SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 1
ISSN 2054-7390

A Deep Hyper Siamese Network for Real-Time Object Tracking

[1]Yongpeng Zhao, [2]Lasheng Yu, [3]Xiaopeng Zheng
[1,2,3]School of Computer Science and Engineering, Central South University, Changsha, Hunan, China;
zhaoyongpeng@csu.edu.cn; yulasheng@csu.edu.cn; 184612313@csu.edu.cn

ABSTRACT

Siamese networks have drawn increasing interest in the field of visual object tracking due to their balance of precision and efficiency. However, Siamese trackers use relatively shallow backbone networks, such as AlexNet, and therefore do not take full advantage of the capabilities of modern deep convolutional neural networks (CNNs). Moreover, the feature representations of the target object in a Siamese tracker are extracted through the last layer of CNNs and mainly capture semantic information, which causes the tracker's precision to be relatively low and to drift easily in the presence of similar distractors. In this paper, a new nonpadding residual unit (NPRU) is designed and used to stack a 22-layer deep ResNet, referred as ResNet22. After utilizing ResNet22 as the backbone network, we can build a deep Siamese network, which can greatly enhance the tracking performance. Considering that the different levels of the feature maps of the CNN represent different aspects of the target object, we aggregated different deep convolutional layers to make use of ResNet22's multilevel feature maps, which can form hyperfeature representations of targets. The designed deep hyper Siamese network is named DHSiam. Experimental results show that DHSiam has achieved significant improvement on multiple benchmark datasets.

Keywords: Deep Learning; Siamese Network; Visual Object Tracking; SiamFC; ResNet.

1 Introduction

Visual object tracking is a fundamental problem of computer vision. It aims to estimate the position of a specified target object in a changing video sequence, given its initial location. Although tracking at real-time speeds plays a vital role in many computer vision applications, such as surveillance, robotics and automatic driving [1], it is still a challenging task due to factors including occlusion, deformation, fast motion and background clutter, to name a few.

Recently, Siamese network-based trackers [2-9] have attracted much attention because of their high speed and precision. However, the backbone network used in these trackers is still the classic AlexNet [10], rather than modern deep CNNs such as ResNet [11]. These deep networks have proven to have more powerful feature extraction and generalization capabilities. However, if we straightforwardly replace the original shallow backbone network with a deeper one, this simple replacement results in a significant drop in performance when the network depth increases, as shown in Figure 1. This phenomenon shows that straightforwardly increasing the network depth has a negative impact on tracker performance.

In addition, the feature representation of the target object in these trackers is extracted through the last layer of CNNs, which focuses primarily on interclass differences. When a similar distractor appears near

DOI: 10.14738/tmlai.81.8020
Publication Date: 07th April, 2020
URL: http://dx.doi.org/10.14738/tmlai.81.8020

the target object, the tracker is likely to drift, as shown in Figure 2. It is well known that the different layers of a CNN represent different aspects of a target object in a feature hierarchy. Earlier layers mainly provide fine-grained features, such as texture and color, which helps to separate the target object from similar distractors. In contrast, features from later layers capture more semantic information that is beneficial in some challenging scenarios.

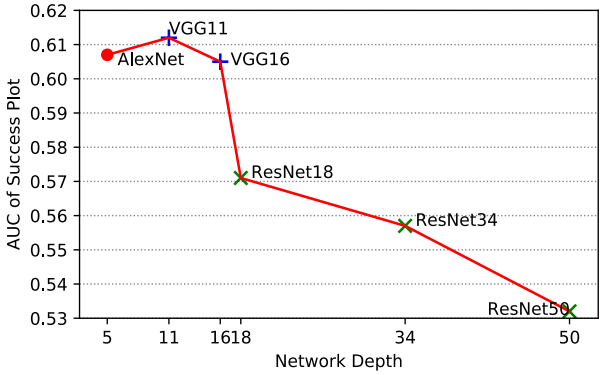


Figure 1. AUC of the success plot vs. network depth on the OTB50 dataset. The results were obtained using SiamFC with different backbone networks. The figure shows that as the network depth increases, the tracker performance decreases dramatically.

Figure 2. Tracking snapshots of SiamFC in a case with similar distractors. The figure shows that after the bottle was partially covered, SiamFC drifts to the neighboring distractors. If the lower-level color information can be used more fully, this drift can be corrected.

This paper addresses tracking issues by designing new residual modules and architectures apply the powerful capabilities of deeper backbone networks in Siamese trackers. Moreover, we aggregate multiple convolutional layers and extract multiple branch feature maps to collaboratively infer the target position. This provides a more comprehensive description of the target object to achieve high-precision tracking. In addition, the designed network is a lightweight architecture that enables the tracker to run at real-time speed.

