Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-14519-6

Real-time object tracking in the wild with Siamese network

Feng Han[1,2] · Shaokui Jiang[1,2] · Jianmin Wu[3] · Baile Xu[1,2] · Jian Zhao[4] · Furao Shen[1,5]

Received: 20 July 2021 / Revised: 18 March 2022 / Accepted: 31 January 2023

Abstract
Single object tracking (SOT) is one of the most important tasks in computer vision. With the development of deep neural networks and the release for a series of large scale datasets for single object tracking, Siamese networks have been proposed and perform better than most of the traditional methods. However, recent Siamese networks are getting slower to obtain better performance as they become deeper. Most of those networks could only meet the needs of real-time object tracking in ideal environments. In order to achieve a better balance between efficiency and accuracy, we propose a simpler Siamese network for single object tracking, which runs fast in poor hardware configurations while remaining an acceptable accuracy. The proposed method consists of three parts: sample generation, SE-Siamese and regression localization. In the sample generation stage, template patch and detection patch are cropped from the selected video frames in a new way. The SE-Siamese subnetwork adopts Siamese network and Squeeze-and-Excitation (SE) network as the feature extractor which is an effective way of speeding up the training phase. The regression localization network aims to compute the location of the tracked object in a more efficient way without losing much precision. To validate the effectiveness of the proposed approach, we conduct extensive experiments on several challenging tracking benchmark datasets, including VOT2015, VOT2016, VOT2017 and OTB-100. The experimental results show that our approach displays significant speed improvements compared to several strong baseline trackers (19.5 FPS to 44.4 FPS).

Keywords Real-time object tracking · Siamese network · Squeeze-and-excitation network · Regression localization

1 Introduction

Object tracking is one of the main applications of computer vision. Single object tracking is fundamental to tracking tasks, which has a totally different methodology with multiple

✉ Furao Shen
frshen@nju.edu.cn

Extended author information available on the last page of the article.

object tracking. Object tracking has been widely used in video surveillance [42], athlete's performance analysis [38], human-machine interaction, image understanding, autonomous driving [2] and so on. It also plays an important role in other related missions. For example, object detection is a relatively more complex application in computer vision, where even state-of-the-art could miss some objects in specific scenes. In that case, object tracking could help. Some missing targets caused by occlusion or light changing could be detected by the assistance of tracking. Furthermore, detecting objects in each frame of a video is time-consuming and unnecessary, whereas tracking detected object could save a lot of time. However, recent tracking algorithms tend to trade speed for accuracy, which deviates from the original intention of target tracking. In some practical applications, the running speed is very important [36, 37].

Generally, the algorithms for object tracking can be divided into two categories. The first category is the generative method, which models the target in the first frame and searches the most similar area in the following frames. Many traditional methods belong to this category, such as Kalman filter [5], particle filter [51] and mean-shift [7]. These methods are easy to implement, and effective in simple scenarios. However, these methods would fail to track the targets when the appearances of the targets or the surrounding light conditions have changed during tracking. Beyond that, some generative methods are not efficient enough for real-time tracking. For example, particle filters needs to generate more and more particles to search the best matching point, and the computing time increases along with the number of particles.

The second category is the discriminative method, which regards the tracked targets as positive samples while the backgrounds as negative. Classifiers are trained to separate the tracked target from backgrounds. A discriminative method using correlation filtering [6] brought the performance of single object tracking to a new level. However, correlation filtering is unstable in tracking, and can not accurately identify objects with obvious changes in appearance. Some recent works use neural networks for object tracking, and solve this problem well. However, most state-of-the-art trackers are not fast enough for real-time tracking, especially in some non-ideal environments. Some trackers achieve high speed and accuracy due to the clever designs, such as a series of trackers using Siamese network. However, training these networks is really a complicated process. For SiamRPN [27], the datasets used for training include a huge one consists of 240000 video clips. As for DaSiamRPN [55], it introduces COCO [30] and ImageNet-DET [41] in order to enrich the training categories. These huge datasets make training process slow, which means it needs more time and more tricks to obtain an excellent model.

In this paper, we focus on designing an accurate single object tracker that runs fast under the condition of limited hardware resources. Firstly, we randomly crop detection patch and template patch from the original video frame as our new training samples. The reconstructed training samples help to avoid overfitting and improve convergence speed of the network. Next, we add the Squeeze-and-excitation (SE) [19] block into the feature extractor as a subnetwork called SE Layer. The SE Layer enables the network to train a suitable number of parameters faster and more precisely. Afterwards, the feature maps of video frames are convolved by the kernels consists of the features of the tracked target. Different from other Siamese networks [4], the convolution result of our work is smaller. By this means the number of parameters in this layer is reduced, and hence the training and inference processes are greatly accelerated. Finally, we get the output by putting the feature map into our regression part, which consists of a kernel size of 1×1 and a fully connected layer. With the output and the size of the search area, we locate the tracked target.

The main contributions of this work can be summarized as follows:

1. We propose a new training sample generation method, similar to a data enhancement method, which can effectively avoid overfitting in the Siamese network training process.
2. We propose a tracking network model named SE-Siamese, where the SE module can focus on important feature map, while reducing the number of parameters, speeding up training and reasoning.
3. We use regression localization instead of calculating multiple rough position, which is a fast and efficient method.

We train the network using only two datasets, GOT [21] and ImageNet-VID [41], which consists of no more than 15000 video clips in total. The network is well-trained within 5 training epochs. We evaluate our method on VOT2015 [23], VOT2016 [24], VOT2017 [22] and OTB100 [49] challenges. The results show our proposed method achieves an excellent running speed on low-end hardware and a comparable performance.

The rest of this paper is organized as follows. In Section 2, we introduced the work that is related to ours. Next, in Section 3 we introduce our proposed model. In Section 4, we describe an experimental validation of our approach. Finally, Section 5 concludes this work.

