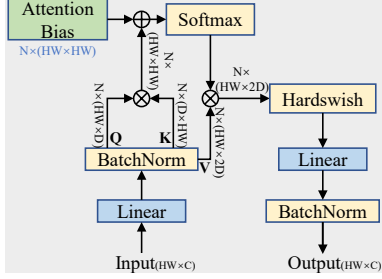
3.1. Overview

As shown in Figure 2, HiT is a one-stream tracking framework consisting of three components: the lightweight hierarchical transformer, the proposed Bridge Module, and the head network. The image pair (including the search region and template images) are fed into the lightweight hierarchical transformer for feature extraction and feature fusion. The core modules of the hierarchical vision transformer are the Multi-Head Attention (MHA), the Shrink Attention (SA), and the dual-image position encoding. MHA

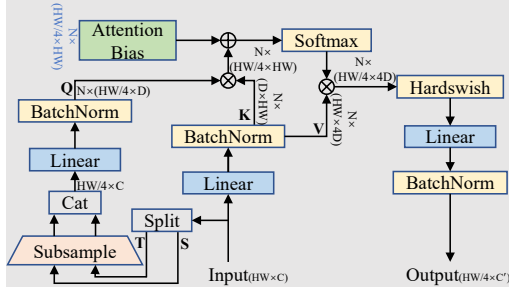
extracts and fuses the features of the search and template images, SA downsamples the features, and dual-image position encoding encodes the position information of the search and template images jointly. From each stage of the hierarchical transformer, we obtain a sequence of features with different resolutions. From the last stage of the hierarchical transformer, we obtain a global vector by averaging the final output features. The feature sequence is input to the Bridge Module, in which features are fused to obtain enhanced features. Finally, the global vector and the enhanced features are input into the prediction head to obtain the tracking result.

3.2. Lightweight Hierarchical Vision Transformer

Hierarchical Backbone. We use LeViT [18], a lightweight hierarchical vision transformer as the backbone of HiT. We adapt it into our tracking framework. Specifically, the input of the transformer is the template image $\mathbf{Z} \in \mathbb{R}^{3 \times H_z \times W_z}$ and the search region image $\mathbf{X} \in \mathbb{R}^{3 \times H_x \times W_x}$. First, downsample the image pair by a factor of 16 through patch embedding to get $\mathbf{Z}_p \in \mathbb{R}^{C \times \frac{H_z}{16} \times \frac{W_z}{16}}$ and $\mathbf{X}_p \in \mathbb{R}^{C \times \frac{H_x}{16} \times \frac{W_x}{16}}$. Then \mathbf{Z}_p and \mathbf{X}_p are flattened and concatenated in the spatial dimension and then fed into the following hierarchical transformer. The hierarchical transformer consists of three stages. The i -th stage has L_i blocks ($L_1=L_2=L_3=4$, by default). Each block consists of a Multi-Head Attention and an MLP in the residual form. Shrink Attention modules are used to



(a) Multi-Head Attention (MHA)



(b) Shrink Attention (SA)

Figure 3: Detailed architectures of MHA and SA.

connect each stage, and it downsamples the features by a factor of 4 in the spatial dimension. For the output features of each stage, we get the partial features corresponding to the search image. For the final stage, we also average its output features and get a global vector $\mathbf{G} \in \mathbb{R}^{1 \times C_{min}}$ and a feature sequence with three feature maps of different size: $\mathbf{S}_{max} \in \mathbb{R}^{H_{max} \times W_{max} \times C_{max}}$, $\mathbf{S}_{mid} \in \mathbb{R}^{H_{mid} \times W_{mid} \times C_{mid}}$, $\mathbf{S}_{min} \in \mathbb{R}^{H_{min} \times W_{min} \times C_{min}}$, where $C_{max} = 384$, $C_{mid} = 512$, $C_{min} = 768$.

Multi-Head Attention (MHA). The structure of MHA is shown in Figure 3a. The number of channels of \mathbf{Q} and \mathbf{K} is half of \mathbf{V} to reduce the amount of calculation. Following the LeViT, we use the attention bias as a relative position encoding rather than the absolute position encoding. We generate the attention bias in the way of our dual-image position encoding, and the details will be introduced later. The mechanism of MHA can be summarized as:

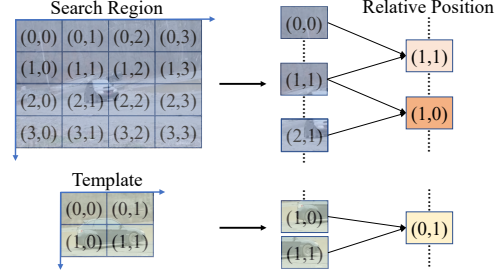
$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{B}_i) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{B}_i\right)\mathbf{V},$$

$$\mathbf{H}_i = \text{Hardswish}(\text{Attn}(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V, \mathbf{B}_i)),$$

