STCT: Sequentially Training Convolutional Networks for Visual Tracking

Lijun Wang[1,2], Wanli Ouyang[2], Xiaogang Wang[2], and Huchuan Lu[1]
[1]Dalian University of Technology, China
[2]The Chinese University of Hong Kong, Hong Kong, China

Abstract

Due to the limited amount of training samples, fine-tuning pre-trained deep models online is prone to over-fitting. In this paper, we propose a sequential training method for convolutional neural networks (CNNs) to effectively transfer pre-trained deep features for online applications. We regard a CNN as an ensemble with each channel of the output feature map as an individual base learner. Each base learner is trained using different loss criterions to reduce correlation and avoid over-training. To achieve the best ensemble online, all the base learners are sequentially sampled into the ensemble via important sampling. To further improve the robustness of each base learner, we propose to train the convolutional layers with random binary masks, which serves as a regularization to enforce each base learner to focus on different input features.

The proposed online training method is applied to visual tracking problem by transferring deep features trained on massive annotated visual data and is shown to significantly improve tracking performance. Extensive experiments are conducted on two challenging benchmark data set and demonstrate that our tracking algorithm can outperform state-of-the-art methods with a considerable margin.

1. Introduction

Visual tracking is a fundamental problem in computer vision that has been receiving a rapidly growing attention. It has a variety of subfields ranging from single-target to multi-target tracking. The focus here is single-target, model-free online tracking [20, 12, 38], where a category agnostic target is indicated by a bounding box in the first frame, and the tracker aims at locating the target in the following each frame. Due to significant target appearance changes caused by abrupt motion, deformation, occlusion and illumination variation, visual tracking is still a very challenging problem.

Prior approaches [11, 15, 40, 41] rely on hand-crafted features to describe the target and have addressed the above challenging factors to a certain extend. However, these hand-crafted features are designed for certain scenarios. Thus, they can not generalize well and are incapable to capture the semantic information of the target, which can easily lead to tracking failure in challenging conditions.

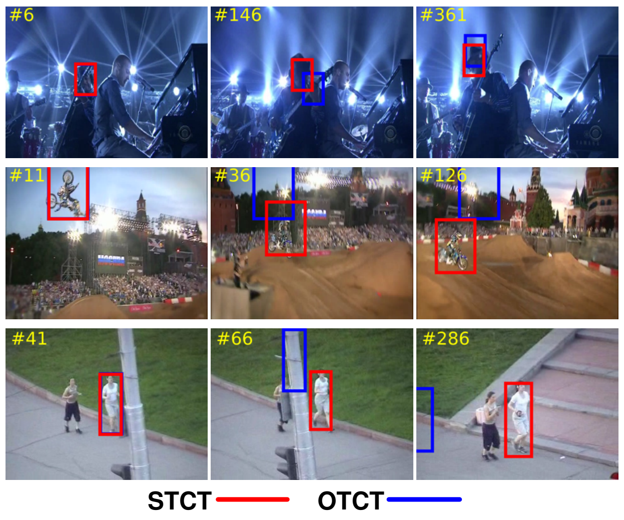

Figure 1. Due to limited number of training samples, online training CNNs for tracking (denoted as OTCT) can easily lead to over-fitting and cause tracking failure. We propose a sequential learning method (denoted as STCT) for CNNs to address this issue.

Recently, deep Convolutional Neural Networks (CNNs) trained on large scale image classification data sets (*e.g.* [5]) have demonstrated great success in many vision tasks [29, 9, 3, 25]. These semantic representations discovered by the learning process are shown to be very effective at distinguishing objects of various categories. However, supervised training of deep CNNs with millions of parameters entails a large number of annotated training samples. To apply deep CNNs for tasks with a limited amount of training samples, previous approaches [24, 30] adopt a transfer learning method by first pre-training a deep CNN on a source task with a large scale training data set and then fine-tuning the learned feature on the target task. Due to the good generalization capability of CNN features across different data sets, this transfer learning approach is effective and has shown state-of-the-art performance.