2 Related work

Image augmentation Image augmentation is an important technique to improve the generalization performance of models in various computer vision tasks such image classification [15], object detection [40], and object tracking [16, 55]. The previous works usually use horizontal and vertical flipping, rotation, and random cropping to generate similar but distinct training examples. All of these methods focus on augmenting more training samples to expand the size of the training set but do not consider time consumption. In contrast to previous works, we introduce random cropping in detection patch generation as a sample generation method, which allows a good trade-off between speed and accuracy for visual object tracking.

Correlation filter-based trackers MOSSE (Minimum Output sum of Squared Error filter) proposed by Bolme [6] is the first work that introduces correlation filter to object tracking. MOSSE started the next 10 years research boom for correlation filter in this task. Inspired by MOSSE, in the year of 2015, Henriques et al. [17] put forward KCF (Kernelized Correlation Filters) algorithm. KCF replaces the sliding windows with cyclic shifts, which enables the calculation be done in Fourier space. KCF speeds up the algorithm greatly. In addition, KCF uses ridge regression as the loss function. The ridge regression in linear spaces is simplified by projecting the calculation to non-linear spaces using kernel functions. More works about kernelized correlation filters, such as DSST [9], used scaling pool to solve the problem of target shift because of the changing size in KCF. Liu et al. [31] proposed part-based visual tracking via adaptive correlation filters.

Siamese network-based trackers After Alexnet [25] was designed in 2012, deep learning was widely used in various fields, including object tracking. GOTURN (Generic Object Tracking Using Regression Networks) [16] proposed by Held et al. adopts Siamese network for feature extraction. After the feature extraction, GOTURN contacts the features

directly for the regression in next steps. GOTURN runs fast because of the simple architecture. SiamFC [4] proposed by Bertinetto et al. is similar to the GOTURN in the structure of feature extractor, but SiamFC put forward to convolve template features on the features of search area, instead of stacking them together. SiamFC achieves state-of-the-art performance in multiple benchmarks, which exploits the power of deep convolutional networks for object tracking. After SiamFC, there are a lot of works in object tracking with correlation filtering and Siamese network, such as the works by Zhang et al. [53] and Li et al. [28].

Attention mechanism The attention mechanism in deep learning has been widely used in RNN and CNN networks. The visual attention model is trying to let the neural network be able to "focus" its "attention" on the interesting part of the image where it can get most of the information, while paying less "attention" elsewhere. Attention mechanism is usually implemented in channel-wise, spatial-wise, or temporal-wise manners. Channel-wise attention models reweight activations in different channels to enhance the most meaningful channels and weaken the others using the attention weights. As a remarkable work of channel-wise attention model, Squeeze-and-Excitation (SE) [19] block can be stacked together with any multi-channel layers, assigning different channels with different weights. The SE block brings significant improvements for existing state-of-the-art CNNs with little computational cost. Spatial-wise attention models address the inter-spatial relationship among features. Convolutional block attention module (CBAM) [48] adopts both channel-wise and spatial-wise attention to modulate the feature maps. On the other hand, non-local networks [47] takes spatial and temporal information into account simultaneously to capture long-range image features dependency. Compared with SE blocks, both CBAM and non-local network increase the number of parameters and the computational complexity. Owing to the reasons above, the SE block is more beneficial for real-time applications.

Region proposal network in trackers SiamRPN [27] points that SiamFC does not regress the position of the tracked target totally. Instead, SiamFC calculates the result using the rough position from heat map and multiple shape. This is one of the reasons why the accuracy of SiamFC was not high enough. SiamRPN makes a better use of the Siamese network in object tracking, introducing the region proposal network [40] into object tracking for replacing the part after feature extractor. SiamRPN and other related papers, DaSiamRPN [55] and SiamRPN++ [26], all achieve excellent performance.

3 Proposed approach

In this section, we explain our approach in detail.

3.1 Overall framework

The framework of our model is illustrated in Fig. 1. The framework consists of three main components, including sample generation, SE-Siamese and regression localization. First, using our proposed sample generation method to select frames from the video sequence and crop them to generate fixed-size template patch and detection patch. Next, we feed the template patch and the detection patch from each frame of the video into the SE-Siamese network. The SE-Siamese network is a combination of convolution neural network (CNN), SE block and correlation operator. The SE block gives different channels their corresponding weights. The correlation operator simulates the progress of searching the tracked object

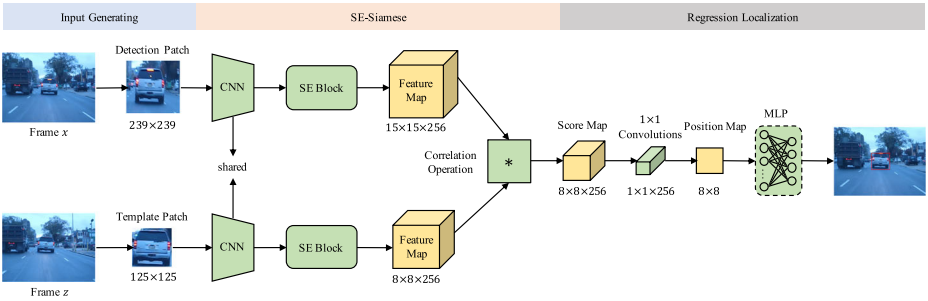

Fig. 1 This is the framework of our network. The CNN feature extractor takes RGB images as input in both branches with shared parameters. After extracting, the feature maps are fed into the SE block. The SE block adjusts the features extracted in different channels. The template features after extracting and weighting are $w_z \times w_z \times c$ in shape, the detection features from the same way are $w_x \times w_x \times c$. $*$ denotes correlation operator, which works for every channel and generates a feature map of $(w_x - w_z + 1) \times (w_x - w_z + 1)$ with c channels. Finally, the regression part turns the feature map to the position of the target in the detection frame. The output is the relative position for the target exactly

in a video frame. The SE-Siamese network outputs the relative position of the target in the detection patch. Finally, we combine the convolution kernel and fully connected layer to turn the result comes from the last step into a vector that outputs the target region.

3.2 Sample generation

The steps of sample generation in the training phase and the inference test phase are not exactly the same, so we will introduce them separately.