The main contributions of this work include the following:

- Based on the proposed NPRU, a deeper network architecture is designed for a Siamese tracker. Experimental results demonstrate that the new architecture achieves significant improvements compare to the baseline trackers.
- The features extracted from multiple layers are used for layerwise aggregation to further enhance tracking precision.

2 Related Work

2.1 Siamese Network-based Tracker

First, let us briefly review Siamese network-based trackers. Siamese network-based trackers are based on a similarity comparison strategy. The pioneering method is SINT [2], which uses a fixed runtime to search for candidate regions that are most similar to the exemplar given in the starting frame. In follow-up work, Bertinetto et al. [3] introduce a fully convolutional Siamese network (SiamFC) to estimate the feature similarity between two frames regionwise. Since then, RASNet [4] has improved this similarity metric by learning the attention mechanism with a residual attention network. GOTURN tracker [5] attempt to predict motion between two frames with a deep regression network. Because no fine-tuning is performed online, these three trackers can be executed on a GPU at 86fps, 83fps, and 100fps. CFNet [6] and DSiam [7] can update the tracking model online by running an average template and a fast transformation learning module, respectively. There are many other follow-up studies, and these trackers can be divided into two types. One uses high-level semantic information or localization models to improve matching precision [1, 8]. The other enhances the offline Siamese model by updating it online [9].

The fully convolutional Siamese tracker, SiamFC, is the basic architecture discussed in this paper. The standard SiamFC architecture takes image pairs as input, including an exemplar image *z* and a search image *x*. The exemplar image *z* represents the target object, and the search image *x* represents the search area. Both input images are processed by a CNN φ with parameters θ. This will generate two feature maps. The feature map generated by the exemplar image *z* is used as a convolution kernel and the convolution operation is performed with the feature map generated by the search image *x* as follows:

$$f_\theta(z,x) = \varphi_\theta(z) * \varphi_\theta(x) + b \tag{1}$$

Where * represents the convolution operation, and $b \in R$ denotes a bias term. Eq. 1 corresponds to an exhaustive search of the pattern *z* on the search area *x*. The purpose is to match the maximum value in the response map *f* to the target location.

To achieve this, the network is trained offline using a random image pair (*z*, *x*) extracted from the training video and the corresponding ground-truth label *y*. The parameters θ of the CNN are obtained by minimizing the following logistic loss *l* on the training set:

$$\arg\min_\theta l\big(y, f_\theta(z,x)\big) \tag{2}$$

Previous methods [**Error! Reference source not found.**-**Error! Reference source not found.**] usually used the classic and relatively shallow AlexNet as the backbone network φ in this architecture. In this paper, a deeper network, ResNet22, is used to learn the model θ, which can greatly enhance the precision of tracking.

2.2 Deeper Backbone Network

Recently, Siamese trackers have received great attention and are developing rapidly. However, a relatively shallow backbone network such as AlexNet is used in these trackers, and they have not benefited from modern deep CNNs such as ResNet. These deep networks have proven to have more powerful feature extraction and generalization capabilities, have been widely used in many fields, and have achieved excellent results.

Zhipeng et al. [1] studied how to use deeper and wider CNNs to enhance the robustness and precision of tracking. The authors found that straightforwardly replacing AlexNet with existing powerful architectures

(such as ResNet) will not bring much improvement and can even cause a significant performance degradation as the network depth increases. Figure 1 illustrates this phenomenon. To investigate the specific reasons for this, the author performed many comparative experiments and determined that the size of the receptive field of the neuron, the network stride and the feature padding are the three important factors that affect tracking precision. The main reason for the performance degradation caused by deeper networks is that the network padding for convolutions causes positional bias in learning, as shown in Figure 3. Siamese trackers are a tracking method based on the similarity comparison strategy. However, the feature padding used for convolution will destroy this similarity when the target moves near the boundary of the search image, thereby causing potential position bias in model training.

Figure 3. Position bias caused by network padding. When the target was far from the boundary, the tracker tracked correctly (left picture), and when it moved near the edge, even if the target was not occluded, the tracker using the original ResNet drifted (middle picture). ResNet with NPRU correctly positioned the target (right picture).

2.3 Combination of Multilevel Feature Maps

Due to the complementary properties of different layers in CNNs, the aggregation of multilevel feature maps has been used in many recent deep learning tasks. In the object detection field, Kong et al. [12] designed a network called HyperNet, which combines the last layer and two intermediate layers and aggregates the hierarchical feature representations of the target; the trained network performs well in both target detection accuracy and recall rate. PVANET [13], proposed by Kye-Hyeon et al., adopts a similar network structure. In the visual object tracking field, Lijun et al. [14] built a general network (GNet) and a specific network (SNet) on top of the conv5 and conv4 feature layers, respectively. These two networks can be used interchangeably according to the distractor detection scheme. Naiyan et al. [15] successively learned correlation filters from the features in the third, fourth and fifth convolutional layers and then comprehensively determined the target location according to the maximum response of each filter. Both of the above tracking algorithms are superior to many state-of-the-art methods.