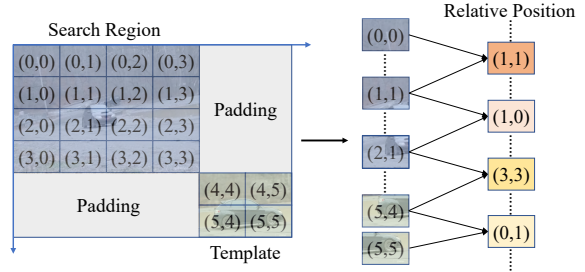
$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\mathbf{H}_1, \dots, \mathbf{H}_N)\mathbf{W}^O, \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{HW \times C}$ is the input, $\mathbf{B}_i \in \mathbb{R}^{HW \times HW}$ is the attention bias, and $\mathbf{W}_i^Q \in \mathbb{R}^{C \times D}$, $\mathbf{W}_i^K \in \mathbb{R}^{C \times D}$, $\mathbf{W}_i^V \in \mathbb{R}^{C \times 2D}$, and $\mathbf{W}^O \in \mathbb{R}^{2ND \times C}$ are parameter matrices.

Shrink Attention (SA). The structure of SA is shown in Figure 3b. The SA connects the stages of the hierarchical transformer and downsamples the features. The architecture of SA is the same as MHA except for the following modifi-



(a) Previous Position Encoding



(b) Our dual-image Position Encoding

Figure 4: Comparison of our dual-image position encoding and the previous position encoding.

cations: 1) To generate the Query \mathbf{Q} , we split the 2D input features into template features (denoted as \mathbf{T}) and search region features (denoted as \mathbf{S}) based on their location. We reshape them to 3D features and subsample them by a factor of 2 in each spatial direction. Then we re-flatten the features and concatenate them in the spatial dimension. In this way, the size of \mathbf{Q} is down-sampled by a factor of 4 in total, thus the final output of SA is also down-sampled. 2) The number of channels of \mathbf{V} is doubled to alleviate the information loss caused by downsampling, and the number of channels of output features is also increased.

Dual-image Position Encoding. Following LeViT, we use attention bias to inject the relative position information into attention maps. To better encode the joint position information of the template and the search region, we generate the attention bias in the way of our dual-image position encoding. Specifically, attention bias is a set of parameters that can be learned. We calculate the relative positions between every two pixels, use the relative positions as indexes to find the corresponding learned parameters, and add them to the attention map to introduce the position information. It is calculated as

$$\text{Bias}^h = \mathbf{B}^h(|x - x'|, |y - y'|), \quad (2)$$

where (x, y) and $(x', y') \in [H] \times [W]$ are the two pixels on the feature map. \mathbf{B}^h is the learned parameters, and Bias^h is the indexed learned parameters. As shown in Figure 4a, the previous position encoding encodes the template and the search region separately, and the positions of the two images partially overlap, causing information confu-

sion. More concretely, the position of the template and the upper left portion of the search region is the same. To address this problem, in our dual-image position encoding, we diagonally arrange the template and the search region and encode their position information jointly as shown in Figure 4b. The diagonal arrangement encodes unique horizontal and vertical coordinates for each pixel of the template and search region, avoiding the confusion of detailed position information.

3.3. Bridge Module and Head

Bridge Module. The Bridge Module fuses features of different stages of the hierarchical transformer to obtain the enhanced feature that contains rich detailed and semantic information. It bridges the lightweight hierarchical transformer and the tracking framework. To ensure the efficiency of the model, we expect the Bridge Module to be a minimal architecture, that is, it should be as concise as possible while being effective. To this end, we employ a very simple architecture for the Bridge Module and find it provides compelling results. As shown in the red box in Figure 2, the transformer outputs three 2D features with different sizes. We reshape these 2D features to 3D feature maps, denoted as \mathbf{S}_{\min} , \mathbf{S}_{mid} , and \mathbf{S}_{\max} . First, we upsample \mathbf{S}_{\min} and add it to \mathbf{S}_{mid} . Then, we upsample the obtained feature and add it to \mathbf{S}_{\max} , getting the final enhanced feature. We employ a transpose convolutional layer with stride 2 for all upsampling. The mechanism of the Bridge Module can be summarized as

$$\mathbf{O}_s = \mathbf{S}_{\max} + \text{Upsample}(\mathbf{S}_{\text{mid}} + \text{Upsample}(\mathbf{S}_{\min})) \quad (3)$$

where $\mathbf{O}_s \in \mathbb{R}^{H_{\max} \times W_{\max} \times C_{\max}}$ is the output of the Bridge Module; $\mathbf{S}_{\max} \in \mathbb{R}^{H_{\max} \times W_{\max} \times C_{\max}}$, $\mathbf{S}_{\text{mid}} \in \mathbb{R}^{H_{\text{mid}} \times W_{\text{mid}} \times C_{\text{mid}}}$ and $\mathbf{S}_{\min} \in \mathbb{R}^{H_{\min} \times W_{\min} \times C_{\min}}$ are feature maps output by the lightweight hierarchical transformer. The Bridge Module combines the deep semantic information and the shallow detail information, alleviating the information loss caused by the large-stride downsampling. This minimal network provides compelling results while remaining efficient.