However, for online visual tracking, the lack of training samples becomes even more severe, since the only training sample with ground truth label is provided in the first frame, and the tracking results used for updating the tracker are also obtained in a sequential manner. Thus, directly online fine-tuning a pre-trained deep CNN is prone to over-fitting, which will degrade the tracker and gradually leads to tracking drift (See Figure 1 as an example).

In order to address the above issue, we propose a sequential training method for CNNs to effectively transfer pre-trained deep features for online visual tracking. Specifically, a CNN is regarded as an ensemble, while each channel of the convolutional feature map is treated as a base learner and is updated using a different loss criterion, such that they are not highly correlated with each other. Online fine-tuning of the CNN is then formulated as a sequential ensemble learning problem. To build the best ensemble, we sequentially select the base learners via important sampling and add them into the ensemble. Online tracking is conducted as foreground/background separation by the sequentially learned ensemble. To further reduce over-fitting, we propose to train the convolutional layers with random binary masks, which can effectively enforce different convolutional kernels to focus on different target parts.

The contribution of this paper can be summarized into three-folds: i) we propose a sequential training method for CNNs, which can effectively transfer pre-trained deep features for online application and reduce over-fitting; ii) we develop an effective visual tracking algorithm based on the proposed sequential learning method, where deep features trained on the Imagenet image classification task [5] are utilized to predict the position and scale of the target simultaneously; iii) extensive experiments are conducted on two popular benchmark data sets and demonstrate that the proposed tracking algorithm performs favorably against state-of-the-art methods.

2. Related Work

A typical tracking method mainly contains two important components: an appearance model to estimate the likelihood of target candidates, and a search strategy to find the most likely target location. In this paper, we mainly focus on the design of a robust appearance model. In some prior methods [2, 23, 32, 31], the target appearance is represented by generative models and the candidate with the maximum likelihood is predicted as the target. The ”EigenTracking” algorithm [2] utilizes pre-trained eigen basis to describe the target appearance. Later on, Ross *et al.* [27] propose to incrementally update both the eigenbasis and mean to adapt to target appearance changes. Sparse representation has also been applied to tracking [23, 33], where the target is reconstructed by a sparse combination of target templates.

Meanwhile, some methods cast visual tracking as a foreground and background separation problem using discriminative models. Online learning algorithms based on boosting [10], structured SVM [11], multiple instance learning [1], and correlation filters [4, 12] are applied in tracking and achieve good performance. Among others, Danelljan *et al.* [4] propose to estimate the scale changes of the target using correlation filters learned from HOG features. Our method bears a similar spirit with [4] in predicting target scale changes. However, ours differs from [4] in that we exploit a scale prediction network trained on deep features which are more robust to significant appearance changes.

Deep convolutional neural networks have improved state-of-the-art performance in many computer vision applications [19, 30, 34, 28, 39, 26] in recent years. Existing methods have also explored the usage of CNNs in online tracking. In [21], a three-layer CNN is trained on-line. Without pre-training and with limited training samples obtained online, CNN fails to capture object semantics and is not robust to deformation. In [36], a deep autoencoder is first pre-trained offline and then finetuned for binary classification in online tracking. Since the pre-training is performed in an unsupervised way by reconstructing gray images with very low resolutions, the learned deep feature has limited discriminative power for tracking. Both [21] and [36] train deep networks online with limited training samples, and inevitably suffer from over-fitting. Consequently, they only achieve comparable or even inferior performance against state-of-the-arts. More recent methods [14, 35, 22] adopt deep convolution networks trained on a large scale image classification task [5] to improve tracking performance. [14] predicts saliency maps using deep features. [35] and [22] propose to estimate foreground heat maps by training either CNNs or correlation filters using feature maps of multiple convolution layers. In contrast, we provide a new paradigm to transfer rich features of pre-trained deep CNNs for online tacking. Instead of directly finetuning deep features like [36, 35], we cast online training CNN as learning ensembles to effectively remove feature correlation and avoid over-fitting.

