Deep Position-Sensitive Tracking

Yufei Zha, Tao Ku, Yunqiang Li, and Peng Zhang, *Member, IEEE*

***Abstract*—Classification-based tracking strategies often face more challenges from intra-class discrimination than from inter-class separability. Even for deep convolutional neural networks that have been widely proven to be effective in various vision tasks, their intra-class discriminative capability is still limited by the weakness of softmax loss, especially for targets not seen in the training dataset. By taking intrinsic attributes of training samples into account, in this paper, we propose a position-sensitive loss coupled with softmax loss to achieve intra-class compactness and inter-class explicitness. Particularly, two additive margins are introduced to encode the position attribute for decision boundary maximization, which is also utilized with the proposed loss to supervise the fine-tuned features on the pre-trained model. With the nearest neighbor ranking measurement in the feature embedding domain, the whole scheme is able to reach an optimized balance between the feature-level inter-class semantic separability and instance-level intra-class relative distance ranking. We evaluate the proposed work on different popular benchmarks, and experimental results demonstrate that our tracking strategy performs favorably against most of the state-of-the-art trackers in the comparison of accuracy and robustness.**

***Index Terms*—Visual tracking, softmax loss, position-sensitive loss, ranking.**

I. Introduction

VISUAL tracking [1]–[3] is usually regarded as a high-level analysis based on fundamental image classification/retrieval models, depending on representative features generation, and tracking accuracy and robustness have been significantly improved when a variety of deep CNNs have been used in recent studies [4]. Although there are some technical overlapping topics between tracking and image classification/retrieval [1], such as the object's semantic categorization, object tracking has its particularity: category unknown with dynamic localization, which has become a challenging source limiting the performance further improvement.

The recently proposed tracking-by-detection approach directly employs the CNN feature embedding to model object appearance for accurate identification. The DeepSRDCF [5] utilizes the feature pre-trained by imagenet-vgg-2048 [6] on the ImageNet dataset [7] to provide superior tracking performance compared to the hard-crafted features. Similarly, the hierarchical CNN features are also simply integrated into the DCF tracking framework [8], [9] with fixed coefficients [10] or adaptive weights [11] to represent the object, and the lightweight networks are also designed to learn the CNN feature or generic representation [4] for the object from the tracking dataset. Unfortunately, such a straightforward integration mechanism is hard to robustly track the unknown object that is not included in the training dataset according to an inter-class semantic separability feature embedding. To overcome this issue, a discriminative feature embedding is required when tracking is on-the-fly.

In fact, in the field of image retrieval tasks [13], [14], the query images usually do not appear in the retrieval image library. For this challenge, some efforts [15], [16] have developed a discriminative feature embedding to enhance the intra-class compactness through improving the loss function of the network. However, they still focus on the feature-level semantics of the given image patch to verify that the input image pair is similar or not. It is not consistent with tracking tasks to accurately locate the object in the frames. Thus, the position relationship between sample and the true object in instance-level is required to be considered in the loss function to learn the discriminative feature embedding, which is helpful to promote the tracking performance.

Summarily, the feature embedding extracted from the network that is mainly based on the softmax loss [4] is able to semantically discriminate the foreground from background, but its limitations for tracking tasks are also obvious:

1) Such inter-class separability of the feature embedding is difficult to distinguish an unknown object that is not contained in the training dataset. It is impractical to gather all the possible objects for training, and the prediction by the CNN feature embedding is not always applicable. Thus, the learned feature embedding is required to be discriminative and generalized enough for identifying a new unseen object for tracking tasks. Discriminative power is characterized by the intra-class compactness and inter-class separability. However, the softmax loss only encourages the inter-class semantic separability of the feature embedding, which makes

Manuscript received February 26, 2019; revised May 19, 2019 and June 5, 2019; accepted June 6, 2019. Date of publication June 11, 2019; date of current version December 31, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants 61773397, 61571362, and 61703423 and in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2018JM6015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Marco Carli. *(Tao Ku is co-first author.) (Corresponding author: Peng Zhang.)*

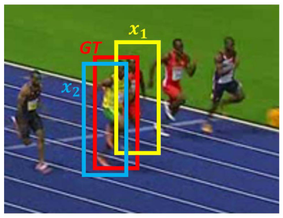
Y. Zha and P. Zhang are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710038, China (e-mail: zhayufei@126.com; zh0036ng@nwpu.edu.cn).

T. Ku is with the Department of Information and Computing Sciences, Utrecht University, Utrecht 3584 CC, The Netherlands (e-mail: t.ku@uu.nl).

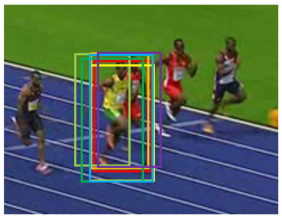
Y. Li is with the Vision Lab, Delft University of Technology, Delft 2628 CD, The Netherlands (e-mail: y.li-19@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

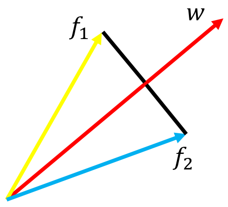
Digital Object Identifier 10.1109/TMM.2019.2922125



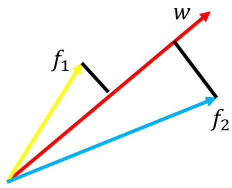
(a) Bounding boxes



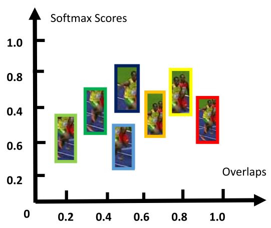
(b) Samples



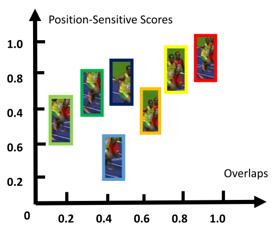
(c) Features embedding of the softmax loss



(d) Features embedding of position-sensitive loss



(e) Softmax scores



(f) Position-sensitive scores

Fig. 1. Softmax Loss V.S. Position-Sensitive Loss. Figure 1(a) shows the sample x_1 in the yellow and sample x_2 in the blue are gathered in terms of the true object labeled with the red box in the sequence *bolt* from the OTB2015 dataset [12]. The feature embedding trained by the softmax loss and by the proposed position-sensitive loss are shown in Figure 1(c) and 1(d), respectively. Here, the overlap between x_1 and the true object is smaller than the overlap between x_2 and the true object. The corresponding features f_1, f_2 and w are extracted by a network with respect to the samples x_1, x_2 and the true object, respectively. The bigger the inner-product value between the features w and f is, the more similar the sample x and the true object is. In Figure 1(b), many samples are gathered with different overlaps according to the true object. The relationship between the overlap and the inner-product score is shown in Figure 1(e) and 1(f), where they are trained by the softmax loss and the proposed position-sensitive loss, respectively.

the tracker unable to follow the object effectively and robustly.

2) The absence of localization confidence makes it difficult to perceive the differences of samples that have been identified as the same class since their intra-class semantic meanings are similar. As shown in Figure 1(c), the projection of the feature f_1 on the feature w is the same as the feature f_2, which means that the softmax loss treats f_1 and f_2 identically even the overlaps between sample x_1 and x_2 with respect to the object have a big difference. When numerous samples are gathered with different overlaps according to the object position in one frame as in Figure 1(b), the relationship between the overlaps and the inner-product scores trained by the softmax loss in Figure 1(e) shows that the softmax scores are ranked according to the samples' semantic meanings rather than the overlaps between the samples and the true object. In a tracking task, accurate localization can be both a goal and a condition, and it also means that the object position attribute should be taken into account during network training, which becomes a motivation of this study.

In our study, two additive margins are designed to encode the object position attribute and then coupled with softmax loss to re-define a novel position-sensitive loss for the discriminative feature embedding learning. By explicitly encouraging instance-level intra-class relative distance ranking and feature-level inter-class separability between the learned feature embedding f_1 and f_2 in Figure 1(d), the difference in similarity between samples x_1, x_2 and the true object can be captured because the new loss takes both the semantic and the position attribute into account, which leads to accurate sample selection: x_2 instead of x_1, as the tracking results. Comparing to the softmax scores, the position-sensitive scores are proportional to the overlaps because the proposed loss depends on both the semantic and the position attributes of the sample as shown in Figure 1(f).