Head. We employ the corner head [50] for prediction. First, we calculate the attention map between \mathbf{G} and \mathbf{O}_s . Then, we re-weight \mathbf{O}_s with the attention map. In this way, the local features are enhanced or suppressed according to global information. Finally, \mathbf{O}_s is fed to a fully-convolution network, obtaining the coordinates of the target.

3.4. Training objective and Inference

We combine the ℓ_1 loss and the generalized GIoU loss [36] as the training objective. The loss function can be formulated as

$$\mathcal{L} = \lambda_G \mathcal{L}_{GIoU}(b_i, \hat{b}_i) + \lambda_l \mathcal{L}_l(b_i, \hat{b}_i). \quad (4)$$

| Model | | HiT-Base | HiT-Small | HiT-Tiny |
|------------------------|-----|----------|-----------|----------|
| PyTorch Speed (fps) | GPU | 175 | 192 | 204 |
| | CPU | 33 | 72 | 76 |
| | AGX | 61 | 68 | 77 |
| ONNX Speed (fps) | GPU | 274 | 400 | 455 |
| | CPU | 42 | 98 | 125 |
| | AGX | 75 | 119 | 145 |
| Macs(G) | | 4.34 | 1.13 | 0.99 |
| Params(M) | | 42.14 | 11.03 | 9.59 |

Table 1: Details of our HiT model variants.

where b_i represents the groundtruth, and \hat{b}_i represents the predicted box. λ_G and λ_l are weights, in experiments, we set $\lambda_G = 2$ and $\lambda_l = 5$. During inference, the template is initialized in the first frame of a video sequence. For each subsequent frame, the search region is cropped based on the target’s bounding box of the previous frame. The whole framework is end-to-end. The template and search images are input into our tracker, and the output of the model is the final result. We do not use any additional post-processing methods, such as window penalty and scale penalty [27].

4. Experiments

4.1. Implementation Details

Model. We develop three variants of HiT models with different lightweight transformers, as elaborated in Tab. 1. We adopt LeViT-384 [18], LeViT-128, and LeViT-128S for HiT-Base, HiT-Small, and HiT-Tiny, respectively. In addition, Tab. 1 reports model parameters, FLOPs, and inference speed on multiple devices. All the models are implemented with Python 3.8 and PyTorch 1.11.0.

Training. The training datasets for our model include the train-splits of TrackingNet [35], GOT-10k [21], LaSOT [16], and COCO2017 [29]. The input of the network is an image pair consisting of a template image and a search image. For video datasets, we sample the image pair from a random video sequence. For the image dataset COCO, we randomly select an image and apply data augmentations to generate an image pair. Common data augmentations such as scaling, translation, and jittering are applied on the image pair. The search region and the template are obtained by expanding the target box by a factor of 4 and 2, respectively. The search and template images are resized to 256×256 and 128×128 , respectively. The transformer is initialized with ImageNet [37] pretrained LeViT [18], and the other parameters of HiT are initialized randomly. The optimizer is the AdamW optimizer [31], with the weight decay of $1e-4$. The initial learning rate of HiT is $5e-4$. We use 4 Nvidia RTX 3090 GPUs to train our model for 1500 epochs with a batch size of 128. Each epoch contains 60,000 sampling pairs. The learning rate is reduced by $10 \times$ at epoch 1200.

Inference. As stated in Sec. 3.4, the HiT framework is

Table 2: State-of-the-art comparison on TrackingNet [35], LaSOT [16], and GOT-10k [21] benchmarks. We use gray color to denote our trackers. The best three real-time results are shown in red, blue and green fonts, and the best non-real-time results are shown in underline font.