3. CNN Training as Ensemble Learning

Before elaborating the proposed sequential training method for CNNs, we first introduce some background of ensemble learning to put our method in a proper context.

3.1. Sequential Sampling for Ensemble Learning

Given a data point $\boldsymbol{x}$, the goal of supervised learning is to predict the likely value $\hat{y}$ of the ground truth value y associated with $\boldsymbol{x}$. The prediction rule is often defined as a function $\hat{y} = F(\boldsymbol{x})$ that can be learned by minimizing the expected loss over all the training data $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_1^N$

$$F^*(\boldsymbol{x}) = \arg\min_{F(\boldsymbol{x})} \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\boldsymbol{x}_i)), \qquad (1)$$

where $L(y, \hat{y})$ indicates the loss of predicting a value $\hat{y}$ when the true value is y. The prediction function can have various forms depending on the particular problem to be handled. In [7], Friedman and Popescu formulate the prediction function as an integral

$$F(\boldsymbol{x}) = \pi_0 + \int_{\Gamma} \pi(\boldsymbol{\gamma}) f(\boldsymbol{x}; \boldsymbol{\gamma}) d\boldsymbol{\gamma}, \tag{2}$$

where $f(\boldsymbol{x}; \boldsymbol{\gamma})$ denotes a base learner parameterized by $\boldsymbol{\gamma} \in \Gamma$, and $\pi(\boldsymbol{\gamma})$ is the coefficient function. Numerical quadrature is utilized to approximate (2) by a linear combination of base learners at M evaluation points $\boldsymbol{\gamma}_m \in \Gamma$ as $F(\boldsymbol{x}) \simeq a_0 + \sum_{m=1}^{M} a_m f(\boldsymbol{x}; \boldsymbol{\gamma}_m)$. The learning problem in (1) then amounts to choosing a good ensemble of evaluation points $\{\boldsymbol{\gamma}_m\}_1^M$ and their corresponding coefficients $\{a_m\}_0^M$.

For a base learner $f(\boldsymbol{x}; \boldsymbol{\gamma})$, its irrelevance to the current problem is defined as

$$Q(\boldsymbol{\gamma}) = \min_{\alpha_0, \alpha} \frac{1}{N} \sum_{i=1}^{N} L(y_i, \alpha_0 + \alpha f(\boldsymbol{x}; \boldsymbol{\gamma})). \tag{3}$$

The optimal single point can then be obtained by minimizing the irrelevance as $\boldsymbol{\gamma}^* = \arg\min Q(\boldsymbol{\gamma})$. For an ensemble of base learners $\{f(\boldsymbol{x}; \boldsymbol{\gamma}_m)\}_1^M$, the characteristic scale can be employed to measure its quality

$$\sigma = \frac{1}{M} \sum_{m=1}^{M} \left[Q(\boldsymbol{\gamma_m}) - Q(\boldsymbol{\gamma}^*) \right]. \tag{4}$$

A small value of σ implies that many base learners in the ensemble are very similar to the optimal base learner $f(\boldsymbol{x}; \boldsymbol{\gamma}^*)$. Since they are highly correlated with each other, the ensemble of base learners fail to provide additional information beyond the single optimal base learner $f(\boldsymbol{x}; \boldsymbol{\gamma}^*)$. On the contrary, if σ is too large, most of the base learners are irrelevant to the problem which will ultimately degrade the performance of the ensemble.