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

- We propose a position-sensitive loss to learn a discriminative feature embedding, which can explicitly encourage intra-class compactness and inter-class separability. Compared with the traditional CNN features, the learned features help to distinguish the invisible objects not included in the training dataset.
- We introduce two additive margins into the common softmax loss. The margin is used to encode the position attribute of the training data. During training, the overlap between the sample and ground truth is also supervised to maximize the decision boundary for discriminative feature embedding. As a result, a sample including the whole object will achieve a high score.
- We use the proposed loss to fine-tune the backbone network in base on the pre-trained model l [4], [6] on the tracking datasets [17]. Thus, the learned feature embedding not only has good separability between different categories but also sorts the samples according to their overlaps with respect to the object.

The remainder of the paper is structured as follows. In Section II, we review most closely related work, our approach is described in Section III and Section IV, experimental results are presented in Section V and conclusions are drawn in Section VI.

II. Related Work

Visual tracking has been studied extensively over the past decades. Comprehensive reviews on object tracking and benchmark evaluations can be found in [18]. In this section, we mainly discuss the deep trackers and the related topic of feature embedding learning with a large margin.

Tracking based on CNN features: Compared with the hand-crafted features [8], [19], CNN features have demonstrated powerful representation ability for visual tracking tasks in recent literature [2], [4], [20]–[23]. In the early works, the last layer [24] or hierarchical layers [10] of CNN features fine-tuned on the pre-trained model from the ImageNet [7] dataset are employed to describe the the object's variations.

In order to learn the specific CNN feature embedding for tracking tasks, the tracking datasets [12], [25] are also employed as the training data and the model is trained with the softmax loss [4], [22], [23] or a contrastive loss [26], [27] in an end-to-end way. The softmax loss is employed to learn inter-class separability feature embedding to identify the object from the background, and then the object's bounding box is inferred in the following frames either by regressing directly [23], [26], or estimating with correlation filter [28], or bounding boxes regression [23]. Specifically, in the work [4], the softmax function is used as a binary classifier to distinguish the object and background on the tracking dataset to achieve the CNN feature embedding for the generic object.

The feature embedding can be trained by the contrastive loss and is simultaneously used to metric the similarity between the object and the sample through Siamese network [29] with the Y-shaped network architecture [26]. Additionally, SORT [30] learned a deep association metric offline and introduce measurement-to-track associations to query the nearest neighbor for multiple objects tracking, and low- and high-dimensional features [31] were fused to track generic human motion.

In our study, the CNN feature embedding is learned by the proposed new loss based on the softmax loss and used to deal with the unseen object [32] in the training dataset since the learned feature embedding is inter-class separability and intra-class ranking.

Discriminative feature embedding learning: In most CNNs, the softmax loss [33] is used to learn the feature embedding with richer identity-related information. In order to enhance the discriminative power of the softmax loss, the center loss [34] is proposed as regularization attached to softmax loss, which simultaneously learns a center for the feature embedding of each class and penalizes the distances between the deep features and their corresponding class centers. Also, the normalized feature embedding is added to the softmax [35] loss to promote the performance. Specifically, when both the feature embedding and parameters are normalized simultaneously [35], the formulation of cross-entropy loss can be consistent with the distance-based metric learning. Namely, the distance is equal to the cosine distance [36], and it still has the same formulation with the softmax loss. Recently, a generalized large margin softmax [15] adds a margin between the angle of different categories, which explicitly encourages intra-class compactness and inter-class separability.

Different from the above multiplicative angular margin, our work is similar to the additive cosine margin [16] which incorporates angular margin and cosine margin into the losses to enhance the discriminative power of the softmax loss. Instead of treating the margin as a hyperparameter, two additive margins are designed to encode the position attribute of the sample and enlarge decision boundary according to the overlap between the sample and the object. As a result, the sample containing the whole object is going to achieve a high score.

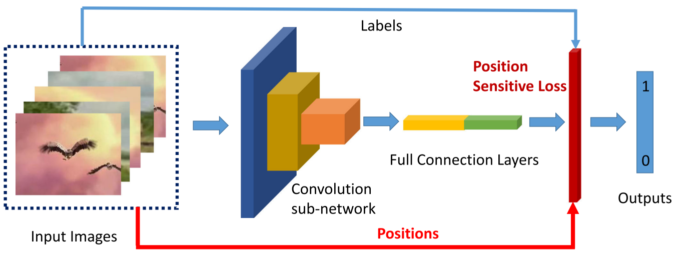


Fig. 2. Feature learning pipeline. The network architecture includes the following parts: (1) the inputs that include the sampled patches gathered from the frame, (2) the convolution sub-network that includes convolutional layers, ReLU layers, and pooling layers, (3) the full connection layers and (4) the loss layer that calculates the loss both by the labels and positions. The output value of the network is between 0 and 1 that denotes the probability of the sample belonging to the object or background.

III. Deep Position-Sensitive Tracking

A. Overview

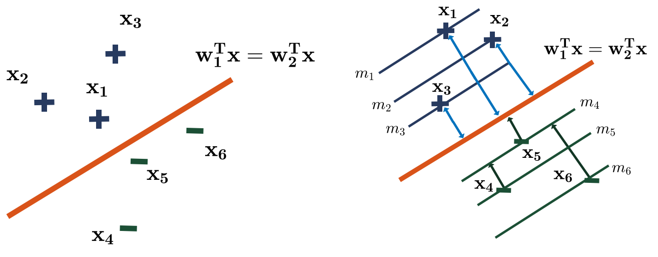
In current tracking frameworks, the softmax loss is widely used to learn the CNN feature embedding to distinguish the foreground from the background. However, the samples around the object are always considered as positives, but they have different metrics in terms of the ground truth. Such weakness in inter-class separability would eventually reduce the accuracy of localization and lead to tracking failure due to ignored intra-class ranking. To solve this problem, a novel position-sensitive loss is introduced and incorporated with a deep neural network to learn a discriminative feature embedding for object tracking, which is illustrated in Figure 2.

In this framework, the incorporated position-sensitive loss is used to constrain the training process by encouraging the intra-class ranking and the inter-class separation, and the network output is the probability of a sample belonging to the object or background. Each convolution layer contains a ReLU and a dropout unit shown in Figure 2 and the SGD (Stochastic Gradient Descent) optimizer is adapted to train the network. Unlike the regression model encoding the position as labels, we incorporate the position attribute intrinsically into the classification model. As a consequence, both the semantic and the position attribute of the samples can be taken into account for learning.

B. Preliminaries

The following notations are used in the rest of this paper. Given training dataset $\{\boldsymbol{I}_i, y_i, m_i | i = 1, 2, \ldots, N\}$, $\boldsymbol{I}_i \in \mathbb{R}^n$ is the *i-th* input sample, and the corresponding label is $y_i \in \{1, 2, \ldots, C\}$. The position attribute $m_i \in [0, 1]$ of the *i-th* sample $\boldsymbol{I}_i$ can be calculated by the definition of CLE or IoU margin, and x_i denotes the feature embedding extracted by the deep network.

We define the CLE based margin as $m^{cle}(p, q) = e^{-||C_p - C_q||^2}$ and IoU based margin as $m^{iou}(p, q) = \frac{|p \cap q|}{|p \cup q|}$, where p and q denote the bounding boxes of the sample and the object, respectively. The C_p and C_q denote their center locations, and the operators $\cap$ and $\cup$ are the intersection and union of two bounding boxes.



(a) softmax Loss (b) Position-Sensitive Loss

Fig. 3. Position margin. The decision boundary learned by the softmax loss is shown in Figure 3(a), where the samples with different labels have been separated on the two sides of the decision boundary. Figure 3(b) shows the learned decision boundary by the position-sensitive loss where m_i is the position margin of the sample x_i with respect to the ground truth. The unordered margin is meaningful separability in inter-class samples while helpless in intra-class samples.

In our study, the **inter-class separability** is used to measure how well two different clusters are separated from each other. While the **intra-class compactness** is employed to encourage the samples with the same label to gather together.