| | Method | TrackingNet | | | LaSOT | | | GOT-10k | | | PyTorch Speed (fps) | | |
|---------------|-----------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------------|--------------------|---------------------|-----|-----|
| | | AUC | P_{Norm} | P | AUC | P_{Norm} | P | AO | SR _{0.5} | SR _{0.75} | GPU | CPU | AGX |
| Real-time | HiT-Base | 80.0 | 84.4 | 77.3 | 64.6 | 73.3 | 68.1 | 64.0 | 72.1 | 58.1 | 175 | 33 | 61 |
| | HiT-Small | 77.7 | 81.9 | 73.1 | 60.5 | 68.3 | 61.5 | 62.6 | 71.2 | 54.4 | 192 | 72 | 68 |
| | HiT-Tiny | 74.6 | 78.1 | 68.8 | 54.8 | 60.5 | 52.9 | 52.6 | 59.3 | 42.7 | 204 | 76 | 77 |
| | FEAR [4] ¹ | - | - | - | 53.5 | - | 54.5 | 61.9 | 72.2 | - | 105 | 60 | 38 |
| | HCAAT [6] | 76.6 | 82.6 | 72.9 | 59.3 | 68.7 | 61.0 | 65.1 | 76.5 | 56.7 | 195 | 45 | 55 |
| | E.T.Track [3] | 75.0 | 80.3 | 70.6 | 59.1 | - | - | - | - | - | 40 | 47 | 20 |
| | LightTrack [51] | 72.5 | 77.8 | 69.5 | 53.8 | - | 53.7 | 61.1 | 71.0 | - | 128 | 41 | 36 |
| | ATOM [12] | 70.3 | 77.1 | 64.8 | 51.5 | 57.6 | 50.5 | 55.6 | 63.4 | 40.2 | 83 | 18 | 22 |
| | ECO [11] | 55.4 | 61.8 | 49.2 | 32.4 | 33.8 | 30.1 | 31.6 | 30.9 | 11.1 | 240 | 15 | 39 |
| Non-real-time | OTrack-256 [52] | 83.1 | 87.8 | 82.0 | 69.1 | 78.7 | 75.2 | 71.0 | 80.4 | 68.2 | 105 | 11 | 19 |
| | MixFormer-L [10] | <u>83.9</u> | <u>88.9</u> | 83.1 | <u>70.1</u> | <u>79.9</u> | <u>76.3</u> | <u>75.6</u> | <u>85.7</u> | <u>72.8</u> | 18 | - | - |
| | Sim-B/16 [5] | 82.3 | - | <u>86.5</u> | 69.3 | <u>78.5</u> | - | 68.6 | <u>78.9</u> | 62.4 | 87 | 10 | 16 |
| | STARK-ST50 [50] | 81.3 | 86.1 | - | 66.6 | - | - | 68.0 | 77.7 | 62.3 | 50 | 7 | 13 |
| | TransT [8] | 81.4 | 86.7 | 80.3 | 64.9 | 73.8 | 69.0 | 72.3 | 82.4 | 68.2 | 63 | 5 | 13 |
| | TrDiMP [43] | 78.4 | 83.3 | 73.1 | 63.9 | - | 61.4 | 68.8 | 80.5 | 59.7 | 41 | 5 | 10 |
| | TrSiam [43] | 78.1 | 82.9 | 72.7 | 62.4 | - | 60.6 | 67.3 | 78.7 | 58.6 | 40 | 5 | 10 |
| | PrDiMP [13] | 75.8 | 81.6 | 70.4 | 59.8 | 68.8 | 60.8 | 63.4 | 73.8 | 54.3 | 47 | 6 | 11 |
| | DiMP [2] | 74.0 | 80.1 | 68.7 | 56.9 | 65.0 | 56.7 | 61.1 | 71.7 | 49.2 | 77 | 10 | 17 |
| | SiamRPN++ [26] | 73.3 | 80.0 | 69.4 | 49.6 | 56.9 | 49.1 | 51.7 | 61.6 | 32.5 | 56 | 4 | 11 |

end-to-end, and we do not involve any hyper-parameters during inference.

4.2. State-of-the-art Comparisons

According to the speed on edge device Nvidia Jetson AGX Xavier, we divide trackers into real-time trackers and non-real-time trackers. Following the VOT real-time setting [23], we set the real-time line at 20 *fps*. We compare HiT with the state-of-the-art real-time trackers and non-real-time trackers on six tracking benchmarks. We evaluate these trackers’ speed on three platforms: Nvidia GeForce RTX 2080 GPU, Intel Core i9-9900K @ 3.60GHz CPU, and Nvidia Jetson AGX Xavier edge device. Tables 2 and 3 show the results.

TrackingNet. TrackingNet [35] is a large-scale dataset containing a variety of situations in natural scenes and multiple categories, and its test set includes 511 video sequences. As reported in Table 2, HiT-Base and HiT-small achieve competitive results compared with the previous real-time trackers. HiT-Base gets the best AUC of 80.0%, surpassing the previous best real-time tracker HCAAT [6] by 3.4%. Compared to non-real-time tracker STARK-ST50 [50], HiT-Base achieves comparable performance to it in AUC (80.0 *vs.* 81.3) while being 3.5 \times faster on the GPU, 4.7 \times faster on the CPU, and 4.7 \times faster on the AGX.

LaSOT. LaSOT [16] is a large-scale, long-term dataset

¹Due to limitations in access to the FEAR-L model, we compare our method with FEAR-XS in this table. Nevertheless, our model also performs better than FEAR-L. For example, HiT-Base performs 6.7% higher than FEAR-L (in AUC) on LaSOT.

containing 1400 video sequences, with 1120 training videos and 280 test videos. The results on LaSOT are shown in Table 2. HiT-Base achieves the best real-time results of 64.6%, 73.3%, and 68.1% in AUC, P_{Norm} , and P, respectively. HiT-Small achieves the second-best AUC score. Compared with the recent efficient tracker FEAR [4], HiT-Base and HiT-Small outperform it by 11.1% and 7.0% in AUC. Moreover, HiT-Base and HiT-Small surpass the third-best real-time tracker HCAAT [6] by 5.3% and 1.2% in AUC. Compared with the non-real-time tracker TransT [8], HiT-Base performs only 0.3% lower but has a much faster speed.