Based on the above observation, Friedman and Popescu [7] propose a sequential sampling method to generate an ensemble of evaluation points, in which the irrelevance measure of each successive point $\boldsymbol{\gamma}_m$ depends on the previously sampled points $\{\boldsymbol{\gamma}_l\}_1^{m-1}$

$$Q_m(\boldsymbol{\gamma} | \{\boldsymbol{\gamma}_l\}_1^{m-1}) = \min_{\alpha_0, \alpha_m} \frac{1}{N} \sum_{i=1}^{N} L\Big(y_i, \alpha_0 + \alpha_m f(\boldsymbol{x}_i; \boldsymbol{\gamma}) + \eta \sum_{l=1}^{m-1} \alpha_l f(\boldsymbol{x}_i; \boldsymbol{\gamma}_l) \Big), \tag{5}$$

where the parameter η controls the impact of previously sampled points on the current relevance measure. Then each sequentially selected parameter point $\boldsymbol{\gamma}_m$ is determined by

$$\boldsymbol{\gamma}_m = \arg\min_{\boldsymbol{\gamma} \in \Gamma} Q_m(\boldsymbol{\gamma} | \{\boldsymbol{\gamma}_l\}_1^{m-1}). \tag{6}$$

As the sampling proceeds, the irrelevance defined by (5) will increasingly differ from that for the single parameter point (3). Consequently, the sampled parameter points will not be highly correlated with each other. We refer the readers to [7] for more information.

3.2. Online Training CNNs as Sequential Ensemble Learning

For online applications, one simple approach to transfer offline pre-trained CNN features is to add one or more randomly initialized CNN layers, named as adaptation layers, on top of the pre-trained CNN model. Then keep the parameters, *i.e.* convolutional kernels and bias, of the pre-trained CNN fixed and only learn the parameters of the adaptation layers online to fit the current task. However, we empirically observe that this transfer learning method suffers from severe over-fitting. The online learned parameters mainly focus on recent training samples and are less likely to well generalize to both historical and future samples. This phenomenon can be fatal to online visual tracking where the target often undergoes significant appearance changes or heavy occlusion.

To tackle the above issue, we propose to train a CNN model online as sequentially learning ensembles to better transfer pre-trained deep features. Denote the pre-trained CNN as *CNN-E*, which takes the RGB image as input and outputs a convolutional feature map $\boldsymbol{X}$. An online adapted CNN, named as *CNN-A* is randomly initialized and consists of two convolutional layers interleaved with an ReLU layer as the nonlinear activation unit. It takes the feature map $\boldsymbol{X}$ as input and generates the final feature map $\{\boldsymbol{F}_2^c(\boldsymbol{X}) | c = 1, 2, \ldots, C_2\}$, where $\boldsymbol{F}_2^c(\boldsymbol{X}) \in \mathbb{R}^{m \times n}$ indicates the c-th channel of the feature map generated by the second layer with spatial size of $m \times n$.

The feature map in the second layer is obtained by convolving the kernel with the feature map in the first layer as

$$\boldsymbol{F}_2^c(\boldsymbol{X}) = \sum_{k=1}^{C_1} \boldsymbol{w}_k^c * \boldsymbol{F}_1^k(\boldsymbol{X}) + b_c, \tag{7}$$

where C_1 denotes the number of channels of the feature map output by the first layer; $\boldsymbol{w}_k^c$ represents the convolution kernel connecting the k-th channel of the first layer feature map with the c-th channel of the second layer feature map; b_c is the bias and $*$ denotes convolution operation; the summation is conducted element-wisely. In order to introduce randomness into the parameter learning process, we regard the output feature map as an ensemble of base learners $\boldsymbol{F}_2^c(\boldsymbol{X}) = \sum_{k=1}^{C_1} \boldsymbol{f}(\boldsymbol{X}; \boldsymbol{\gamma}_k^c)$, where each base leaner is defined as $\boldsymbol{f}(\boldsymbol{X}; \boldsymbol{\gamma}_k^c) = \boldsymbol{w}_k^c * \boldsymbol{F}_1^k(\boldsymbol{X}) + b_c^k$, and the parameter $\boldsymbol{\gamma}_k^c$ indicates the corresponding kernel weights and bias in both the first layer (*i.e.*, weights and bias of $\boldsymbol{F}_1^k(\boldsymbol{X})$)and the second layer (*i.e.*, $\boldsymbol{w}_k^c$ and b_c^k) of *CNN-A*. The online training