Training phase For each video sequence, there are annotations of object positions in each frame. We process every sequence and corresponding annotation using the five steps as shown in Fig. 2:

1. Select two frames with an interval randomly selected from 1 to 100. If the target is partly or entirely out of view in either frame, redo this step.
2. Crop the target in one frame as the template patch.
3. For another frame, crop the image as the detection patch according to $left, top, right, bottom$. These values are obtained randomly between the bounding box and the given edge $\hat{left}, \hat{top}, \hat{right}, \hat{bottom}$, just as

$$\begin{cases} left = \text{random}(\hat{left}, x_1) \\ top = \text{random}(\hat{top}, y_1) \\ right = \text{random}(x_2, \hat{right}) \\ bottom = \text{random}(y_2, \hat{bottom}) \end{cases}. \tag{1}$$

As for the edge, it is selected randomly from the (2), (3) and (4), which is also illustrated in Fig. 3,

$$\begin{cases} \hat{left} = \max(0, x_1 - w_b) \\ \hat{right} = \min(x_2 + w_b, w_f) \\ \hat{top} = \max(0, y_1 - h_b) \\ \hat{bottom} = \min(y_2 + h_b, h_f) \end{cases}, \tag{2}$$

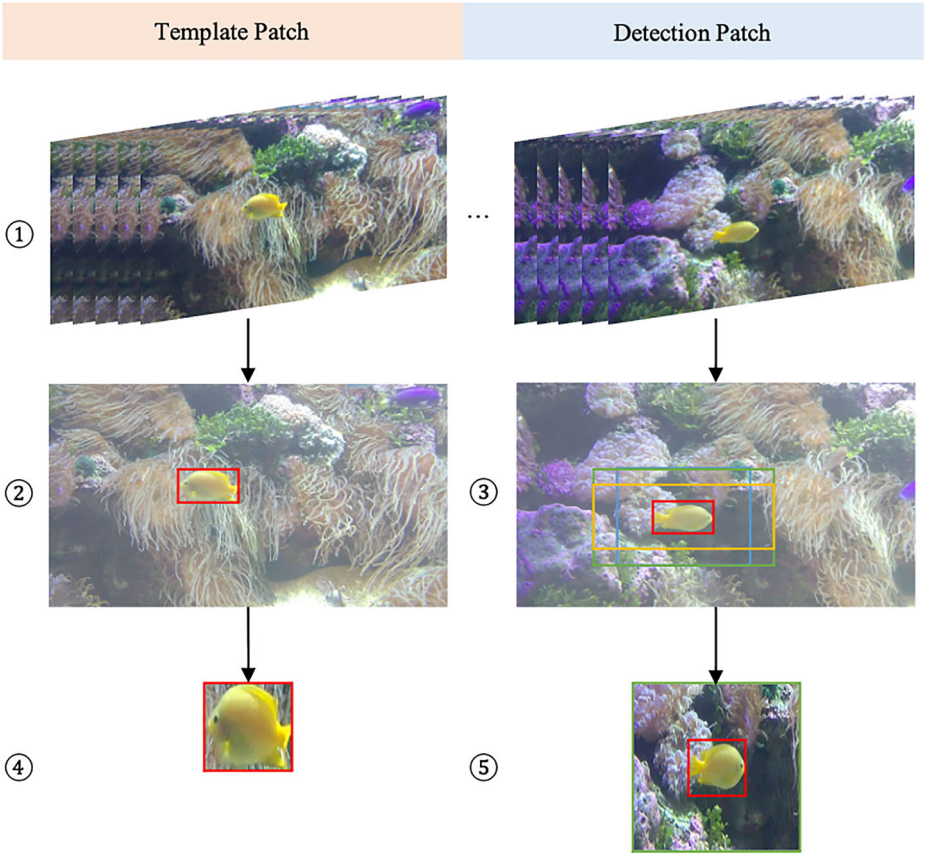

Fig. 2 Illustration of steps for generating template patches and detection patches from sequences of datasets

$$\begin{cases} \hat{left} = \max(0, x_1 - w_b/2) \\ \hat{right} = \min(x_2 + w_b/2, w_f) \\ \hat{top} = \max(0, y_1 - h_b) \\ \hat{bottom} = \min(y_2 + h_b, h_f) \end{cases}, \tag{3}$$

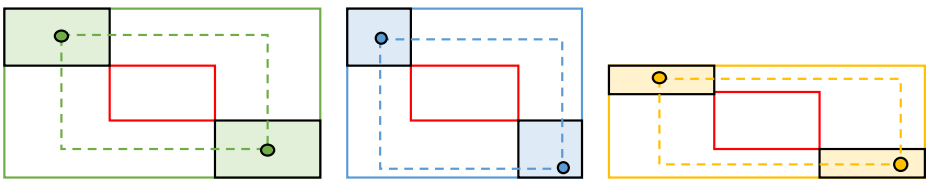

Fig. 3 Detection patch generation methods

$$\begin{cases} \hat{left} = \max(0, x_1 - w_b) \\ \hat{right} = \min(x_2 + w_b, w_f) \\ \hat{top} = \max(0, y_1 - h_b/2) \\ \hat{bottom} = \min(y_2 + h_b/2, h_f) \end{cases}, \tag{4}$$

where x_1, y_1, x_2, y_2 denote the x-axis coordinate, y-axis coordinate of the top-left corner and the bottom-right corner of the target bounding box in current frame respectively. w_b and h_b denote the width and height of the target in current frame respectively. w_f denotes the width of the current frame while h_f denotes the height.