GOT-10k. GOT-10k [21] is a large-scale and challenging dataset that contains 10k training sequences and 180 test sequences. As shown in Table 2, HiT-Base obtains the second best real-time results of 64.0% AO score. HiT-Small obtains the third-best AO score of 62.6%. HiT-Base surpasses the recent efficient tracker FEAR [4] by 2.1% .

Speed. Table 2 reports the speeds of trackers. On the GPU, HiT-Base, HiT-Small, and HiT-Tiny run at 175 *fps*, 192 *fps*, and 204 *fps*, which are 1.66 \times , 1.82 \times , and 1.94 \times faster than FEAR [4]. On the AGX edge device, HiT-Base, HiT-Small, and HiT-Tiny run at 61 *fps*, 68 *fps*, and 77 *fps*, which are 1.61 \times , 1.79 \times , and 2.03 \times faster than FEAR. On the CPU, HiT-Base, HiT-Small, and HiT-Tiny run at 33 *fps*, 72 *fps*, and 76 *fps*. Only HiT-base is slower than FEAR but still achieves real-time speed. Overall, HiT achieves fast speeds on multiple devices. We believe that the fast speed is beneficial for the applicability of tracking.

NFS. NFS [22] is a challenging dataset with fast-moving objects, which includes 100 video sequences. Table 3 shows

| | Method | NFS | UAV123 | LaSOT _{ext} |
|---------------|-----------------|-------------|-------------|----------------------|
| Real-time | HiT-Base | 63.6 | 65.6 | 44.1 |
| | HiT-Small | 61.8 | 63.3 | 40.4 |
| | HiT-Tiny | 53.2 | 58.7 | 35.8 |
| | HCAT [6] | 63.5 | 62.7 | - |
| | FEAR [4] | 61.4 | - | - |
| | E.T.Track [3] | 59.0 | 62.3 | - |
| | LightTrack [51] | 55.3 | 62.5 | - |
| | ATOM [12] | 58.4 | 64.2 | 37.6 |
| | ECO [11] | 46.6 | 53.2 | 22.0 |
| Non-real-time | OTrack-256 [52] | 64.7 | 68.3 | 47.4 |
| | TransT [8] | 65.7 | 69.1 | - |
| | TrDiMP [43] | 66.5 | 67.5 | - |
| | TrSiam [43] | 65.8 | 67.4 | - |
| | PrDiMP [13] | 63.5 | 68.0 | - |
| | DiMP [2] | 62.0 | 65.3 | 39.2 |
| | SiamRPN++ [26] | 50.2 | 61.6 | 34.0 |

Table 3: Comparison with state-of-the-art methods on additional benchmarks in AUC score.

| Method | EAO | Accuracy | Robustness |
|----------------------|--------------|--------------|--------------|
| HiT-Base | 0.252 | 0.447 | 0.688 |
| FEAR [4] | 0.250 | 0.436 | 0.655 |
| STARK-S [50] | 0.237 | 0.407 | 0.631 |
| STARK-Lightning [50] | 0.204 | 0.391 | 0.565 |
| LightTrack [51] | 0.225 | 0.391 | 0.641 |
| E.T.Track [3] | 0.224 | 0.372 | 0.631 |

Table 4: VOT real-time experiment on NVidia Jetson AGX.

that HiT-Base and HiT-Small achieve the best and the third-best real-time performance, respectively.

UAV123. The UAV123 dataset [34] is constructed with low-altitude UAVs and contains 123 video clips. As shown in Table 3, HiT-Base achieves the best results compared to the other real-time trackers, achieving an AUC score of 65.6%. HiT-Base performs superior to HCAT [6] and E.T.Track [3] with 2.9% and 3.3%, respectively.

LaSOT_{ext}. LaSOT_{ext} [15] is a recently released tracking dataset consisting of 150 videos from 15 object classes. It is an extension of LaSOT. The results of HiT on LaSOT_{ext} are shown in Table 3. HiT-Base, HiT-Small, and HiT-Tiny achieve competitive results with 44.1%, 40.4%, and 35.8% AUC scores, respectively.

VOT. We also conduct VOT real-time experiments on NVidia Jetson AGX using the VOT2021 benchmark [24]. The results are shown in the Table 4. HiT-Base achieves the best results compared to the other real-time trackers, achieving an EAO score of 25.2%.

4.3. Ablation Study and Analysis

In this section, we provide detailed ablation experiments to analyze our HiT method. For the ablation study, we use HiT-Base as the baseline model. All models in ablation experiments are trained for 500 epochs.