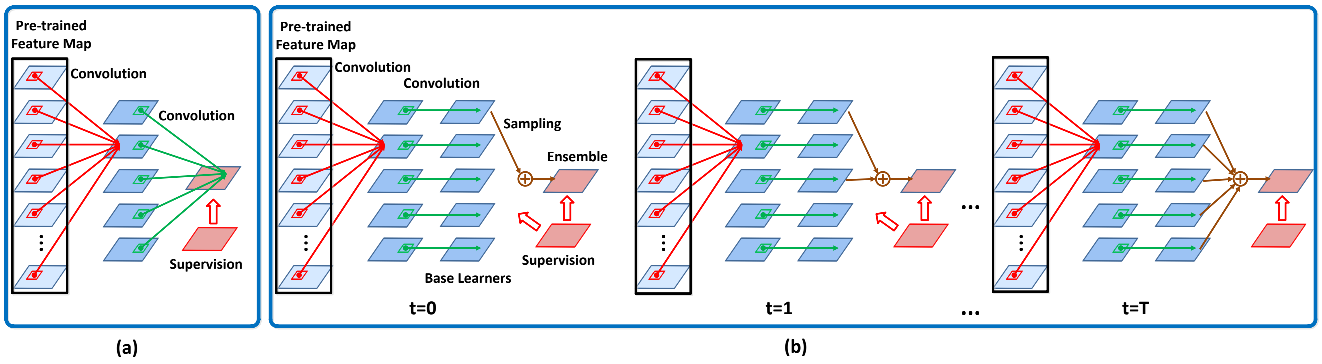

Figure 2. Illustration of the proposed sequential training method for CNNs. (a) Conventional method for training a two-layer CNN online to transfer pre-trained deep features. (b) The proposed method trains the CNN model via sequentially sampling optimal base learners into an ensemble.

of the *CNN-A* network is then equivalent to online updating each base learner and sequentially sample an optimal set of base learners into the ensemble. Since our proposed online training method is conducted independently in each channel of the output feature map, in the following discussion, we will take one output channel as an example to describe the training method. For notational simplicity, we omit the superscript channel number and use $\{\gamma_k | k = 1, 2, \ldots, C_1\}$ to denote the parameters of the base learners for any one output feature map channel.

At the beginning of the online training process, the parameter γ_k of each base learner is randomly initialized and independently trained using the first training sample by stochastic gradient descent (SGD). The parameter $\boldsymbol{\gamma}^*$ with the smallest training error is selected as the single optimal parameter and added to the ensemble set $\mathcal{E}$, while the rest $C_1 - 1$ parameters constitute the candidate set $\mathcal{C}$.

In the following training process, the parameters in the candidate set is sequentially added to the ensemble set. All the parameters in the ensemble set are used to form an ensemble with output $\boldsymbol{F}(\boldsymbol{X}; \mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{\gamma_i \in \mathcal{E}} \boldsymbol{f}(\boldsymbol{X}; \gamma_i)$ for online testing.

At the t-th time step, a new training sample $\boldsymbol{X}_t$ with target output $\boldsymbol{Y}_t$ is obtained. The parameters in the ensemble set $\mathcal{E}$ is jointly updated by SGD using the loss function $L_{\mathcal{E}} = L(\boldsymbol{Y}_t, \boldsymbol{F}(\boldsymbol{X}_t; \mathcal{E}))$. Meanwhile, each parameter $\gamma_j \in \mathcal{C}$ is updated independently by SGD using the following loss function

$$L_{\mathcal{C}}(\boldsymbol{Y}_t, \boldsymbol{f}(\boldsymbol{X}_t; \boldsymbol{\gamma}_j)) = L(\boldsymbol{Y}_t, \boldsymbol{f}(\boldsymbol{X}_t; \boldsymbol{\gamma}_j) + \eta \boldsymbol{F}(\boldsymbol{X}_t; \mathcal{E})) \quad (8)$$