3 Our Approach

The proposal of the deep residual network (ResNet) [11] is a milestone event in the history of CNN. The core idea of ResNet is to introduce a so-called 'identity shortcut connection' to skip one or more layers and pass the original input signals directly to the subsequent layers. The neural network in this layer can thereby learn the residuals from the previous network instead of the entire output. A residual unit consists of 3 stacked convolutional layers and a shortcut connection that by-passes them. The three convolutional layers are 1×1, 3×3, and 1×1 convolution. The role of the 1×1 convolution layer is to reduce and then restore the number of input feature channels; their padding is 0. The 3×3 convolutional layer uses a padding of size 1 to ensure that the output size is the same as the input. The ResNet greatly reduces the number of network parameters while greatly deepening the network, thereby ensuring the real-time

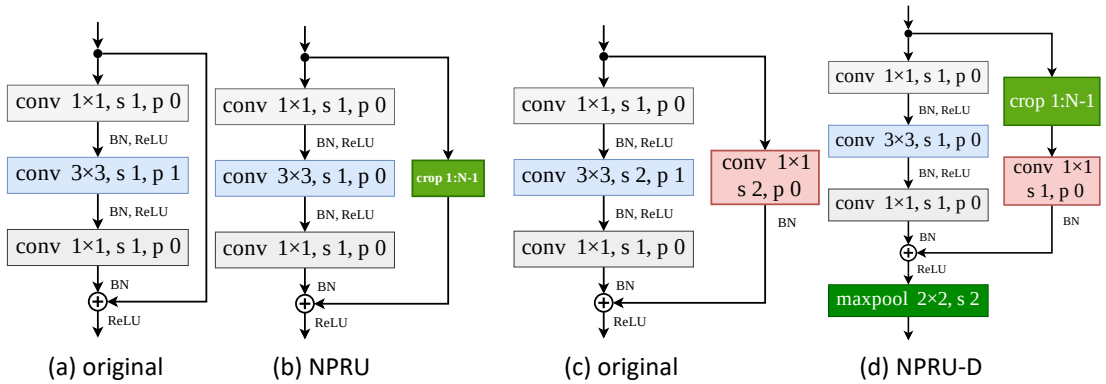
speed of the architecture designed in this paper. In fact, AlexNet has 5 convolutional layers and approximately 3.7 million parameters, while ResNet22 has 22 convolutional layers and approximately 1.4 million parameters.

In this section, a new residual module, called a nonpadding residual unit (NPRU), will be designed to eliminate the underlying position bias caused by feature padding. Then, we build a deeper backbone network by stacking NPRUs. We carefully control the network stride of the CNN and the size of the output response map. Finally, the designed network architecture is applied to the baseline tracker SiamFC.

3.1 Nonpadding Residual Units (NPRUs)

Network padding may cause position bias in Siamese trackers. To eliminate this effect, for a general residual unit, a new nonpadding residual unit is designed in this subsection. As shown in Figure 4(b), compared with the original residual unit shown in Figure 4(a), the padding of the 3×3 convolution layer is removed. To ensure that the shortcut connection and the output have the same size, the shortcut connection is cropped. The cropped shortcut connection is added to the output of the second 1×1 convolutional layer to form the output of the entire residual unit. This simple operation eliminates the effect of feature padding in the residual unit.

The role of the downsampling residual unit, as shown in Figure 4(c), is to reduce the size of the output feature maps to one-half of the size of the input while doubling the number of feature channels. It uses a convolution with a stride of 2 in the 3×3 convolution layer to achieve downsampling. In this subsection, the nonpadding downsampling residual unit is designed to be similar to the corresponding general residual unit. As shown in Figure 4(d), first, the stride size of the 3×3 convolution layer is modified to 1, and the padding is removed. At the same time, after the cropping operation is performed, the stride of the shortcut connection convolution layer is modified to 1. The processed shortcut connection is added to the output of the second 1×1 convolution layer. Thus far, the feature padding of the downsampling residual unit has also been removed. However, in order to achieve downsampling, a max pooling layer with a stride of 2 is applied to the output of the residual unit, and the size of the output is kept as half of the size of the input.



(a) original (b) NPRU (c) original (d) NPRU-D

Figure 4. The proposed NPRU. (a) and (b) are the original residual unit and nonpadding residual unit NPRU, while (c) and (d) are the original downsampling residual unit and nonpadding downsampling residual unit NPRU-D. The NPRU removes the padding of the middle 3×3 convolutional layer and crops the shortcut connection to ensure that the two outputs are the same size. NPRU-D removes the padding of the 3×3 convolutional layer while modifying its network stride to 1. Correspondingly, the stride of the shortcut connection is also modified to 1, and the cropping operation is performed. Finally, we add a max pooling layer with a stride of 2 to achieve downsampling.