C. Position-Sensitive Loss

In tracking evaluation, the position information can be measured by the center location error (CLE) and intersection over union (IoU) between the bounding boxes for precision, which can be also regarded as suitable metrics for representing the sample position attribute. To improve the discriminability of the feature embedding, the relative position relationship between the sample and the object is utilized as the marginal supervision to build position-sensitive loss.

With the softmax loss, these samples can be effectively classified into different categories by the learned feature embedding. But in tracking tasks, the sample classification needs to enlarge the intra-class ranking as well as the inner-class separability. The softmax loss can be reformulated as the last full-connected layer input $\{\boldsymbol{x}_i \in \mathbb{R}^d, i = 1, \ldots, N\}$ with weights $\{\boldsymbol{w}_i \in \mathbb{R}^d \in, i = 1, \ldots, C\}$:

$$\mathcal{L}_{softmax} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i}}{\sum_{j=1}^{c} e^{\boldsymbol{w}_j^T x_i}}, \tag{1}$$

where C is the number of classes.

Our target is to rank the samples that have the same category by introducing position attribute as the soft margin in the softmax loss:

$$\mathcal{L}_{position} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - m_i}}{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - m_i} + \sum_{j=1, j \neq y_i}^{c} e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}}. \tag{2}$$

The following discussion about position margin may help to understand position-sensitive loss clearly.

D. Position Margin

The samples with the different categories will be separated according to the hyper-planes generated under the supervising signal of the softmax loss shown in Figure 3(a). Unfortunately, the learned feature embedding is limited to present unseen objects. Luckily, the parallel hyper-planes will be achieved shown in Figure 3(b) due to the item m_i in the position-sensitive loss represented in the formulation 2. We define the region bounded by these hyperplanes as the positioning margin according to the item m_i. For samples with the same category, the smaller the position margin, the further away from the decision boundary. Summarily, this position margin can be derived from inter-class semantic separability and intra-class position ranking accordingly.

Inter-class semantic separability: When C is set to 2, the loss becomes a two-class classification problem as:

$$p_1 = \frac{e^{\boldsymbol{w}_1^T \boldsymbol{x}_i - m_i}}{e^{\boldsymbol{w}_1^T \boldsymbol{x}_i - m_i} + e^{\boldsymbol{w}_2^T \boldsymbol{x}_i}}, \tag{3}$$

$$p_2 = 1 - p_1 = \frac{e^{\boldsymbol{w}_2^T \boldsymbol{x}_i}}{e^{\boldsymbol{w}_1^T \boldsymbol{x}_i - m_i} + e^{\boldsymbol{w}_2^T \boldsymbol{x}_i}}, \tag{4}$$

where p_1 and p_2 denote the probabilities that $\boldsymbol{x}_i$ belongs to class 1 and class 2. If $\boldsymbol{x}_i$ is a member of class 1, we prefer that $p_1 > p_2$, that is,

$$e^{\boldsymbol{w}_1^T \boldsymbol{x}_i} > e^{\boldsymbol{w}_2^T \boldsymbol{x}_i + m_i} > e^{\boldsymbol{w}_2^T \boldsymbol{x}_i}. \tag{5}$$

The model $\boldsymbol{w}$ trained by the softmax loss can separate the samples into different categories, but fail to reflect their intra-class ranking as shown in Figure 3(a). Comparatively, model $\boldsymbol{w}$ trained by the position-sensitive loss is able to identify that the samples with a large margin of m have a bigger inner product. This means that the proposed loss encourages that the closer the margin between the sample and ground truth is, the larger score the inner product of the model and sample is. The equation 5 illustrates such differences, the margin m_i enables model $\boldsymbol{w}_1$ to be more discriminative than the one trained by the traditional softmax loss.

Intra-class position ranking: For two-class problem, to avoid the softmax loss degenerating into logistical regression, the probabilities of samples can then be formulated as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, when they both belong to the class 1.

$$p_1 = \frac{1}{1 + e^{-(\boldsymbol{w}_1^T \boldsymbol{x}_1 - m_1)}}, p_2 = \frac{1}{1 + e^{-(\boldsymbol{w}_1^T \boldsymbol{x}_2 - m_2)}}. \tag{6}$$

The probability of sample $\boldsymbol{x}_1$ is supposed to be bigger than the sample $\boldsymbol{x}_2$, when the corresponding margin is $m_1 > m_2$, correspondingly $p_1 > p_2$:

$$e^{-\boldsymbol{w}_1^T \boldsymbol{x}_1} < e^{-(\boldsymbol{w}_1^T \boldsymbol{x}_2 + \Delta m)} < e^{-\boldsymbol{w}_1^T \boldsymbol{x}_2}, \tag{7}$$

where $\Delta m = m_1 - m_2$. If the position margin m_1 of sample $\boldsymbol{x}_1$ is bigger than the margin m_2 of sample $\boldsymbol{x}_2$, denoting that the inner product of sample pair $(\boldsymbol{x}_1, \boldsymbol{w})$ is bigger than $(\boldsymbol{x}_2, \boldsymbol{w})$ with a generated distance Δm. This means that when a network is trained by increasing the spatial position of the sample, the probability of the sample can be given a margin according to its Δm. The rank of the sample with a small position with respect to the true object will be advanced, and vice versa.

Algorithm 1: Training of Position-Sensitive Loss

```
Require: The feature of the softmax layer:
  ${z_i ∈ \mathbb{R}^{d×C}}$;
    The labels: ${y_i ∈ {1, 2, ..., C}}$; The position
    margins: ${m_i ∈ [0, 1]}$.
Ensure: The Loss: $\mathcal{L}$; The derivation: $\nabla\mathcal{L}$.
  1: $q : index(\boldsymbol{z}_i \in y)$
  2: $\boldsymbol{z}_i(q) = \boldsymbol{z}_i(q) - m_i$;
  3: $\boldsymbol{z}_{max} = \max \boldsymbol{z}$;
  4: $\boldsymbol{ez} = \exp(\boldsymbol{z} - \boldsymbol{z}_{max})$;
  5: Forward:
  6: $\mathcal{L} = \boldsymbol{z}_{max} + \log \sum \boldsymbol{ez} - \boldsymbol{z}(q)$;
  7: Backward:
  8: $\nabla\mathcal{L} = \boldsymbol{ez} / \sum \boldsymbol{ez}$;
  9: $\nabla\mathcal{L} = \nabla\mathcal{L}(q) - 1$.
```

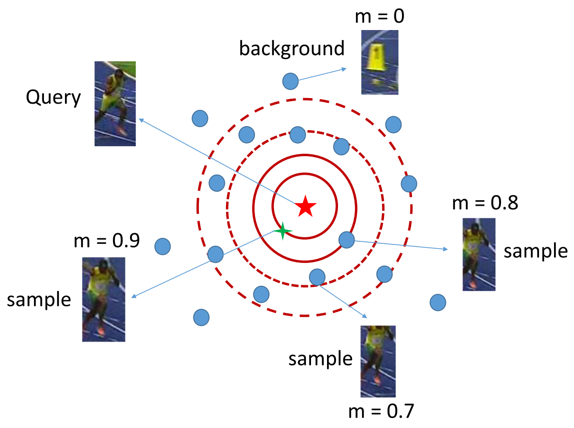


Fig. 4. Feature embedding. The red five-pointed star is the feature embedding of the true object, while the feature embedding of the samples in the gallery are denoted as blue circles. The sample with margin $m = 0.9$ has the nearest European distance with respect to the true object.

E. Optimization

The proposed position-sensitive loss incorporates both the semantic label and position attribute of the sample into the base of the softmax loss, its back-propagation derivation can be written as:

$$\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}_k} = \Bigg[-\delta(y_i, k) + \frac{\delta(y_i, k) e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - m_i} + (1 - \delta(y_i, k)) e^{\boldsymbol{w}_k^T \boldsymbol{x}_i}}{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - m_i} + \sum e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}} \Bigg] \boldsymbol{x}_i, \tag{8}$$

where $\delta()$ is the indication function. The derivation of the inputs:

$$\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{x}_i} = \frac{\sum_{j=1, j \neq y_i}^{c} (\boldsymbol{w}_j - \boldsymbol{w}_{y_i}) e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}}{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - m_i} + \sum_{j=1, j \neq y_i}^{c} e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}}. \tag{9}$$

The calculation of Eq. 8 and Eq. 9 is similar to the traditional softmax loss, but the score of the correct category is subtracted by the margin m_i of the corresponding sample $\boldsymbol{x}_i$. When $m_i = 0$, this derivation will turn back to its original form of the softmax loss.