where $\boldsymbol{F}(\boldsymbol{X}_t; \mathcal{E})$ is fixed and the parameter η is used to balance the impact of the ensemble on the candidate base learners, such that the update of the base learner parameter $\gamma_j \in \mathcal{C}$ considers both the target output $\boldsymbol{Y}_t$ and the output of the ensemble $\boldsymbol{F}(\boldsymbol{X}_t; \mathcal{E})$. If the training error $L_{\mathcal{E}}$ is higher than a predefined threshold and the candidate set $\mathcal{C}$ is not empty, a new base learner parameter is sampled from the candidate set $\mathcal{C}$ according to the following sampling probability density

$$p(\boldsymbol{\gamma}) = q(L_{\mathcal{C}}(\boldsymbol{Y}_t, \boldsymbol{f}(\boldsymbol{X}_t; \boldsymbol{\gamma}))), \ \boldsymbol{\gamma} \in \mathcal{C} \quad (9)$$

where $q(\cdot)$ is a monotonically decreasing function of its argument. And the sampled parameter is removed from the candidate set $\mathcal{C}$ and added into the ensemble set $\mathcal{E}$.

The above online training approach is conducted sequentially in each time step as illustrated in Figure 2 (b). When all the base learner parameters are sampled from the candidate set to the ensemble set, the ensemble $\boldsymbol{F}(\boldsymbol{X}; \mathcal{E})$ evolves into a complete CNN model (Figure 2 (b), $t = T$). The parameters of this CNN model are trained using different loss criterions and thus demonstrate a moderate diversity, which is empirically shown in our experiments to improve performance and reduce over-fitting.

3.3. Convolutional with Mask Layer

Dropout [13] is commonly used to regularize the deep neural networks by randomly setting a subset of activations in a fully connected layer into zero. However, this regularization is not suitable for convolutional layers. As an alternative, SpatialDropout is proposed in [30] to improve generalization performance for convolutional layers, which sets all the values across the randomly selected channels of the feature map into zeros. This regularization is effective for offline training. However, we find in our initial experiments that randomly ”dropping-out” all the activations in a subset of feature map channels sometimes leads to divergence when training the CNN online with limited amount of training samples.

Instead, we propose a convolutional with mask layer, which aims at further reducing the correlation between the learned features and preventing over-training. Specifically, each channel of the output feature map is associated with an individual binary mask which has the same spatial size with the input feature map. All the masks are initialized in a

Algorithm 1 Online tracking algorithm

```
Input: Initial target location $\boldsymbol{p}_1$, pre-trained CNN-E and
    random initialized CNN-A.
Output: Predicted target location $\boldsymbol{p}_t$.
 1: Initialize each base learner via (11).
 2: $\mathcal{E} \leftarrow \{\gamma^*\}, \mathcal{C} \leftarrow \{\gamma_j | \gamma_j \neq \gamma^*,\ j = 1, 2, \ldots, 100\}$.
 3: Initialize SPN via (12).
 4: repeat
 5:     Crop region $\boldsymbol{I}_t$ at last location and extract feature
        map $\boldsymbol{X}_t$.
 6:     Predict heat map $\hat{\boldsymbol{M}}_t = \frac{1}{|\mathcal{E}|} \sum_{\gamma_i \in \mathcal{E}} \boldsymbol{f}(\boldsymbol{X}_t; \gamma_i)$ with
        confidence $conf_t$.
 7:     Crop region $\hat{\boldsymbol{I}}_t$ at predicted location and extract fea-
        ture map $\hat{\boldsymbol{X}}_t$.
 8:     Predict target scale as $\hat{s} = \arg\max_{s_l \in \mathcal{S}} F_S(\mathcal{T}(\hat{\boldsymbol{X}}_t, s_l))$.
 9:     if $conf_t > \theta$ then
10:         Update base learner and SPN via (13) and (12).
11:         if $L_{\mathcal{E}} > \varepsilon$ and $\mathcal{C} \neq \emptyset$ then
12:             Sample $\hat{\gamma}^*$ from $\mathcal{C}$ via (14).
13:             $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{\hat{\gamma}^*\}, \mathcal{C} \leftarrow \mathcal{C} / \{\hat{\gamma}^*\}$.
14:         end if
15:     end if
16: until end of video sequence.
```