In summary, the designed NPRU and NPRU-D not only keep the output size unchanged but also allow each pixel of the input to participate in the feature computation. No information of any pixel is lost despite the removal of feature padding, and the invariance of feature mapping is guaranteed.

3.2 Network Architectures

This subsection applies the NPRU designed in the previous subsection to build a deeper network, named ResNet22. Table 1 shows the detailed structure of ResNet22. First, according to the suggestion in [1], we chose to build a three-stage network with a stride of 8, consisting of 22 weighted convolutional layers. In the first stage, according to the original ResNet, downsampling was performed using conv1 and max pool with a stride of 2 to make the network stride reach 4. The second stage includes 3 NPRUs with a stride of 1, and the third stage includes one NPRU-D and 3 NPRUs, increasing the total network stride to 8. When downsampling the feature map size, the number of filters doubles to improve feature discriminability. The size of the output feature map is 6×6, and each feature map pixel receives a signal from a region of size 93×93 pixels on the input image plane.

Table 1. DHSiam backbone network architecture

Layer	Kernel size	EFS	SFS
Input		127×127×3	255×255×3
conv1	7×7,64,s2,p3	64×64×64	128×128×64
max pool	2×2,64,s2,p0	32×32×64	64×64×64
conv2_1	1×1,64,s1,p0	32×32×64	64×64×64
	3×3,64,s1,p0	30×30×64	62×62×64
	1×1,256,s1,p0	30×30×256	62×62×256
conv2_2	1×1,64,s1,p0	30×30×64	62×62×64
	3×3,64,s1,p0	28×28×64	60×60×64
	1×1,256,s1,p0	28×28×256	60×60×256
conv2_3	1×1,64,s1,p0	28×28×64	60×60×64
	3×3,64,s1,p0	26×26×64	58×58×64
	1×1,256,s1,p0	26×26×256	58×58×256
conv3_1	1×1,128,s1,p0	26×26×128	58×58×128
	3×3,128,s1,p0	24×24×128	56×56×128
	1×1,512,s1,p0	24×24×512	56×56×512
max pool	2×2,512,s2,p0	12×12×512	28×28×512
conv3_2	1×1,128,s1,p0	12×12×128	28×28×128
	3×3,128,s1,p0	10×10×128	26×26×128
	1×1,512,s1,p0	10×10×512	26×26×512
conv3_3	1×1,128,s1,p0	10×10×128	26×26×128
	3×3,128,s1,p0	8×8×128	24×24×128
	1×1,512,s1,p0	8×8×512	24×24×512
conv3_4	1×1,128,s1,p0	8×8×128	24×24×128
	3×3,128,s1,p0	6×6×128	22×22×128
	1×1,512,s1,p0	6×6×512	22×22×512
OFS		6×6×512	22×22×512
Cross correlation Eq.1			
RMS		17×17×1	

Except for the 7×7 convolutional layer conv1, all other convolutional layers are NPRUs, where conv3_1 is an NPRU-D. EFS represents the feature map size of the exemplar image z, and SFS represents the feature map size of the search image x. OFS represents the output feature map size of the ResNet22 backbone network. RMS represents the response map size.

3.3 Layerwise Aggregation

Using a deep network such as ResNet22, it becomes possible to aggregate different deep layers. Intuitively speaking, object tracking requires rich representations with layers from low to high, scales from small to large, and resolutions from fine to coarse. Even if there are deep features in the convolutional network,

an isolated layer is not enough to handle this variety of representations. Concatenating and aggregating these representations improves the inference of recognition and localization. In previous work that only utilizes shallow networks such as AlexNet, multilevel features cannot provide a wide variety of representations. However, considering that the receptive field varies widely, it makes more sense to aggregate the different layers in ResNet22. Features in earlier layers mainly represent low-level information, such as texture and color, which are necessary for localization but lack semantic information. Features from later layers have rich semantic information, which is beneficial in some challenging scenarios, such as motion blur and deformation. Using this rich hierarchical information will help in tracking.