4. Resize the cropped detection patch in x, y direction separately.
5. Calculate the label $[O_1^\star, O_2^\star, O_3^\star, O_4^\star]$ for this sample as

$$\begin{cases} O_1^\star = \frac{\hat{x_1} - \frac{\hat{w_f}}{2}}{\hat{w_f}} = \frac{\hat{x_1}}{\hat{w_f}} - \frac{1}{2} \\ O_2^\star = \frac{\hat{y_1} - \frac{\hat{h_f}}{2}}{\hat{h_f}} = \frac{\hat{y_1}}{\hat{h_f}} - \frac{1}{2} \\ O_3^\star = \frac{\hat{x_2} - \frac{\hat{w_f}}{2}}{\hat{w_f}} = \frac{\hat{x_2}}{\hat{w_f}} - \frac{1}{2} \\ O_4^\star = \frac{\hat{y_2} - \frac{\hat{h_f}}{2}}{\hat{h_f}} = \frac{\hat{y_2}}{\hat{h_f}} - \frac{1}{2} \end{cases}, \tag{5}$$

where $\hat{x_1}$, $\hat{y_1}$, $\hat{x_2}$, $\hat{y_2}$ denote the x-axis coordinate, y-axis coordinate of the top-left corner and the bottom-right corner of the target bounding box in the cropped image, which is the detection patch, respectively. $\hat{w_f}$ denotes the width of the detection patch while $\hat{h_f}$ denotes the height.

Once we generate enough samples, we can start training the model.

Inference phase In the inference phase, we take the template patch and the detection patch as input, the output of the model is $[O_1, O_2, O_3, O_4]$ mentioned above. Then, we calculate the position according to the output and relevant frames. The detailed steps are as follows:

1. Crop the target in the first frame as the template patch, the template patch is fixed till the tracking progress ends. Record the position of template patch as $[left^\star, top^\star, width^\star, height^\star]$.
2. Crop area according to $left, top, right, bottom$ of the current frame as detection patch by:

$$\begin{cases} left = \max(0, left^\star - width^\star * \delta) \\ top = \max(0, top^\star - height^\star * \delta) \\ right = \max(left^\star + width^\star * (1 + \delta), w_f) \\ bottom = \max(top^\star + height^\star * (1 + \delta), h_f) \end{cases}. \tag{6}$$

3. Take the detection patch and the template patch into the tracker, and obtain the target region x_1, y_1, w_1, h_1 in the detection patch.
4. Considering the position of detection patch in the current frame, update the output as

$$\begin{cases} x_1 = x_1 + left \\ y_1 = y_1 + top \\ x_2 = x_2 + left \\ y_2 = y_2 + top \end{cases}. \tag{7}$$

5. Update the position in current frame as $[left^\star, top^\star, width^\star, height^\star]$.
6. Go back to step 2 till the tracking ends.

3.3 SE-Siamese

The SE-Siamese network is a combination of CNN for feature extraction, SE block and correlation operator.

The feature extractor has two branches with shared parameters: the template branch takes the image patch containing the target as input, which is denoted as z; the detection branch takes the patch x from a video frame as input. The layers in the feature extractor are adopted in both the template branch and the detection branch with shared parameters. These layers aim to extract the features that make the greatest response in two similar patches, regardless of which branch it is belonged to. We denote F as the operation of feature extraction, $F(z)$ as the output of the template branch, and $F(x)$ as the output of the detection branch.

The feature extractor of our work is alternative. It can be replaced by most of the backbones commonly used, such as AlexNet, ResNet [15] and so on. One thing we should notice is that the complexity of the backbone. According to our experiments, the more complex the backbone, the better the result for tracking. We found that ResNet18 has a better balance between the running speed and the accuracy. Therefore, we recommend you use ResNet18 for the feature extractor.

Inspired by the SE network, we add the block after the feature extractor. The SE block is one of the attention mechanism enabling feature recalibration. With SE block, our model performs better and converges faster.

The SE block consists of an average pool, fully connected layers and activation functions as shown in Fig. 4. The input of the SE block is the feature maps come out of the feature extractor, and its output keeps the same size with the input. We denote the above transformation as S, the output for both branches could be expressed as

$$\begin{cases} F_x = S(F(x)) \\ F_z = S(F(z)) \end{cases}, \tag{8}$$

where F_x denotes the output of the SE block in detection branch, while F_z denotes the one in template branch.

As mentioned earlier, the searching feature map is convolved by the kernel consists of the template feature. We make the convolution at every channel separately. The convolution process of the c-th channel is depicted as

$$\mathrm{conv_c}(F_x, F_z) = F_x^c * F_z^c, \tag{9}$$

where F_x^c denotes the c-th channel for F_x, while F_z^c denotes the c-th channel for F_z, $*$ denotes correlation operator.

The features from detection branch $F(x)$ after extracting and weighting is $w_x \times w_x \times c$ in shape, the template features $F(z)$ from the same way is $w_z \times w_z \times c$. The convolution works for every channel, the result after convolution is $(w_x - w_z + 1) \times (w_x - w_z + 1) \times c$, which contains the information of the target position in detection patch. The relative position based on the feature map is obtained by feeding the convolution result into the regression part.

3.4 Regression localization

In this section, we introduce the regression part in our framework. As we mentioned before, the regression part generates the relative position based on the feature map. This is the most