Different combinations of features. To verify the effec-

| # | Max | Mid | Min | LaSOT | TrackingNet | GOT-10k |
|---|-----|-----|-----|-------------|-------------|-------------|
| 1 | ✓ | ✓ | ✓ | 63.7 | 78.9 | 65.4 |
| 2 | ✓ | | | 62.1 | 78.3 | 63.4 |
| 3 | | ✓ | | 61.9 | 78.1 | 64.1 |
| 4 | | | ✓ | 57.9 | 73.0 | 62.1 |
| 5 | ✓ | ✓ | | 62.6 | 78.7 | 63.0 |
| 6 | ✓ | | ✓ | 58.8 | 77.2 | 60.4 |
| 7 | | ✓ | ✓ | 60.3 | 78.4 | 63.6 |

Table 5: Comparison with different feature combining manners in AUC. We use gray color to denote the default setting. The best results are shown in the red fonts. Max, Mid, and Min denote the features of the transformer’s first, second, and third stages.

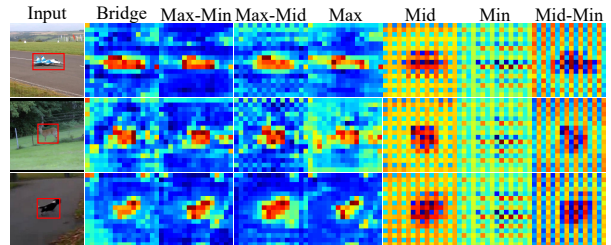


Figure 5: Visualization of the attention maps in the corner head of different feature combining manners. Bridge means our default manner, Max-Min means combining the Max and the Min features, Max-Mid means combining the Max and the Min features, Max, Mid, and Min mean only using the Max feature, Mid feature, and Min feature, respectively.

tiveness of the Bridge Module and explore which features are important, we compare different feature combinations in the Bridge Module. Table 5 shows the results. Max, Mid, and Min denote the features of the transformer’s first, second, and third stages, respectively. For a fair comparison, the features are upsampled to the same resolution in the comparison. The first row (#1) is our default setting. First, we do not use our Bridge Module and make predictions on the independent Max, Mid and Min features. Table 5 (#2, #3, and #4) shows these methods lead to inferior results, demonstrating the effectiveness of feature fusing with our Bridge Module. Second, Table 5 (#5, #6, and #7) reports the results of other candidate combination manners, and our default method works best. In our default method, using all three features bring more semantic and detailed information, leading to better results.

To further understand the Bridge Module, we visualize the attention map in the corner head of these features combining manners in Figure 5. In the visualization results, first, we find a collapse phenomenon in the methods that do not use the Max feature. Taking the Mid manner as an example, the final feature is from the second stage of the transformer, and it is up-sampled by a factor of 2. In this way, one pixel on the feature map is up-sampled to four pixel

| # | PE | LaSOT | TrackingNet | GOT-10k |
|---|-----|-------------|-------------|-------------|
| 1 | DI | 63.7 | 78.9 | 65.4 |
| 2 | Abs | 60.2 | 77.2 | 61.2 |
| 3 | Sep | 62.4 | 77.6 | 63.1 |
| 4 | Ver | 61.1 | 78.4 | 63.5 |
| 5 | Hor | 61.0 | 78.5 | 63.7 |

Table 6: Comparison of different Position Encoding (PE) in AUC score. DI denotes our dual-image PE. Abs denotes the absolute PE. Sep denotes the relative PE which encodes the template and search region separately. Ver and Hor denote the joint encoding of the template and search images in a vertical and horizontal arrangement, respectively.

points. In the visualization result, we can see that the attention collapses to a relatively fixed distribution for every four upsampling grids. The Min column and the Mid-Min column are similar to the Mid column. This shows that even if the deep feature is up-sampled to a larger resolution, it does not bring more detailed information. Therefore, it is crucial to involve the shallow large-resolution feature to supplement the information. Second, we find the attention map of our default method is more accurate than the methods that do not use the Min feature. This demonstrates that using deep features to supplement semantic information helps to improve the discriminative ability.

Different Position Encoding. Previous transformer-based trackers [8, 50] encode the position information of the search image and the template image separately. In our dual-image position encoding method, we assign a unique position for each image and jointly encode their position information. Here, we compare our method with four potential encoding methods, and the results are reported in Table 6. First, we compare our method with the absolute position encoding (denoted as Abs) and the relative position encoding which encodes the search and template images separately (denoted as Sep). Table 6 (#1 and #2) shows these methods perform inferior to our dual-image position encoding. The separate encoding does not model the positional relationship between the search and template images, and introduces overlapping positions of them, leading to inferior performance. Second, in our dual-image position encoding, we also explore different arrangements of the template and search region. By default, we diagonally arrange the template and the search region, as shown in Figure 4b. Here we compare it with two other arrangements: the vertical arrangement (denoted as Ver) and the horizontal arrangement (denoted as Hor). Table 6 (#1, #4 and #5) shows the default diagonally arrangement achieves the best performance. In the vertical and horizontal arrangements, the horizontal and vertical positions of the template and the search region are overlapping, leading to information loss. The diagonal arrangement assigns unique horizontal and vertical positions for the template and the search region, which is more informative. Therefore, we choose the diagonal arrangement.