In the network of this paper, we aggregate multilevel layers, extract multibranch features to collaboratively infer target location, and provide a more comprehensive description of the target to achieve efficient tracking. As shown in Figure 5, multilevel features extracted from three branches of conv1, conv2_3, and conv3_4 are used for multiscale feature map aggregation. For all selected layers, the feature map generated by the exemplar image is used as a convolution kernel and the convolution operation is performed with the feature map generated by the search image to generate a response map. Convolutions for different layers use different strides, for the conv1 layer, a convolution with a stride of 4 is performed, and the generated response map is $F_1(x)$; for the conv2_3 layer, a convolution with a stride of 2 is performed, and the generated response map is $F_2(x)$; for the conv3_4 layer, a convolution with a stride of 1 is performed, and the generated response map is $F_3(x)$. All response maps are the same size, and the final $F(x)$ is obtained by weighting the sum of these three outputs:

$$F(x) = w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) \tag{3}$$

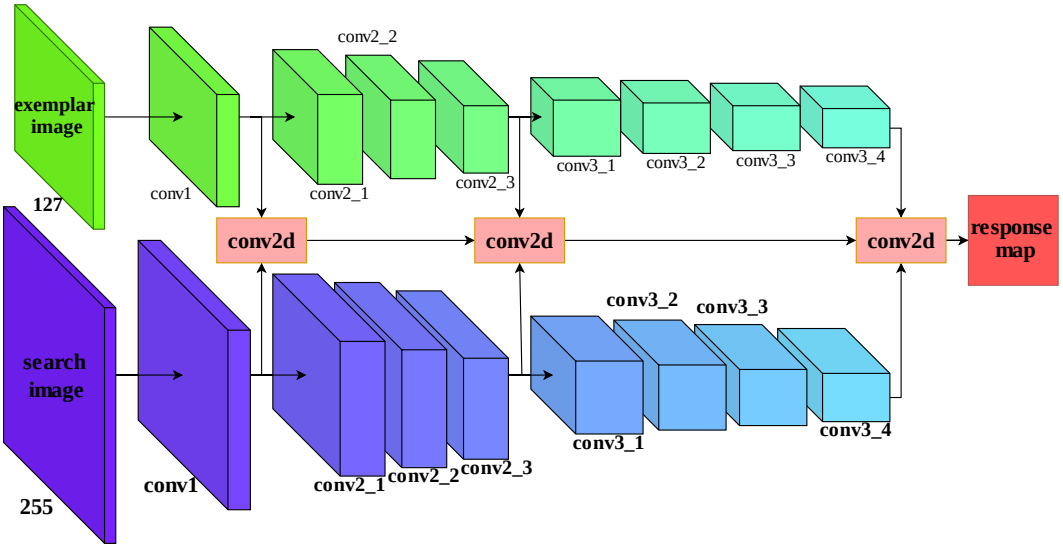


Figure 5. DHSiam architecture.

4 Experiments and Analysis

4.1 Experimental Datails

Training. Initialization using pretrained weights on ImageNet has proven to be an effective way to improve performance. The parameters of the ResNet22 network are also initialized in this way. First, ResNet22 is used as a classifier to train on ImageNet. Then, the pretrained parameters are used as the initial weights when training for object tracking tasks. We use image pairs collected from the ImageNet VID dataset for

training. During training, we freeze the weights of the conv1 layer and gradually fine-tune the weights of the other layers from back to front. The learning rate was reduced logarithmically from 10^{-3} to 10^{-7}. The weight decay is set to 10^{-4}, and the momentum is set to 0.9. The size of the exemplar image is 127×127 pixels, and the size of the search image is 255×255 pixels; all them are RGB images.

Evaluation. This paper uses the OTB and VOT datasets to evaluate the performance of trackers. OTB50 contains 50 real-world sequences collected from commonly used tracking sequences, and OTB100 adds another 50 sequences on this basis. These sequences have 11 interference attributes and two evaluation metrics, i.e., precision plots and the area under curve (AUC) of success rate plots. Precision refers to the percentage of video frames in which the Euclidean distance between the center point of the bounding box estimated by the tracking algorithm and the center point of the ground-truth bounding boxes is less than a given threshold. This paper uses a threshold of 20 pixels to evaluate the performance of these algorithms. Setting different thresholds results in different percentages, which can be plotted on a curve called a precision plot.

If the target position obtained by the tracking algorithm is *a* and the ground-truth label is *b*, the overlap rate (OS) is defined as:

$$\mathrm{OS} = \frac{|a \cap b|}{|a \cup b|} \tag{4}$$

When the OS of a certain frame is greater than a given threshold, this frame is considered as successful, and the percentage of successful frames is the success rate. Setting different thresholds results in different success rates. Therefore, a curve can be drawn, which is called a success plot. The area enclosed by this curve and the coordinate system is the AUC of the success rate plots.

For the VOT dataset, the evaluation is performed by the official toolkit. VOT16 contains 60 sequences with different challenge attributes, while VOT17 replaces 10 sequences in VOT16 and recalibrates the ground-truth label to the pixel level. The VOT dataset includes three evaluation metrics, namely, accuracy (A), robustness (R), and expected average overlap (EAO). Accuracy describes the average of the OS mentioned above. When the OS is 0, this indicates failure. Robustness refers to the average failure rate. The EAO is the expected nonreset overlap for a short-term image sequence, which is the most important metric for evaluating the performance of a tracking algorithm.