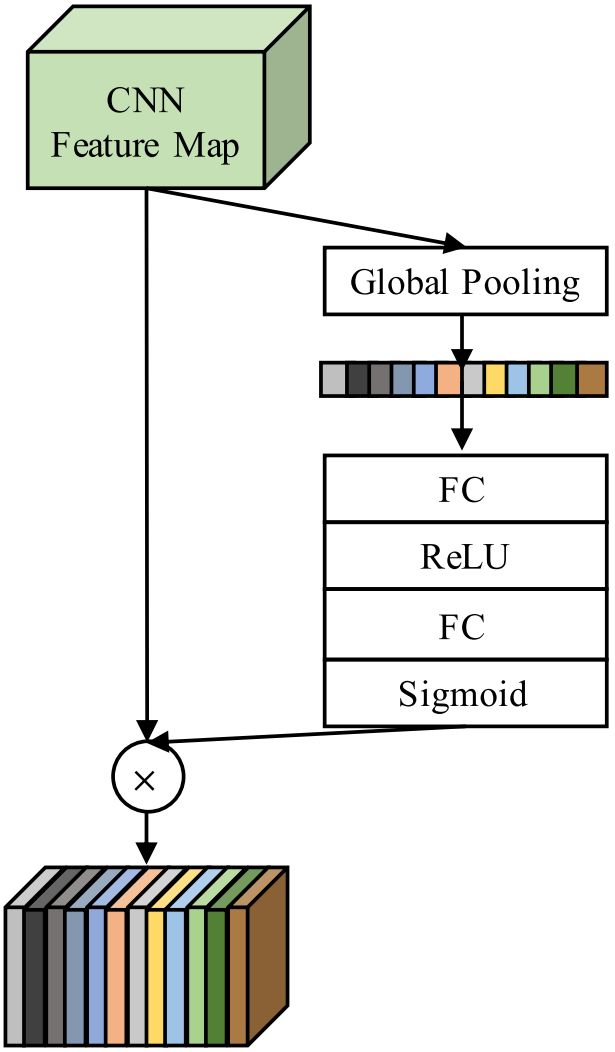

Fig. 4 The structure of the SE block, which contains the vector of the same length as the number of channels. We acquire this vector by training the sequences of activation functions and fully connected layers. The final feature map is obtained by multiplying the vector to relevant channels

important part in the tracking model, especially for the Siamese network. The regression part turns the feature map after convolution to the final position of the target.

In our network, we propose a brand new approach for regression, which is lightweight and quite simple. We use a convolution kernel of size 1×1 to train the weight for different channels. After this step, the shape of the result is $(w_x - w_z + 1) \times (w_x - w_z + 1)$. Then, we use a fully-connected subnetwork to connect the convolution result and the final position. Instead of regressing position directly, we use relative position just like most

networks do. The relative position is a vector containing four float numbers, which is denoted as $[O_1, O_2, O_3, O_4]$. The position of target in the detection patch is computed as

$$\begin{cases} x_1 = O_1 * w_f + w_f/2 \\ y_1 = O_2 * h_f + h_f/2 \\ x_2 = O_3 * w_f + w_f/2 \\ y_2 = O_4 * h_f + h_f/2 \end{cases}, \tag{10}$$

where x_1, y_1, x_2, y_2 denote the x-axis coordinate, y-axis coordinate of the top-left corner and the bottom-right corner of the target bounding box in the detection frame. h_f and w_f denote the height and width of the detection patch, respectively.

For bounding box regression, the objective function to train the network is defined as

$$\mathcal{L} = \sum_{i=1}^{n} \text{smooth}_{L_1}(O_i^\star - O_i, \sigma), \tag{11}$$

with a smooth L1 loss function defined as

$$\text{smooth}_{L_1}(x, \sigma) = \begin{cases} 0.5\sigma^2 x^2 & \text{if } |x| < \frac{1}{\sigma^2} \\ |x| - \frac{1}{2\sigma^2} & \text{otherwise} \end{cases}, \tag{12}$$

where $O_i^\star$ denotes the groundtruth of the target position.

4 Experiment and analysis

4.1 Experimental settings

We use 239×239, 125×125 for the shape of two branch inputs. We set w_x as 15 and w_z as 8. Training the model within 5 epoches with the learning rate of $1e-3$ and batch size of 80. During the testing phase, we set the value of δ in (6) as 0.67. We deploy the networks and test the speed on an NVIDIA GeForce 840M GPU.

4.2 Datasets

We trained our tracker with 16,000 training samples generated from GOT and ImageNet-VID, and then test the tracker on VOT and OTB100.

In the training phase, we select two datasets, GOT and ImageNet-VID, for generating samples of the detection frame and the template frame. GOT aims to provide a dataset with wide coverage of common objects for training visual object trackers. GOT covers 560 classes of moving objects and 87 motion patterns. There are more than 10000 video segments provided by GOT with 1.5 million labeled bounding boxes. ImageNet-VID provides annotations with 30 basic-level categories.

In the testing phase, we picked up the versions of 2015, 2016, 2017 from the VOT challenges. Considering the test set of VOT2019 containing the GOT dataset which is one of the datasets in our training phase, we remove the experiment on VOT2019 for fairness.

4.3 Evaluation metrics

We use accuracy, robustness, EAO and EFO for our indicators in VOT2015 and VOT2016. In VOT2017, we run the real-time experiment and a running speed test for some excellent trackers. We add the speed test because the real-time experience show no difference when

the speed of the trackers reach a certain value. For the comparative experimental results of other excellent trackers, we take the values for accuracy from the VOT reports to avoid the differences because of the implementation details.

Accuracy means the average overlap between the tracking result A_t^T and groundtruth A_t^G. The accuracy at time-step t could be calculated by

$$\phi = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}. \tag{13}$$

The tracker would be reset if the overlap becomes 0. To avoid a higher accuracy caused by the re-initialization, the initialization is executed 5 frames after the tracking failure and the accuracy of next 10 frames would be ignored.

The robustness measure indicates the average number of failures per sequence. The failure of N sequences could be defined as

$$\text{robustness} = \frac{1}{\sum l_i} \sum_{0}^{N-1} f_i, \tag{14}$$

where f_i and l_i denote the initiation times and length of the i-th sequence.

EAO (expect average overlap rate) is a new measure adopted in VOT2015, which is complicated to calculate. In short, we establish a curve about the length of a sequence and the raw accuracy $\hat{\Phi_{N_s}}$, where N_s is the length of sequences. $\hat{\Phi_{N_s}}$ could be obtained by averaging the overlaps Φ_{N_s} on a large set of N_s frames. The EAO $\hat{\Phi}$ is the expected curve value over an interval $[N_{lo}, N_{hi}]$, denoted as

$$\begin{cases} \Phi_{N_s} = \frac{1}{N_s} \sum_{i=1:N_s} \phi_i \\ \hat{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s = N_{lo}:N_{hi}} \hat{\Phi_{N_s}} \end{cases}. \tag{15}$$

EFO (equivalent filter operations) reduces the variability by dividing the operation time of filtering on an image with fixed shape, which is used for indicating the running speed from VOT2014 to VOT2016. But the filtering operation could not represent all the operations in the different platform and different programming languages, which means EFO could not eliminate the variability.