The training procedure is shown in the Algorithm 1. The line 2 in Algorithm 1 is used to subtract the position margin of the corresponding sample, and other operations still follow the original softmax loss.

IV. Online Tracking

As shown in Figure 4, The feature embedding of the object and samples are extracted by the trained network and the distance measurement is then directly used to evaluate the similarity between the object and samples. In the feature embedding domain, the nearest sample to the object is considered as the instant output.

For instance, given an object $\boldsymbol{t}$, the embedding feature $f(\boldsymbol{t})$ is regarded as a query, a set of N samples around the previous object position are extracted from the current frame, and they can be denoted as $\mathcal{S} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\}$, which includes both background and distractors. Those samples act as the input for the network including both background and distractors. Here $\boldsymbol{z}_i \rightarrow \boldsymbol{x}_i$ denotes the mapping from the sample domain to the feature embedding domain. Thus, $\mathcal{E} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is the set of the feature embedding. Then the Euclidean distances between the object and samples can be calculated as:

$$D_i = ||f(\boldsymbol{t}) - f(\boldsymbol{x}_i)||_2^2. \tag{10}$$

After that, the nearest sample is considered as the tracking result of the current frame.

$$i^* = \arg \min_{i=1,2,\ldots,N} D_i. \tag{11}$$

V. Experiments

The proposed work is implemented using MatConvNet toolbox [37], and the performance is evaluated on two public benchmark dataset of OTB [12] and VOT2016 [38]. The average speed is around 1 fps without any code optimization on the testing platform of Intel I7 3.6GHz and GeForce GTX 1080Ti.

A. Implementation Details

Network architecture: By replacing the softmax loss with position-sensitive loss shown as with red color in Figure 2, we build our network based on the architectures of MDNet [4] and VGG-M [37]. There are three convolutional layers (conv1,conv2, and conv3) utilized to extract the feature embedding, and two full connected layers (fc4 and fc5) to calculate the final similarity score. The number of filters in each layer are 96, 256 and 512 with a uniform size of 3-by-3 and two full connection layers have 512 units each. The inputs are 107-by-107 RGB images and the output value is between 0 and 1.

Training data setup: The dataset ALOV300++ [39] is employed as training data, which of 315 video sequences. This dataset consists of the real-life videos from YouTube with 64 different types of targets ranging from human face, a person, a ball, an octopus, microscopic cells, a plastic bag to a can. Note that 12 videos in ALOV300++ [39] are not included in the training data because they also appear in OTB2015 [12] and VOT2016 [38]. Positives are sampled around the ground truth with overlap more than 0.7, and the random samples with overlaps less than 0.5 are chosen as negatives.

Off-line training: For each iteration of off-line training, we construct a mini-batch with samples collected from a single frame like MDNet [4]. We randomly sample 8 frames in the selected sequence, and gather 32 positives, 96 negatives from each frame, and then re-arrange them into a batch of 256 positives and 768 negatives. Instead of back-propagating in each iteration, we accumulate the gradients from backward passes in multiple iterations, and the network is updated at every 50 iterations in our experiments. The weights of the three convolutional layers are transferred from the corresponding parts in VGG-M network vgg14 pre-trained on ImageNet [40] while the fully connected layers and the loss layer are trained from scratch.

Online tracking: The feature embedding extracted by the convolutional layers is treated as the generic representation of an object, which means the pre-trained network needs to be fine-tuned on the first frame of each testing video. 500 positives and 5000 negatives are collected in terms of the same IoU criteria during the off-line stage. While the training data for online updating are collected in each frame, where 50 positives ($> 0.7IoU$) and 200 negatives ($< 0.3IoU$) are gathered in terms of the location estimations in current frame. To avoid redundant computation, we only keep their feature representations instead of the image patches and carry out updating every 10 frames.

Optimization: The proposed network is trained by a stochastic gradient descent (SGD) method. For off-line learning, we first freeze all the layers except the full-connected layers and train the network with the loss layer for 1000 epochs with learning rate 0.01. Then, we unfreeze all the layers to do training one more time with the learning rate group $[0.00001, 0.0001, 0.001]$ of convolution layers, full connection layers and loss layer respectively. During the online updating, the number of iterations for fine-tuning is set to 15 and the learning rate is set to 0.0003. The weight decay and momentum are fixed to 0.0005 and 0.9, respectively.

B. Ablation Study

To analyze each component, we conduct an ablation study on the OTB50 dataset [41], and the precision rate and success rate are used as our evaluation criteria. Unlike softmax loss focusing on classification error rate, our loss pays attention both on the position ranking and the semantic separability of the learned feature embedding.

Position-sensitive loss *v.s.* softmax loss: The network consists of a backbone sub-network and a head sub-network. In order to fairly verify the superiority of the proposed loss in this paper, we evaluate different loss defined by the head sub-network, and its input is provided by the backbone sub-network. The experiments are performed on the ALOV300++ [39] dataset for training and the OTB50 dataset [41] for testing. Figure 5 shows that the proposed position-sensitive loss has effectively improved the tracking precision (0.758) and the AUC score(0.647), in comparison to 0.708 and 0.606 obtained by the softmax loss. The reason is that the position-sensitive loss takes the position ranking into account with the semantic separability of the softmax loss. The

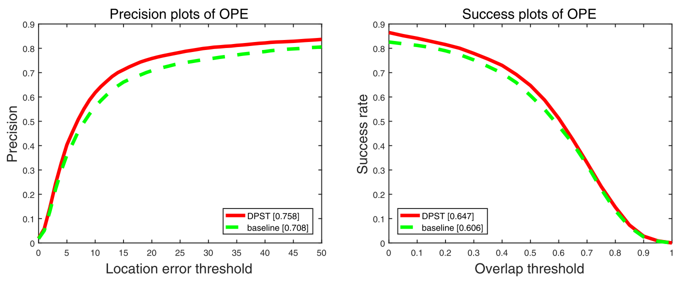


Fig. 5. Performances of position-sensitive loss and softmax Loss. The green line is the performance with the softmax loss for object semantic, on the contrary, the performance of the position-sensitive loss is shown in red line.

TABLE I
PRECISION AND SUCCESS ON CLE AND IOU

Measurements	Precision	Success
Center Location Error (CLE)	0.767	0.660
Intersection over Union (IoU)	0.763	0.674

overall performance has been improved nearly 5% on the OTB50 dataset mainly because the learned feature embedding has discriminative capability based on the object's difference position. And this advantage also enhances the separability between the object and background containing objects' parts.

Margin comparison: Two margins: center location error (CLE) and intersection over union (IoU), are also tested on the OTB50 and the results in Table I. The similar scores show that both margins are effective in position attribution description. The CLE margin is slightly better than the IoU margin in terms of accuracy score, but its AUC score is slightly worse. The reason behind might be that the features learned with CLE margin mainly focus more on the center coordinates, but the IoU margin expects a better overlap between the sample and the ground truth.

Intra-class ranking: To verify the intra-class ranking of samples on OTB50, the sequence *bolt* is chosen for visualized analysis shown as Figure 6. The blue plot and red plot represent the scores of feature embedding trained with softmax loss and position-sensitive loss respectively. The horizontal axis denotes the overlap changes and the vertical axis denotes the corresponding similarity scores. It can be found that the scores trained with position-sensitive loss approximate a linear proportion of the overlaps between the corresponding sample and the real object. Namely, the scores of samples (a) and (b) based on softmax loss are close, then they will be considered to have the same semantic. But the difference between samples (a) and (b) can be found by the position-sensitive loss: the closer the sample near the real object, the higher the score of the sample.