| | | LeViT-384 [18] | PVT-Small [45] |
|---------------------|-------------|----------------|----------------|
| Benchmarks | LaSOT | 63.7 | 63.9 |
| | TrackingNet | 78.9 | 78.4 |
| | GOT-10k | 65.4 | 64.8 |
| PyTorch Speed (fps) | GPU | 175 | 91 |
| | CPU | 33 | 22 |
| | AGX | 61 | 30 |
| ONNX Speed (fps) | GPU | 274 | 133 |
| | CPU | 42 | 25 |
| | AGX | 75 | 32 |

Table 7: HiT with different lightweight hierarchical vision transformers.

Different Backbones. To evaluate the generalization of our HiT framework. We expand our framework with another hierarchical vision transformer PVT [45], the results are shown in Table 7. We employ PVT-Small[45] as the transformer backbone, and the other parts are consistent with HiT-Base. From Table 7, we can see that HiT with PVT-Small obtains a 63.9% AUC score on LaSOT, 78.4% AUC score on TrackingNet, and 64.8% AO score on GOT-10k, while the speed on all three platforms is real-time. This is also a competitive result compared with our base model with LeViT-384 and other efficient trackers. This demonstrates a superior generalization ability of our framework.

5. Conclusion

This work proposes a new family of efficient transformer-based tracking models, named HiT. HiT alleviates the gap between the tracking framework and the lightweight hierarchical transformers through our Bridge Module and dual-image position encoding. Extensive experiments demonstrate HiT achieves promising performance compared to state-of-the-art efficient trackers while running at a very fast speed. We hope this work could facilitate the practical applicability of visual tracking and narrow the gap between the tracking and lightweight transformer research.

Limitation. One limitation of HiT is that, despite achieving good performance, it shows the difficulty in dealing with distractors, since the method does not employ an explicit distractor-handling module. Moreover, this work focuses on bridging the gap between lightweight hierarchical transformers and the tracking framework. Therefore, we only make minimal adjustments to the existing hierarchical transformer but do not design a new transformer. In future work, we will investigate the lightweight transformer customized for tracking, and we hope this work could provide a basis for this.

Acknowledgement. Dr. Wang and Dr. Lu was supported in part by National Natural Science Foundation of China (Nos.62293540, 62293542, 62022021), in part by Joint Fund of Ministry of Education for Equipment Pre-research (No.8091B032155), in part by Fundamental Research Funds for the Central Universities (No.DUT22QN228).