As for the OTB100, we draw the success plot and precision plot comparing with some other trackers, including some state-of-the-art trackers in these years and some fast traditional trackers. According to the precision plot and success plot, we can have a better evaluation for trackers.

Success plot is similar to the accuracy measure in VOT. The success plot is obtained by calculating the overlaps in the sequences by altering the overlap threshold.

Precision plot focuses on the average Euclidean distance between the center point of the predicted position and the annotated position. Given different thresholds, corresponding ration could be shown as a curve called precision plot. However, one disadvantage of this measure is the failure to reflect the change in target size and scale.

Finally, we compare the size of our network with some other Siamese tracking networks in the last subsection.

Table 1 Comparsion with top trackers in VOT2015

Tracker	Accuracy	Robustness	EAO	EFO
DeepSRDCF [10]	0.56	1.00	0.32	0.38
EBT [54]	0.47	1.02	0.31	1.76
SRDCF [10]	0.56	1.24	0.29	1.99
LDP [17]	0.51	1.84	0.28	4.36
sPST [20]	0.55	1.48	0.28	1.01
SC-EBT [45]	0.55	1.86	0.25	0.80
Ours	0.55	1.95	0.20	52.37

4.4 Experiments on VOT benchmarks

Results on VOT2015 The VOT2015 provides fully-annotated sequences for visual object tracking. These sequences cover various scenarios, most of which correspond to one or more difficult problems in tracking. We test our tracker on the VOT2015 based on the rules, other tracker results for comparison are taken from the VOT2015 report [23]. The methods used for comparison include DeepSRDCF [10], EBT [54], SRDCF [10], LDP [17], sPST [20] and SC-EBT [45].

As shown in the Table 1, we learn that most of trackers run slowly. Comparing to other trackers, our tracker achieves similar accuracy but works far faster.

Results on VOT2016 The VOT2016 shares the same sequences with VOT2015, just re-annotates the bounding boxes in the VOT2015 dataset. But there appears more excellent trackers in this year. The scores of trackers in this year are high in general, which may be caused by the similarity of the testing sequences. We select some excellent tracker for comparison, including C-COT [13], TCNN [34], SSAT [35], MLDF [44], and Staple [3].

As shown in Table 2, in the sequences of VOT2016, C-COT performs best in our test. Our tracker runs faster than all of these trackers with good accuracy.

Results on VOT2017 In VOT2017, they propose a new experiment, real-time experiment, for testing other indicators with time limit. If the tracker cannot give the result of current frame in a given time, the testing system will use the last reported bounding box for the result. It means the evaluation result drops sharply if the running speed lower than the specified value. But this experiment just divide the tracker into two parts, one part is slower than the specified value and the other part is faster. The differences between two trackers

Table 2 Comparsion with top trackers in VOT2016

Tracker	Accuracy	Robustness	EAO	EFO
C-COT [13]	0.54	0.24	0.33	0.51
TCNN [34]	0.56	0.27	0.33	1.05
SSAT [35]	0.58	0.29	0.32	0.48
MLDF [44]	0.49	0.23	0.31	1.48
Staple [3]	0.54	0.38	0.30	11.11
Ours	0.51	0.53	0.19	53.05

Table 3 Comparison with top trackers in VOT2017 real-time experiment

Tracker	Accuracy	Robustness	EAO
CSRDCF++ [33]	0.46	0.40	0.21
SiamFC [4]	0.50	0.60	0.18
ECOhc [8]	0.49	0.57	0.18
Staple [3]	0.53	0.69	0.17
ASMS [43]	0.49	0.63	0.17
Ours	0.45	0.81	0.13

in one part can not be told. Therefore, we add an addition experiment for running some state-of-the-art trackers of VOT2017 on one machine.

The result of the real-time experiment of VOT2017 is shown in Table 3. The methods used for comparison include CSRDCF++ [33], SiamFC [4], ECOhc [8], Staple [3], and ASMS [43].

In order to better demonstrate the speed advantage of our tracker, we make the test on a low-end device, the results are shown in Table 4 and the unit is FPS (frame per second). The methods used for comparison include MCPF [52], C-COT [13], MFT [1], CFWCR [8], SiamVGG [29], ECO [8], LADCF [50], SiamFC [4] and SiamRPN [27].

In VOT2017, some sequences in the data seems not challenging enough for state-of-the-art trackers, and hence are replaced by new sequences. From the reported results of the trackers, it is observable that sequences in VOT2017 are harder to track. But our tracker keeps the good performance. From the speed test, even on low-end device, our tracker still has an excellent speed, which is much faster than other trackers.

4.5 Experiments on OTB benchmarks

OTB2013 have two version, OTB50 and OTB100. OTB100 contains all of the sequences in OTB50. We use OTB100 to test our tracker for wider coverage. OTB100 evaluates trackers with one pass, which has not reinitialization after failure. This kind of evaluation is a big challenge for light network. In fact, this method is easy to waste the test set, because once the tracking fails, the following frames will be in vain.

In this experiment, we compare with some excellent tracker in these years, such as DeepSRDCF [10], SiamRPN [27], SiamFC [4], fDSST [11]. Aparting from this, we add some traditional trackers DCF, KCF [17], CN [12], CSK [18], and DAT [39] that are not state-of-art but light and fast for deploying and running just like ours. As shown in Fig. 5, even if OTB100 evaluates trackers with one pass, which has not reinitialization after failure, our tracker still have a good performance in success plot and precision plot.