C. Comparison With the State-of-the-Art

The verification of the proposed tracker is carried out on the popular benchmark datasets of OTB2015 [12] and VOT2016 [38], and we compare with the other state-of-the-art tracking works including TCNN [42], CCOT [43], Staple [44], DeepSRDCF [5], EBT [45], DAT [46], DSST2014 [47], TGPR [48]

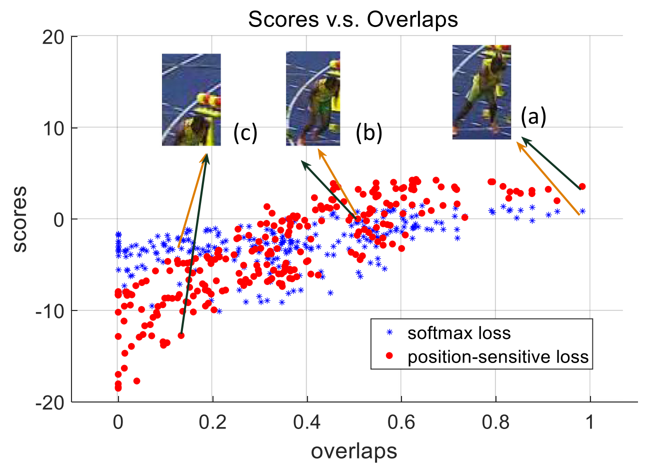


Fig. 6. Scores of the position-sensitive loss and the softmax loss. The horizontal axis is the overlaps between samples and the true object, while the similarity scores between the samples and the true object are shown in the vertical axis. The blue and red dots are used to represent the scores of the corresponding feature embedding trained by the softmax loss and by position-sensitive loss, respectively. The sample (a), (b) and (c) are gathered with the overlaps 0.95, 0.58 and 0.18, respectively. The corresponding scores are 0.2, -5 and -4 by the softmax loss, and 0.4, -5 and -13 by the position-sensitive loss.

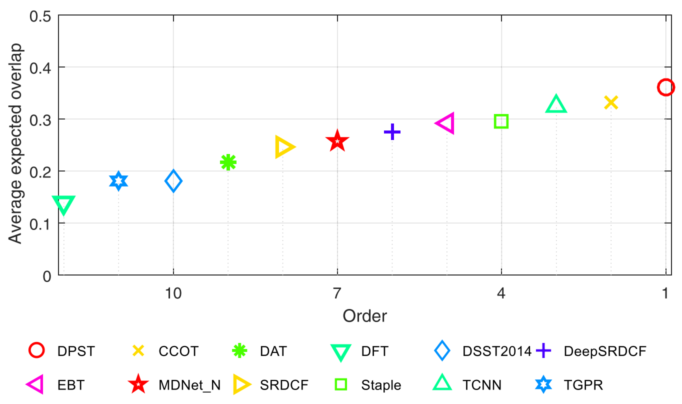


Fig. 7. Average expected overlaps with rank. The X-axis is in descending order. It shows our DPST tracker outperforms the compared trackers greatly on average expected overlaps with a score of 0.3602.

and DFT [49] to demonstrate our advantage. Especially, the recent works [50]–[52] are performed based on the basic MDNet [4] tracker. Besides, the recent trackers based on the shrinkage loss [53] and triplet loss [54] with Siamese network are also compared with the proposed loss.

D. VOT2016 Dataset

By using the VOT toolkit [17], we obtain Figure 7 and Table II to compare our tracker with others on VOT2016 challenge. We also choose a reset-based evaluation methodology which has been applied in VOT challenges [17], where 3 primary measures [38] , e.g. accuracy, robustness and expected average overlap are used to analyze performance.

Figure 7 shows the average expected overlap scores for experiment baseline, and Table II reports the average scores, ranks of accuracy, the number of failures, ranks of robustness, the average scores and the overall rank of the expected overlap scores. As illustrated in Figure 7 and Table II, our tracker obviously outperforms most of the compared trackers and rank excel sightly the TCNN [42] and CCOT [43] in the expected overlap evaluation. Meanwhile, our tracker achieves the second score of 0.9 for the robust score.

The CCOT [43] tracker is a state-of-the-art tracker, which achieved the top rank at the Visual Object Tracking challenge 2016. It has introduced a novel formulation for training continuous convolution filters to pose the learning problem in the continuous spatial domain. The EAO of our method is 0.3602 , while 0.3310 for CCOT [43] tracker. This may be due to the fact that feature embedding adopted in the tracking is different.

Our feature embedding is trained offline to achieve more powerful ability to represent the object and adapt to the target variation through online learning. However, the feature in the CCOT [43] tracker is pre-defined deep features that are learned for the image classification task. TCNN tracker collaborates multiple CNNs to estimate target states and determine the desirable paths for online model updates in the tree. However, it requires making multiple decisions for each frame since only one score of the candidate is generated at a time. It achieves good score 0.9589 on the VOT2016 dataset.

These experimental results and analyses on VOT2016 dataset [17] illuminate that the proposed tracking algorithm performs well against most of the recent state-of-the-art trackers in both accuracy and robustness.

E. OTB Dataset

The one pass evaluation [12] is employed for tracking evaluation because it only needs to initialize the object in the first frame, and the whole tracking process will not be influenced by supervised information until the end. For the metric measurements used in our experiment, the center location error with a threshold of 20 and overlap ratio with a threshold of 0.5 are utilized to generate the precision and success plots. Here, we mainly report the results on the OTB2015 [12] dataset since the other one is a subset.

Overall performance: Tracking algorithms are evaluated on the OTB dataset with both distance precision and overlap success rate [12] using one-pass evaluation (OPE) protocol, where each tracker is evaluated in the initial frame with ground truth until the end of each sequence. Figure 8 shows the plots of the distance precision rate and the overlap success rate with the OPE against other methods on OTB2015 [12]. We list the quantitative results of the distance precision rate at 20 pixels, the area-under-the-curve (AUC) score and tracking speed in Table III. The results show that the proposed algorithm performs well against the state-of-the-art trackers in the area under the curve (AUC) score and distance precision rate (DP) at a threshold of 20 pixels. It can be seen that, compared with the other algorithms, the proposed DPST tracker has achieved the best performance in accuracy and success, which are 91.9% and 68.6%, respectively. Also, our algorithm achieves a certain improvement compared with the baseline MDNet [4] (AUC: 67.8% and DP: 90.9%) and tracker

TABLE II
BASELINE EVALUATION ON VOT2016. THE BASELINE IS A RE-SET BASED EVALUATION ACCURACY AND ROBUSTNESS ARE ANALYZED BY CORRESPONDING SCORES AND RANKS. THE EXPECTED AVERAGE OVERLAP IS ALSO INCLUDED IN THE BASELINE. THE RED, GREEN, AND BLUE NUMBER DENOTE THE 1ST, 2ND, AND 3RD RANK IN EACH ROW RESPECTIVELY

Trackers	Acc. Score	Acc. Rank	Rob. Score	Rob. Rank	EAO
TCNN [43]	0.5391	5.4931	0.9589	5.4000	0.3249
CCOT [44]	0.5226	5.9651	0.8500	5.0167	0.3310
MDNet [4]	0.5331	5.6254	1.2044	5.7333	0.2572
Staple [45]	0.5382	5.6945	1.3500	6.3667	0.2952
DeepSRDCF [5]	0.5134	6.2119	1.1667	6.0750	0.2763
SRDCF[56]	0.5235	6.1801	1.5000	6.5833	0.2471
EBT [46]	0.4407	7.7926	0.9000	5.0583	0.2913
DAT[47]	0.4626	7.3101	1.7167	6.7250	0.2167
DSST2014 [48]	0.4841	6.7834	2.5167	8.1694	0.1814
TGPR[49]	0.4499	7.8484	2.2500	8.3250	0.1811
DFT [50]	0.4327	8.0454	3.5833	9.7194	0.1395
DPST	0.5610	4.9420	0.9000	4.8000	0.3602

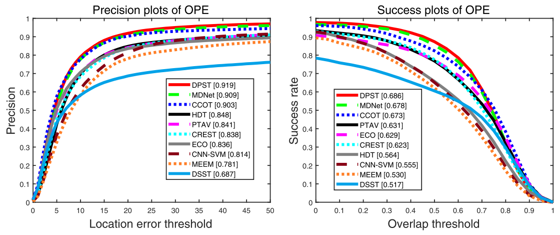


Fig. 8. Distance precision and overlap success plots on the OTB2015 [12] dataset. Quantitative results on the 100 benchmark sequences using OPE. The legend of distance precision contains threshold scores at 20 pixels, while the legend of overlap success contains area-under-the-curve score for each tracker. The proposed algorithm performs favorably against the state-of-the-art trackers.

with the shrinkage loss [53] (AUC: 66.0% and DP: 90.9%). Overall, the proposed algorithm performs favorably against the state-of-the-art methods on the OTB2015 dataset [12].