4.2 Ablation Study

To explore the influence of the ResNet22 backbone network and different aggregation modes on tracking performance, first, an ablation study is performed. We replace the backbone network of SiamFC with ResNet22 to create SiamFC-ResNet22. In addition, we aggregate the outputs of conv1, conv2_3, conv3_4 to create DHSiam. We aggregate the outputs of conv1 and conv3_4 to create DHSiam1. Similarly, DHSiam2 is the aggregated output of conv2_3 and conv3_4. We compare these four trackers with the baseline tracker SiamFC on the OTB100 dataset.

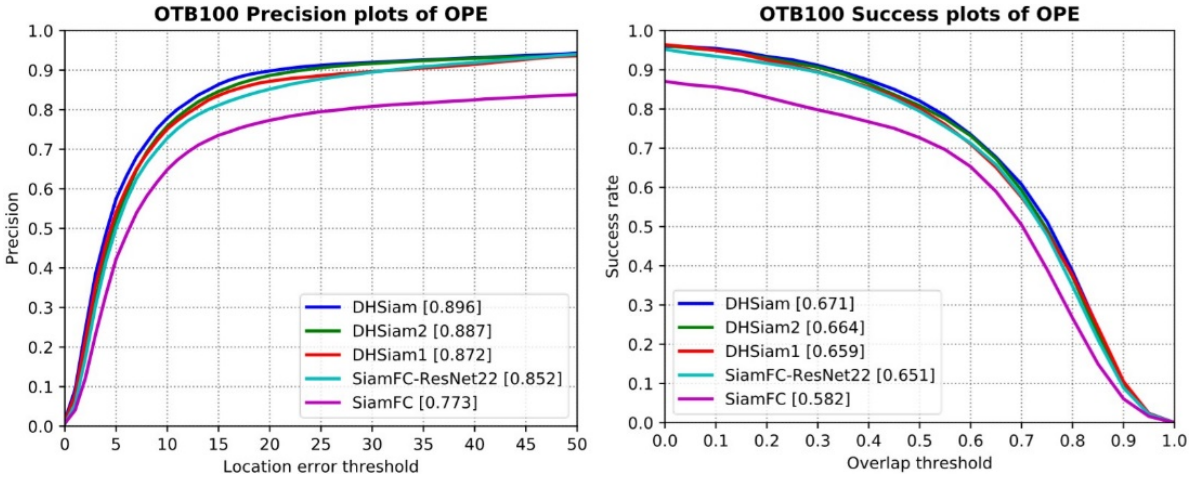


Figure 6. Performance comparison on OTB100. The figure shows the performance of the tracker for different aggregation modes, including the precision plot (left) and the success plot (right). The figure shows that simply replacing the backbone network with ResNet22 brings improvements of 10.2% and 11.9% in precision and AUC, respectively. Compared with non-aggregation, DHSiam1 improved precision and AUC by 2.3% and 1.2%, respectively. DHSiam2 improved precision and AUC by 4.1% and 2.0%. Notably, DHSiam improved precision and AUC by 5.2% and 3.1%.

Figure 6 shows the performance of the five trackers. The figure shows that the backbone network ResNet22 designed in this paper can bring huge performance improvements to the tracker, and the aggregation of multilevel feature maps can also obtain considerable benefits. Finally, compared with the baseline tracker SiamFC, the DHSiam designed in this paper achieved precision and AUC improvements of 15.9% and 15.3%, respectively.

4.3 Comparison to State-of-the-Art Trackers

To validate the effectiveness of the algorithm proposed in this paper, we compared it with several other state-of-the-art trackers. These trackers can be roughly divided into three categories: (1) deep learning trackers, including ECO [16], MDNet [17], and CFNet [6]; (2) correlation filter trackers, including Staple [18], PTAV [19], KCF, and SAMF; (3) trackers using single or multiple online classifiers, with MEEM as a representative algorithm. Table 2 lists the performance of these trackers on four benchmark datasets, including OTB50, OTB100, VOT16 and VOT17. Figure 7 shows the precision plots and success plots of all 10 trackers mentioned above on OTB100.

On the OTB50 and OTB100 datasets, DHSiam achieved the most outstanding performance after ECO and MDNet. However, ECO can only run at 6fps and MDNet at only 1fps, while DHSiam can reach 63fps on a single GeForce GTX 1080 GPU, which far exceeds the 24fps required for real-time tracking. As shown in Figure 8, DHSiam has the best performance among trackers that can run at real-time tracking speed. Especially in performance on the OTB100 dataset, DHSiam is not only twice as fast as the second-best tracker PTAV among those that can run in real-time, but its AUC is 5.67% higher. On the VOT16 dataset, DHSiam surpassed MDNet and achieved second place. On the VOT17 dataset, DHSiam is second only to ECO.