References

- [1] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H S Torr. Fully-Convolutional Siamese Networks for Object Tracking. In *ECCV*, pages 850–865, 2016. [2](#)
- [2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning Discriminative Model Prediction for Tracking. In *ICCV*, pages 6181–6190, 2019. [1](#), [6](#), [7](#)
- [3] Philippe Blatter, Menelaos Kanakis, Martin Danelljan, and Luc Van Gool. Efficient Visual Tracking with Exemplar Transformers. In *WACV*, pages 1571–1581, 2023. [6](#), [7](#)
- [4] Vasyly Borsuk, Roman Vei, Orest Kupyn, Tetiana Martyniuk, Igor Krashenyi, and Jiří Matas. FEAR: Fast, Efficient, Accurate and Robust Visual Tracker. In *ECCV*, pages 644–663, 2022. [2](#), [6](#), [7](#)
- [5] Boyu Chen, Peixia Li, Lei Bai, Lei Qiao, Qihong Shen, Bo Li, Weihao Gan, Wei Wu, and Wanli Ouyang. Backbone is All Your Need: A Simplified Architecture for Visual Object Tracking. In *ECCV*, pages 375–392, 2022. [1](#), [2](#), [6](#)
- [6] Xin Chen, Ben Kang, Dong Wang, Dongdong Li, and Huchuan Lu. Efficient Visual Tracking via Hierarchical Cross-Attention Transformer. In *ECCVW*, pages 461–477, 2022. [6](#), [7](#)
- [7] Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *CVPR*, pages 14572–14581, 2023. [1](#)
- [8] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer Tracking. In *CVPR*, pages 8126–8135, 2021. [1](#), [2](#), [6](#), [7](#), [8](#)
- [9] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese Box Adaptive Network for Visual Tracking. In *CVPR*, pages 6667–6676, 2020. [2](#)
- [10] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-End Tracking with Iterative Mixed Attention. In *CVPR*, pages 13598–13608, 2022. [1](#), [2](#), [6](#)
- [11] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient Convolution Operators for Tracking. In *CVPR*, pages 6931–6939, 2017. [2](#), [6](#), [7](#)
- [12] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate Tracking by Overlap Maximization. In *CVPR*, pages 4660–4669, 2019. [2](#), [6](#), [7](#)
- [13] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic Regression for Visual Tracking. In *CVPR*, pages 7181–7190, 2020. [6](#), [7](#)
- [14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. [1](#), [2](#)
- [15] 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. [7](#)
- [16] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking. In *CVPR*, pages 5374–5383, 2019. [5](#), [6](#)
- [17] Shenyan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. AiATrack: Attention in attention for transformer visual tracking. In *ECCV*, pages 146–164, 2022. [2](#)
- [18] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. LeViT: a Vision Transformer in ConvNet’s Clothing for Faster Inference. In *ICCV*, pages 12239–12249, 2021. [1](#), [2](#), [3](#), [5](#), [8](#)
- [19] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking. In *CVPR*, pages 6268–6276, 2020. [2](#)
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. [1](#)
- [21] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. *IEEE TPAMI*, 43(5):1562–1577, 2021. [5](#), [6](#)
- [22] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for Speed: A Benchmark for Higher Frame Rate Object Tracking. In *ICCV*, pages 1134–1143, 2017. [6](#)
- [23] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking VOT2020 challenge results. In *ECCV*, pages 547–601, 2020. [1](#), [6](#)
- [24] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Čehovin, Alan Lukežič, et al. The ninth visual object tracking vot2021 challenge results. In *ICCV*, pages 2711–2738, 2021. [7](#)
- [25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. [1](#)
- [26] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. In *CVPR*, pages 4282–4291, 2019. [2](#), [6](#), [7](#)
- [27] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High Performance Visual Tracking With Siamese Region Proposal Network. In *CVPR*, pages 8971–8980, 2018. [1](#), [2](#), [5](#)
- [28] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, pages 936–944, 2017. [2](#)
- [29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, pages 740–755, 2014. [5](#)
- [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 *ICCV*, pages 9992–10002, 2021. 2
- [31] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 5
- [32] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *CVPR*, pages 8731–8740, 2022. 2
- [33] Sachin Mehta and Mohammad Rastegari. MobileViT: Lightweight, General-purpose, and Mobile-friendly Vision Transformer. In *ICLR*, 2022. 1, 2
- [34] Matthias Mueller, Neil Smith, and Bernard Ghanem. A Benchmark and Simulator for UAV Tracking. In *ECCV*, pages 445–461, 2016. 7
- [35] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In *ECCV*, pages 310–327, 2018. 5, 6
- [36] Hamid Rezaatoughi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *CVPR*, pages 658–666, 2019. 5
- [37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Sathesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 5
- [38] Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Transformer tracking with cyclic shifting window attention. In *CVPR*, pages 8791–8800, 2022. 2
- [39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1
- [40] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese Instance Search for Tracking. In *CVPR*, pages 1420–1429, 2016. 2
- [41] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 2
- [42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 1, 2
- [43] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking. In *CVPR*, pages 1571–1580, 2021. 1, 2, 6, 7
- [44] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast Online Object Tracking and Segmentation: A Unifying Approach. In *CVPR*, pages 1328–1338, 2019. 2
- [45] 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. In *ICCV*, pages 548–558, 2021. 2, 8
- [46] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing Convolutions to Vision Transformers. In *ICCV*, pages 22–31, 2021. 2
- [47] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. TinyViT: Fast Pretrainobjecting Distillation for Small Vision Transformers. In *ECCV*, pages 68–85, 2022. 1, 2
- [48] Fei Xie, Chunyu Wang, Guangting Wang, Yue Cao, Wankou Yang, and Wenjun Zeng. Correlation-aware deep tracking. In *CVPR*, pages 8751–8760, 2022. 1, 2
- [49] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. In *AAAI*, pages 12549–12556, 2020. 2
- [50] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning Spatio-Temporal Transformer for Visual Tracking. In *ICCV*, pages 10428–10437, 2021. 1, 2, 5, 6, 7, 8
- [51] Bin Yan, Houwen Peng, Kan Wu, Dong Wang, Jianlong Fu, and Huchuan Lu. LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search. In *CVPR*, pages 15180–15189, 2021. 2, 6, 7
- [52] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint Feature Learning and Relation Modeling for Tracking: A One-Stream Framework. In *ECCV*, pages 341–357, 2022. 1, 2, 6, 7
- [53] Bin Yu, Ming Tang, Linyu Zheng, Guibo Zhu, Jinqiao Wang, Hao Feng, Xuetao Feng, and Hanqing Lu. High-Performance Discriminative Tracking with Transformers. In *ICCV*, pages 9836–9845, 2021. 1
- [54] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet. In *ICCV*, pages 538–547, 2021. 2
- [55] Zhipeng Zhang and Houwen Peng. Deeper and Wider Siamese Networks for Real-Time Visual Tracking. In *CVPR*, pages 4591–4600, 2019. 2