Attribute-based evaluation: We analyze the tracker performance using 11 annotated attributes in the OTB2015 [12] dataset: illumination variation (IV35), out-of-plane rotation (OR59), scale variation (SV61), occlusion (OCC44), deformation (DEF39), motion blur (MB29), fast motion (FM37), in-plane rotation (IR51), out-of-view (OV14), background clutter (BC31), and low resolution (LR9) (number of videos for each attribute is appended to the end of each abbreviation). Figure 9 and Figure 10 present the results under one-pass evaluation regarding these challenging attributes and the proposed tracker is able to handle the challenges of motion blur, fast motion, background cluster, out-of-plane rotation, occlusion, illumination variation, in-plane rotation, and out-of-view very well. This should be attributed to the feature embedding robustness against the deformation and rotation based on the position-sensitive loss.

Quantitative evaluation: Figure 11 shows the qualitative comparisons with the performing tracking methods: MDNet [4], CCOT [43], MEEM [58], CNN-SVM [64], ECO [65], DSST [47], PTAV [56] and the proposed method on ten challenging image sequences including *bird1, box, diving, jumping, motor-Rolling, matrix, basketball, trellis and bolt2*. Overall, our tracker is able to locate the object well in these complex scenes.

TABLE III
COMPARISONS WITH STATE-OF-THE-ART TRACKERS ON OTB2015 [12] BENCHMARK SEQUENCES. OUR METHOD PERFORMS FAVORABLY AGAINST EXISTING METHODS IN THE AREA UNDER THE CURVE (AUC) SCORE AND DISTANCE PRECISION RATE (DP) AT A THRESHOLD OF 20 PIXELS. THE FIRST AND SECOND BEST VALUES ARE HIGHLIGHTED BY RED AND BLUE COLORS, RESPECTIVELY.

Trackers	AUC(%)	DP(%)	Speed(FPS)
CCOT [44]	67.3	90.3	0.3
SANet [51]	69.2	92.8	1
DSLT [54]	66.0	90.9	5
HDT [11]	56.4	84.8	10
PTAV [57]	63.1	84.1	25
CREST [58]	62.3	83.8	1
MDNet[4]	67.8	90.9	1
MEEM[59]	53.0	78.1	10
DSST [48]	51.7	68.7	24
TCNN [8]	65.4	88.4	1
ADNet [60]	64.6	88.0	3
VITAL [61]	68.2	91.7	1
DeepSRDCF [62]	56.0	85.1	0.6
CF2 [10]	56.2	83.7	11
KCF [8]	40.3	69.3	172
DAT [52]	66.8	89.5	1
RT-MDNet [53]	65.0	88.5	25
DaSiamRPN [63]	66.8	88.0	160
SiamRPN [64]	63.7	85.1	200
SiamFC-tri [55]	59.0	78.1	55.4
DPST	68.6	91.9	1

F. Failure Analysis

We show some tracking failures by the proposed tracker in Figure 12. For the *diving* and *jump* sequences, the person in the presence of drastic in-plane changes and fast motion, the model can not adjust to the object quickly yet to the background. In the *Bike* sequence, the search region is not enough to cover the object since the object re-appears far away from where it disappears. For the *Coupon* sequence, the distractor that has the same semantics with the object is around the object, thus our approach cannot identify the distractor.

Experiments show that the performance improvement of our tracker on the tracking dataset, especially on the VOT2016 dataset. This dataset is more complex than the OTB2015 dataset, where the scale, rotation, and deformation of the object are

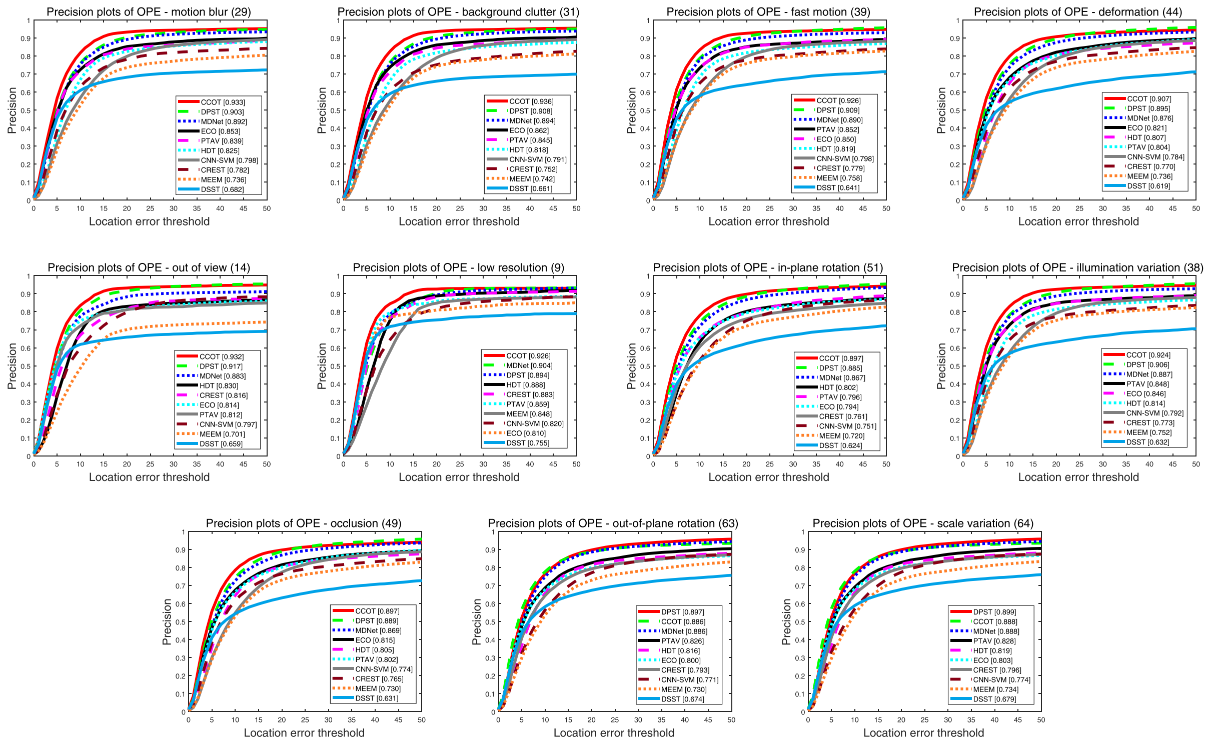


Fig. 9. Attribute-based distance precision metric on the OTB2015 dataset [12], where the legend of distance precision contains threshold scores at 20 pixels. Performance evaluation on benchmark attributes: illumination variation (35), out-of-plane rotation (59), scale variation (61), occlusion (44), deformation (39), motion blur (29), fast motion (37), in-plane rotation (51), out-of-view (14), background clutter (31), and low resolution (9). The later digits mean the number of videos with that attribute. The proposed algorithm performs well against state-of-the-art results.