Table 4 Speed comparison of excellent trackers in VOT2017

Tracker	MCPF [52]	C-COT [13]	MFT [1]	CFWCR [8]	SiamVGG [29]	ECO [8]	LADCF [50]	SiamFC [4]	SiamRPN [27]	Ours
Speed (FPS)	0.26	0.41	0.72	2.55	4.72	5.93	10.42	12.6	19.5	44.4

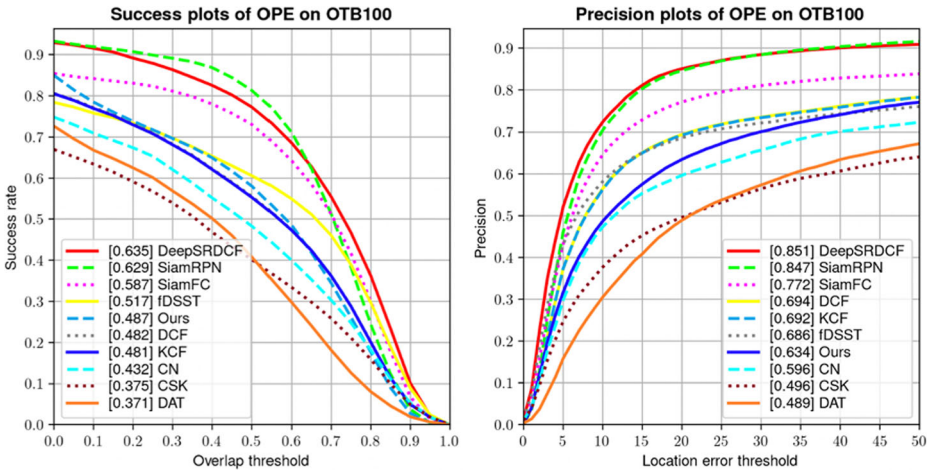

Fig. 5 Success plot and precision plot of OTB100

4.6 Model size comparison

Benefiting from the simplicity of our net, the model size is small enough. It makes it possible to deploy the model to low-end devices and even the mobile devices. We compare our model size with some trackers (DaSiamRPN [55], SiamMask [46], SiamRPN [27], SiamRPN++ [26], SiamVGG [29], SiamFC [4]) based on Siamese network. All of these models are end-to-end networks, the model size reflects the parameter size.

As shown in Table 5, our model is slighter than most of the Siamese models. SiamFC has the lightest model, but the running speed is far lower than ours.

5 Conclusion

In this paper, we propose a fast and effective object tracking algorithm based on Siamese network. Compared to previous trackers, our model can effectively avoid overfitting and reduce the computation time by selecting frames and cropping patches with some randomness to generate small size training samples. Further, an efficient and real-time tracker is proposed by stacking the SE block into Siamese network in an easily replaceable way. In addition, our proposed method can compute the location of the tracked object more efficiently by introducing a simple yet effective regression localization strategy. Experiments on four benchmarks demonstrate that the proposed model achieves real-time tracking even on low-end hardware without sacrificing much accuracy compared to the state-of-the-art trackers.

Table 5 Size comparison of Siamese related models

Tracker	DaSiamRPN [55]	SiamMask [46]	SiamRPN [27]	SiamRPN++ [26]	SiamVGG [29]	SiamFC [4]	Ours
Model Sieze (Mb)	345	96.8	86	41.9	31.5	9.4	16

The proposed efficient tracker can be applied to a variety of computer vision tasks that require acceleration, especially for some tasks that perform time-consuming frame-by-frame processing, such as video object detection, video object segmentation, and anomaly detection in surveillance videos.

In the future, we will investigate approaches that improve accuracy while preserving the high-speed achieved in this work. A possible way is to incorporate the recent novel idea of vision transformer (e.g., ViT [14] and Swin Transformer [32]) to replace convolutional neural networks to achieve efficient and robust object tracking. In addition, we will explore our proposed method in more complex scenarios such as multi-object tracking and 3D object tracking.

Acknowledgements This work is supported by the National Natural Science Foundation of China under Grant Nos. (62276127).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Bai S, He Z, Dong Y, Bai H (2020) Multi-hierarchical independent correlation filters for visual tracking. In: Proceedings of the IEEE international conference on multimedia and expo, pp 1–6
2. Benenson R, Petti S, Fraichard T, Parent M (2008) Towards urban driverless vehicles. J Vehicle Auton Syst 1/2(6):4–23
3. Bertinetto L, Valmadre J, Golodetz S, Miksik O, Torr PHS (2016) Staple: complementary learners for real-time tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1401–1409
4. Bertinetto L, Valmadre J, Henriques JF, Vedaldi A, Torr PHS (2016) Fully-convolutional siamese networks for object tracking. In: Proceedings of the European conference on computer vision workshops, vol 9914, pp 850–865
5. Beymer D, Konolige K (1999) Real-time tracking of multiple people using continuous detection. In: Proceedings of the IEEE frame rate workshop, pp 1–8
6. Bolme DS, Beveridge JR, Draper BA, Lui YM (2010) Visual object tracking using adaptive correlation filters. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2544–2550
7. Comaniciu D, Ramesh V, Meer P (2003) Kernel-based object tracking. IEEE Trans Pattern Anal Mach Intell 25(5):564–577
8. Danelljan M, Bhat G, Khan FS, Felsberg M (2017) ECO: efficient convolution operators for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6931–6939
9. Danelljan M, Häger G, Khan F, Felsberg M (2014) Accurate scale estimation for robust visual tracking. In: British machine vision conference, pp 65.1–65.11
10. Danelljan M, Håger G, Khan FS, Felsberg M (2015) Learning spatially regularized correlation filters for visual tracking. In: Proceedings of the IEEE international conference on computer vision, pp 4310–4318
11. Danelljan M, Håger G, Khan FS, Felsberg M (2017) Discriminative scale space tracking. IEEE Trans Pattern Anal Mach Intell 39(8):1561–1575
12. Danelljan M, Khan FS, Felsberg M, van de Weijer J (2014) Adaptive color attributes for real-time visual tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1090–1097
13. Danelljan M, Robinson A, Khan FS, Felsberg M (2016) Beyond correlation filters: learning continuous convolution operators for visual tracking. In: Proceedings of the European conference on computer vision, vol 9909, pp 472–488
14. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations

15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
16. Held D, Thrun S, Savarese S (2016) Learning to track at 100 fps with deep regression networks. In: Proceedings of the European conference on computer vision, vol 9905, pp 749–765
17. Henriques JF, Caseiro R, Martins P, Batista J (2015) High-speed tracking with kernelized correlation filters. IEEE Trans Pattern Anal Mach Intell 37(3):583–596
18. Henriques JF, Caseiro R, Martins P, Batista JP (2012) Exploiting the circulant structure of tracking-by-detection with kernels. In: Proceedings of the European conference on computer vision, vol 7575, pp 702–715
19. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141
20. Hua Y, Alahari K, Schmid C (2015) Online object tracking with proposal selection. In: Proceedings of the IEEE international conference on computer vision, pp 3092–3100
21. Huang L, Zhao X, Huang K (2021) Got-10k: a large high-diversity benchmark for generic object tracking in the wild. IEEE Trans Pattern Anal Mach Intell 43(5):1562–1577
22. Kristan M, Leonardis A, Matas J, Felsberg M, Pflugfelder R, Cehovin Zajc L, Vojir T, Hager G, Lukezic A, Eldesokey A et al (2017) The visual object tracking VOT2017 challenge results. In: Proceedings of the IEEE international conference on computer vision workshops, pp 1949–1972
23. Kristan M, Matas J, Leonardis A, Felsberg M, Cehovin L, Fernández G, Vojír T, Häger G, Nebehay G, Pflugfelder RP (2015) The visual object tracking VOT2015 challenge results. In: Proceedings of the IEEE international conference on computer vision workshop, pp 564–586
24. Kristan M et al (2016) The visual object tracking VOT2016 challenge results. In: Proceedings of the European conference on computer vision workshops, vol 9914, pp 777–823
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1106–1114
26. Li B, Wu W, Wang Q, Zhang F, Xing J, Yan J (2019) Siamrpn++: evolution of siamese visual tracking with very deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4282–4291
27. Li B, Yan J, Wu W, Zhu Z, Hu X (2018) High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8971–8980
28. Li C, Lin S, Qiao J, An S (2021) Partial tracking method based on siamese network. Vis Comput 37(3):587–601
29. Li Y, Zhang X (2019) Siamvgg: visual tracking using deeper siamese networks. arXiv:1902.02804
30. Lin T, Maire M, Belongie SJ, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: Proceedings of the European conference on computervision, vol 8693, pp 740–755
31. Liu T, Wang G, Yang Q (2015) Real-time part-based visual tracking via adaptive correlation filters. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4902–4912
32. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 10012–10022
33. Lukezic A, Vojir T, Zajc LC, Matas J, Kristan M (2017) Discriminative correlation filter with channel and spatial reliability. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4847–4856
34. Nam H, Baek M, Han B (2016) Modeling and propagating cnns in a tree structure for visual tracking. arXiv:1608.07242
35. Nam H, Han B (2016) Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4293–4302
36. Ning X, Duan P, Li W, Shi Y, Li S (2020) A CPU real-time face alignment for mobile platform. IEEE Access 8:8834–8843
37. Ning X, Duan P, Li W, Zhang S (2020) Real-time 3d face alignment using an encoder-decoder network with an efficient deconvolution layer. IEEE Signal Process Lett 27:1944–1948
38. Ong P, Chong TK, Ong KM, Low ES (2021) Tracking of moving athlete from video sequences using flower pollination algorithm. Vis Comput
39. Possegger H, Mauthner T, Bischof H (2015) In defense of color-based model-free tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2113–2120
40. Ren S, He K, Girshick R, Sun J (2017) Faster r-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149

41. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Li F (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
42. Vishwakarma S, Agrawal A (2013) A survey on activity recognition and behavior understanding in video surveillance. Vis Comput 29(10):983–1009
43. Vojir T, Noskova J, Matas J (2014) Robust scale-adaptive mean-shift for tracking. Pattern Recognit Lett 49:250–258
44. Wang L, Ouyang W, Wang X, Lu H (2015) Visual tracking with fully convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp 3119–3127
45. Wang N, Yeung D (2014) Ensemble-based tracking: aggregating crowdsourced structured time series data. In: Proceedings of the international conference on machine learning, vol 32, pp 1107–1115
46. Wang Q, Zhang L, Bertinetto L, Hu W, Torr PH (2019) Fast online object tracking and segmentation: a unifying approach. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1328–1338
47. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7794–7803
48. Woo S, Park J, Lee JY, So Kweon I (2018) Cbam: convolutional block attention module. In: Proceedings of the European conference on computer vision, pp 3–19
49. Wu Y, Lim J, Yang MH (2013) Online object tracking: a benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2411–2418
50. Xu T, Feng Z, Wu X, Kittler J (2019) Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. IEEE Trans Image Process 28(11):5596–5609
51. Yang C, Duraiswami R, Davis LS (2005) Fast multiple object tracking via a hierarchical particle filter. In: Proceedings of the IEEE international conference on computer vision, pp 212–219
52. Zhang T, Xu C, Yang M (2017) Multi-task correlation particle filter for robust object tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4819–4827
53. Zhang W, Du Y, Chen Z, Deng J, Liu P (2021) Robust adaptive learning with siamese network architecture for visual tracking. Vis Comput 37(5):881–894
54. Zhu G, Porikli F, Li H (2015) Tracking randomly moving objects on edge box proposals. arXiv:1507.08085
55. Zhu Z, Wang Q, Li B, Wu W, Yan J, Hu W (2018) Distractor-aware siamese networks for visual object tracking. In: Proceedings of the European conference on computer vision, pp 101–117

Affiliations

Feng Han[1,2] · Shaokui Jiang[1,2] · Jianmin Wu[3] · Baile Xu[1,2] · Jian Zhao[4] · Furao Shen[1,5]

Jian Zhao
jianzhao@nju.edu.cn

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China

[2] Department of Computer Science and Technology, Nanjing University, Nanjing, 210023, China

[3] State Grid Shanghai Maintenance Company, Shanghai, 200063, China

[4] School of Electronic Science and Engineering, Nanjing University, Nanjing, 210023, China

[5] School of Artificial Intelligence, Nanjing University, Nanjing, 210023, China