Table 2. **Performance comparison of state-of-the-art trackers on four benchmark datasets**

Tracker	OTB50		OTB100		VOT16			VOT17			FPS
	Prec.	AUC	Prec.	AUC	A	R	EAO	A	R	EAO	
ECO	0.874	0.643	0.910	0.691	0.55	0.20	0.375	0.483	0.276	0.280	6
MDNet	0.890	0.645	0.909	0.678	0.54	0.34	0.257	-	-	-	1
CFNet	0.803	0.603	0.789	0.598	-	-	-	-	-	-	75
Staple	0.801	0.600	0.784	0.581	0.54	0.38	0.295	0.530	0.688	0.169	80
PTAV	0.887	0.633	0.851	0.635	-	-	-	-	-	-	25
KCF	0.611	0.403	0.696	0.477	-	-	-	0.477	0.773	0.135	172
SAMF	0.650	0.469	0.751	0.553	-	-	-	-	-	-	7
MEEM	0.712	0.473	0.781	0.530	-	-	-	0.463	0.534	0.192	10
SiamFC	0.809	0.607	0.773	0.582	0.53	0.46	0.235	0.502	0.585	0.188	86
DHSiam	0.883	0.638	0.896	0.671	0.54	0.38	0.327	0.517	0.493	0.233	63

Red, green, and blue fonts represent the top three trackers.

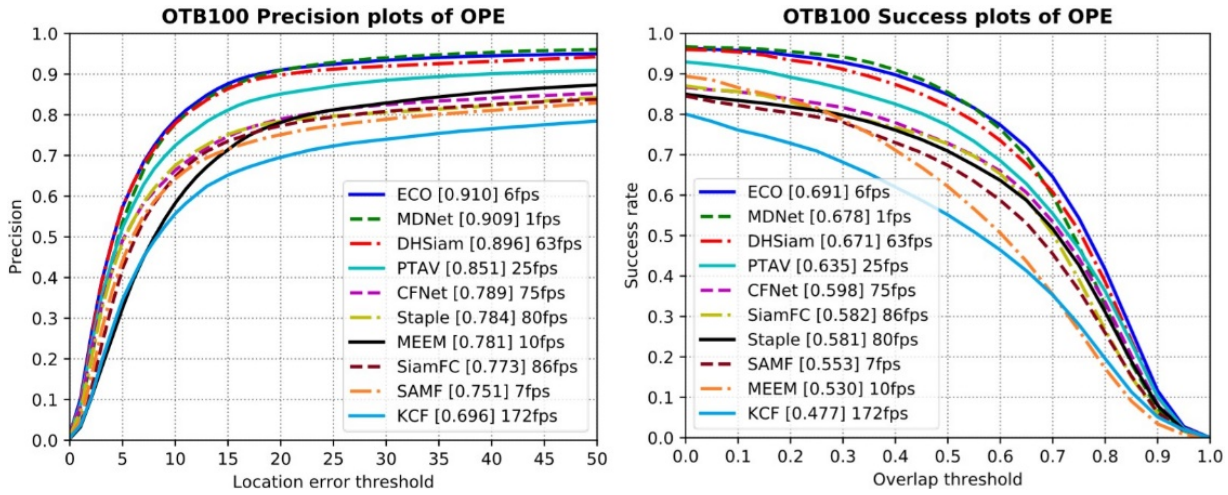


Figure 7. Comparison of state-of-the-art trackers on OTB100. The figure shows that DHSiam achieved very similar performance to ECO and MDNet in terms of precision (left) and success rate (right), which is far better than that of non-deep learning trackers. In particular, the DHSiam tracker can run at 63 fps, which is far better than ECO and MDNet, and can be used in real-time tracking scenarios.

In summary, DHSiam achieved very competitive performance on multiple benchmark datasets. As shown in Figure 8, DHSiam achieved the best performance among trackers that can run at real-time tracking speed.