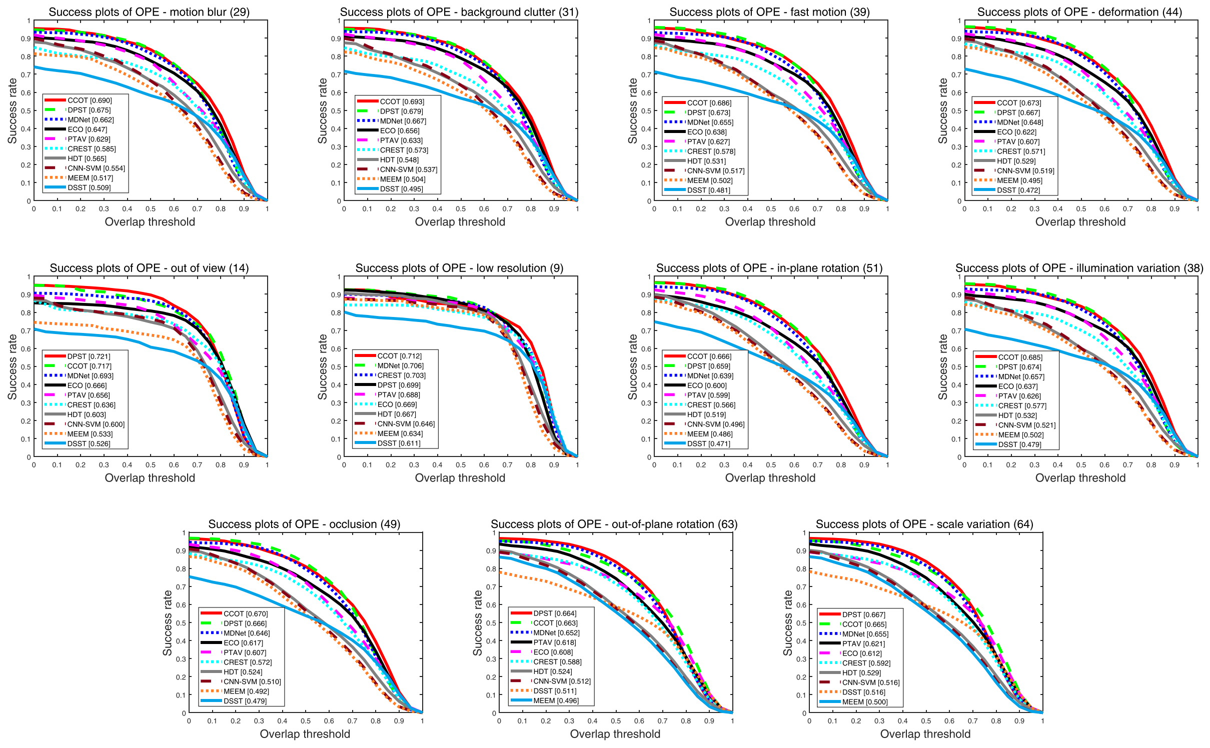


Fig. 10. Attribute-based distance precision metric on the OTB2015 dataset [12], where the legend of overlap success contains area-under-the-curve score for each tracker. Performance evaluation on benchmark attributes: illumination variation (35), out-of-plane rotation (59), scale variation (61), occlusion (44), deformation (39), motion blur (29), fast motion (37), in-plane rotation (51), out-of-view (14), background clutter (31), and low resolution (9). The later digits mean the number of videos with that attribute. The proposed algorithm performs well against state-of-the-art results.

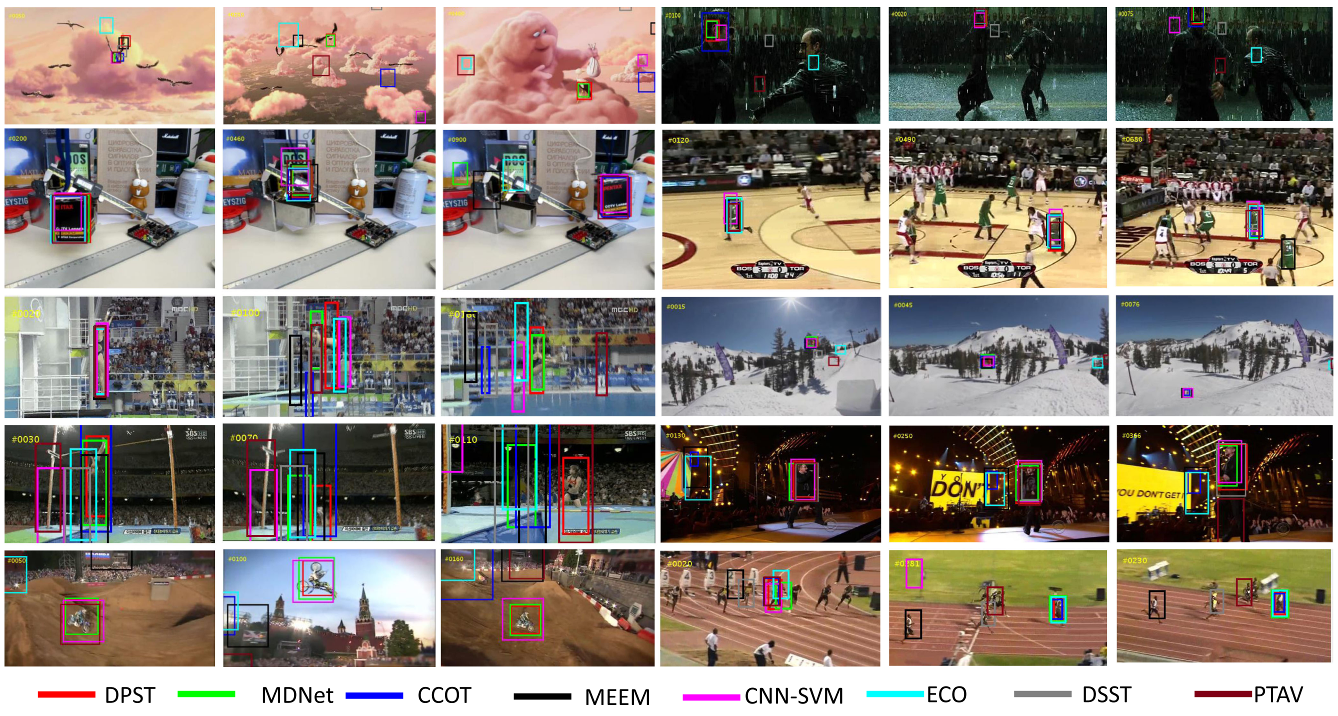


Fig. 11. Sample tracking results on challenging image sequences (from left to right and top to down are *bird1, box, diving, jumping, motorRolling, matrix, basketball, trellis,and bolt2*), respectively. We show some tracking results of MDNet [4], CCOT [43], MEEM [58], CNN-SVM [64], ECO [65], DSST [47], PTAV [56] methods, as well as the proposed algorithm.

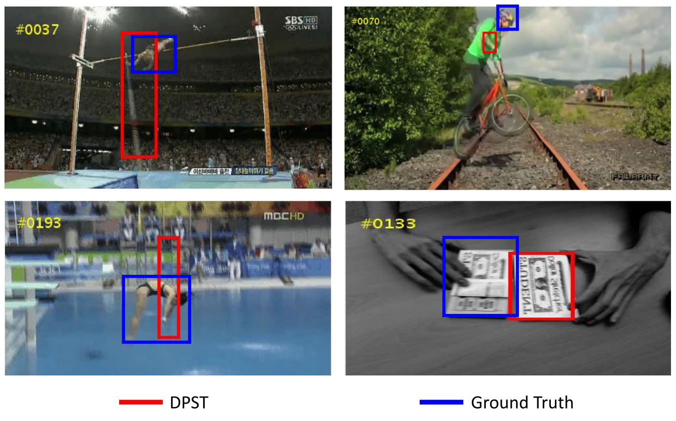


Fig. 12. Failure cases on the *diving, bike, jump and Coupon* sequences on the OTB2015 [12] . Red boxes show our results and the blue ones are ground truth.

more challenging. In summary, it is sufficient to prove that the position-sensitive loss can improve location accuracy.

VI. Conclusion

In this paper, two position margins are designed coupled with the softmax loss to learn discriminative feature embedding by the network for visual tracking task. Since the semantic and position attribute of samples is taken into account, the new loss introduces intra-class compactness and relative ranking into the softmax loss to achieve an accurate location for the unseen object. The experimental results show that the proposed tracker exceeds the comparison trackers and achieves excellent performance on the OTB2015[12] and VOT2016 [38].

We also summarize the potential directions to improve our approach and shed light on our future works. The recent tracking datasets such as TrackingNet [66], LaSOT [67] can also be used to train the network. Besides, the speed of our algorithm is not satisfactory for the real-time application. The reason is that our method requires sampling candidate regions, which are passed through a CNN pre-trained on a large-scale dataset and fine-tuned at the first frame in a test video. Since every candidate is processed independently, it suffers from high computational complexity in terms of time and space. Like the Mask R-CNN, RoIAlign can be employed to accelerate feature extraction procedure.

Acknowledgment

Thanks to Yule Zhang and Peng Zhu for their discussion and help.

References

[1] F. Liu *et al.*, “Robust visual tracking revisited: From correlation filter to template matching,” *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2777–2790, Jun. 2018.
[2] H. Hu *et al.*, “Robust object tracking using manifold regularized convolutional neural networks,” *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 510–521, Feb. 2019.
[3] N. Liang, G. Wu, W. Kang, Z. Wang, and D. D. Feng, “Real-time long-term tracking with prediction-detection-correction,” *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2289–2302, Sep. 2018.
[4] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2016, pp. 4293–4302.

[5] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. IEEE Int. Conf. Comput. Vision Workshops*, 2015, pp. 2837–2846.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Int. Conf. Learn. Representations*, May 2015.

[7] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2009, pp. 248–255.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[9] R. Yao, S. Xia, Z. Zhang, and Y. Zhang, "Real-time correlation filter tracking by efficient dense belief propagation with structure preserving," *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 772–784, Apr. 2017.

[10] C. Ma, J. Huang, X. Yang, and M. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 3074–3082.

[11] Y. Qi *et al.*, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 4303–4311.

[12] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[13] I. Gonzlez-Daz, M. Birinci, F. Daz-de-Mara, and E. J. Delp, "Neighborhood matching for image retrieval," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 544–558, Mar. 2017.

[14] J. Liang, Q. Hu, W. Wang, and Y. Han, "Semisupervised online multikernel similarity learning for image retrieval," *IEEE Trans. Multimedia*, vol. 19, no. 5, pp. 1077–1089, May 2017.

[15] W. Liu, Y. Wen, Z. Yu, and M. M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. IEEE Int. Conf. Mach. Learn.*, 2016, pp. 507–516.

[16] J. Deng, J. Guo, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 4690–4699.

[17] M. Kristan *et al.*, "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2137–2155, Nov. 2016.

[18] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognit.*, vol. 76, pp. 323–338, Apr. 2018.

[19] S. He, Q. Yang, R. W. H. Lau, J. Wang, and M. Yang, "Visual tracking via locality sensitive histograms," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2013, pp. 2427–2434.

[20] K. Chen and W. Tao, "Learning linear regression via single-convolutional layer for visual object tracking," *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 86–97, Jan. 2019.

[21] Q. Wang, C. Yuan, J. Wang, and W. Zeng, "Learning attentional recurrent neural network for visual tracking," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 930–942, Apr. 2019.

[22] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1420–1429.

[23] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vision*, 2016, pp. 8749–8765.

[24] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1834–1848, Apr. 2016.

[25] X. Wang *et al.*, "Online scale adaptive visual tracking based on multilayer convolutional features," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 146–158, Jan. 2019.

[26] K. Chen and W. Tao, "Once for all: A two-flow convolutional neural network for visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 12, pp. 3377–3386, Dec. 2018.

[27] X. Yang *et al.*, "Deep relative attributes," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1832–1842, Sep. 2016.

[28] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vision Workshops*, 2016, pp. 850–865.

[29] R. P. Pflugfelder, "Siamese learning visual tracking: A survey," 2017, arXiv:1707.00569.

[30] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.

[31] J. Cui, Y. Liu, Y. Xu, H. Zhao, and H. Zha, "Tracking generic human motion via fusion of low- and high-dimensional approaches," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 996–1002, Jul. 2013.

[32] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 3361–3368.

[33] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[34] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vision*, 2016, pp. 815–823.

[35] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L2 hypersphere embedding for face verification," in *Proc. ACM Int. Conf. Multimedia*, 2017, pp. 1041–1049.

[36] Y. Liu, H. Li, and X. Wang, "Rethinking Feature Discrimination and Polymerization for Large-scale Recognition," *Adv. Neural Inf. Process. Syst. (NIPS) deep learn. workshop*, 2017.

[37] A. Vedaldi and K. Lenc, "MatConvNet—Convolutional neural networks for MATLAB," in *Proc. ACM Int. Conf. Multimedia*, 2015, pp. 689–692.

[38] L. Cehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1261–1274, Mar. 2016.

[39] A. W. M. Smeulders *et al.*, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[41] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2013, pp. 2411–2418.

[42] H. Nam, M. Baek, and B. Han, "Modeling and propagating CNNs in a tree structure for visual tracking," *Eur. Conf. Comput. Vision (ECCV) vot workshop*, 2016.

[43] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vision*, 2016, pp. 472–488.

[44] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1401–1409.

[45] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 943–951.

[46] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 2113–2120.

[47] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1561–1575, Aug. 2017.

[48] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. Eur. Conf. Comput. Vision*, 2014, pp. 188–203.

[49] L. Sevilla-Lara and E. G. Learned-Miller, "Distribution fields for tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2012, pp. 1910–1917.

[50] H. Fan and H. Ling, "SANet: Structure-aware network for visual tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit Workshops*, 2017, pp. 2217–2224.

[51] S. Pu, Y. Song, C. Ma, H. Zhang, and M.-H. Yang, "Deep attentive tracking via reciprocative learning," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 1935–1945.

[52] I. Jung, J. Son, M. Baek, and B. Han, "Real-time MDNet," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 89–104.

[53] X. Lu *et al.*, "Deep regression tracking with shrinkage loss," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 369–386.

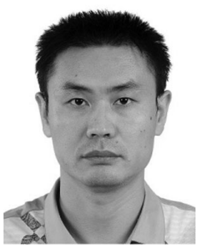
[54] J. S. Xingping Dong, "Triplet loss in Siamese network for object tracking," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 472–488.

[55] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 4310–4318.

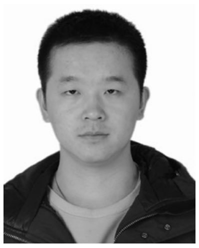
[56] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 5487–5495.

[57] Y. Song *et al.*, "CREST: Convolutional residual learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 2574–2583.

[58] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Proc. Eur. Conf. Comput. Vision*, 2014, pp. 188–203.
[59] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 1349–1358.
[60] Y. Song *et al.*, "Vital: Visual tracking via adversarial learning," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 8990–8999.
[61] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1430–1438.
[62] Z. Zhu *et al.*, "Distractor-aware Siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 103–119.
[63] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with Siamese region proposal network," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 8971–8980.
[64] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proc. IEEE Int. Conf. Mach. Learn.*, 2015, pp. 597–606.
[65] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 6931–6939.
[66] M. Mller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem, "TrackingNet: A large-scale dataset and benchmark for object tracking in the wild," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 310–327.
[67] H. Fan *et al.*, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 5374–5383.



Yufei Zha received the Ph.D. degree in information and communication engineering from Air Force Engineering University, Xi'an, China, in 2009. He is currently an Associate Professor with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His current research interests include object detection, visual tracking, and machine learning.



Tao Ku is currently working toward the Ph.D. degree at the Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands. His research interests include deep learning, 3-D computer vision, and visual object tracking.

Yunqiang Li received the B.S. degree in electric engineering and M.S. degree in computer science and technology from the School of Air Force Engineering University, Xi'an, China, in 2014 and 2017, respectively. Since 2018, he has been working toward the Ph.D. degree at the Vision Lab, Delft University of Technology, Delft, The Netherlands. His research interests include deep learning, computer vision, and object tracking.

Peng Zhang received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2001, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2011. He is currently a Professor with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His current research interests include object detection and tracking, computer vision, and pattern recognition. He has authored/coauthored more than 80 high ranked international conference and journal papers. He is a member of ACM.