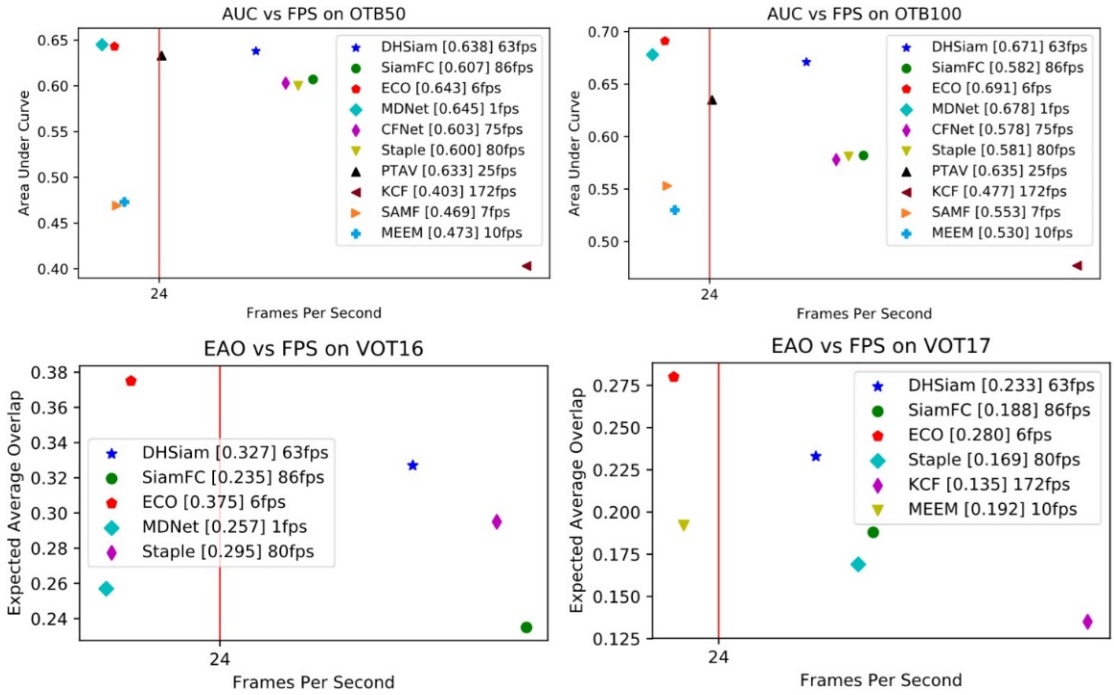


Figure 8. Performance vs. FPS on OTB and VOT. Trackers to the right of the red line can run in real-time. Among the trackers that can run in real-time, DHSiam (blue star) achieved the best performance.

5 Conclusion

This paper proposes a new visual object tracking algorithm based on a Siamese network. Compared with previous algorithms, we designed a deeper network architecture for Siamese trackers, which can extract more advanced semantic information about the target object and improve the generalization ability of the tracker. We also adopt layerwise aggregation to extract multilevel features to collaboratively infer the target location and to provide a more comprehensive description of the target object. Experimental results demonstrate the advanced abilities of our tracker.

REFERENCES

[1]. Zhang Z, Peng H. Deeper and wider siamese networks for real-time visual tracking[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 4591-4600.

[2]. Tao R, Gavves E, Smeulders A W M. Siamese instance search for tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 1420-1429.

[3]. Bertinetto L, Valmadre J, Henriques J F, et al. Fully-convolutional siamese networks for object tracking[C]//European conference on computer vision. Springer, Cham, 2016: 850-865.

[4]. Wang Q, Teng Z, Xing J, et al. Learning attentions: residual attentional siamese network for high performance online visual tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4854-4863.

[5]. Held D, Thrun S, Savarese S. Learning to track at 100 fps with deep regression networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 749-765.

[6]. Valmadre J, Bertinetto L, Henriques J, et al. End-to-end representation learning for correlation filter based tracking[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 2805-2813.

[7]. Guo Q, Feng W, Zhou C, et al. Learning dynamic siamese network for visual object tracking[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 1763-1771.

[8]. Fan H, Ling H. Siamese cascaded region proposal networks for real-time visual tracking[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 7952-7961.

[9]. Zhu Z, Wang Q, Li B, et al. Distractor-aware siamese networks for visual object tracking[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 101-117.

[10]. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[11]. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[12]. Kong T, Yao A, Chen Y, et al. Hypernet: Towards accurate region proposal generation and joint object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 845-853.

[13]. Kim K H, Hong S, Roh B, et al. Pvanet: Deep but lightweight neural networks for real-time object detection[J]. arXiv preprint arXiv:1608.08021, 2016.

[14]. Wang L, Ouyang W, Wang X, et al. Visual tracking with fully convolutional networks[C]//Proceedings of the IEEE international conference on computer vision. 2015: 3119-3127.

[15]. Wang N, Li S, Gupta A, et al. Transferring rich feature hierarchies for robust visual tracking[J]. arXiv preprint arXiv:1501.04587, 2015.

[16]. Danelljan M, Bhat G, Shahbaz Khan F, et al. Eco: Efficient convolution operators for tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 6638-6646.

[17]. Nam H, Han B. Learning multi-domain convolutional neural networks for visual tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 4293-4302.

[18]. Bertinetto L, Valmadre J, Golodetz S, et al. Staple: Complementary learners for real-time tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 1401-1409.

[19]. Fan H, Ling H. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 5486-5494.