random manner and then fixed throughout the online training process. The forward propagation of the convolutional layer at the training stage is then conducted as

$$\boldsymbol{F}^c = \sum_{k=1}^{K} \boldsymbol{w}_k^c * (\boldsymbol{M}^c \odot \boldsymbol{X}^k) + b_c, \tag{10}$$

where $\boldsymbol{X}^k$ indicates the k-th channel of the input feature map; $\boldsymbol{M}^c$ denotes the binary mask associated with the c-th channel of the output feature map $\boldsymbol{F}^c$; and $\odot$ is the Hadamard product. Accordingly, the backward propagation is also conducted by considering the binary masks. Trained in this way, the learned convolution kernels are enforced to focus on different part of the input feature maps through the binary masks. For inference, the convolution is conducted in a conventional way without mask, such that the learned kernels can search for certain input pattern throughout the whole input feature map.

The initialization manner of the binary masks can be customized for the particular problem at hand. In our method, we divide each mask into a grid of 2×2 blocks. All the values within each block are initialized by one random variable which is drawn from a Bernoulli distribution. The case where all the four blocks of the mask are set to zeros are deliberately avoided by re-initialization.

4. Tracking Algorithm

Overview. The overall tracking procedure is presented in Algorithm 1. The feature extraction network *CNN-E* consists of the first ten convolutional layers of the 16-layer VGG network [29] trained on Imagenet Classification task [5], which takes an RGB image as input and outputs a feature map $\boldsymbol{X}$ of 512 channels. The first layer of the adaptation network *CNN-A* employs the proposed convolution with mask layer with convolution kernels of 5×5 spatial size and generates a feature map of 100 channels (corresponding to 100 base learners). The second layer of *CNN-A* is a convolution layer with kernel size 3×3 and produces a feature map of one channel for target localization. To handle scale variation, a scale prediction network *SPN* is further built on top of the pre-trained *CNN-E*. The *SPN* network takes the output feature map $\boldsymbol{X}$ of *CNN-E* as input and first applies a set of predefined scale transformations $\mathcal{S} = \{s_l | l = 1, 2, \ldots, n_s\}$ to obtain the corresponding scale-transformed feature maps $\{\mathcal{T}(\boldsymbol{X}, s_l)|\}_1^{n_s}$, where s_l denotes the parameter (scale factor) for the l-th scale transformation. Then all the transformed feature maps are passed through a fully connected layer which predicts an optimal scale s^* for the current target.

Initialization. Given the ground truth target location in the first frame, we crop a rectangle image region $\boldsymbol{I}_1$ centered at the target location with twice the size of the target bounding box. The corresponding feature map $\boldsymbol{X}_1$ is extracted by *CNN-E*. As described in Section 3.2, the base learners $\{\boldsymbol{f}(\boldsymbol{X}_1; \gamma_k)\}_1^{100}$ are initialized independently to predict the target score map $\boldsymbol{M}_1$ using the Euclidean loss

$$L(\boldsymbol{M}_1, \boldsymbol{f}(\boldsymbol{X}_1; \gamma_k)) = \|\boldsymbol{M}_1 - \boldsymbol{f}(\boldsymbol{X}_1; \gamma_k)\|_2^2, \tag{11}$$

where $\boldsymbol{M}_1$ is a Gaussian distribution centered at the ground truth target location with a small scale. The optimal parameter $\gamma^* \in \{\gamma_k\}_1^{100}$ for the base learner with the smallest training error is used to initialize the ensemble set $\mathcal{E}$, whereas the rest constitute the candidate set $\mathcal{C}$. Meanwhile, the *SPN* network is trained to predict the current scale of the target using a hinge loss

$$L_S = \max\left(0, 1 + \max_{s_l \neq s^*, s_l \in \mathcal{S}} F_S(\mathcal{T}(\boldsymbol{X}, s_l)) - F_S(\mathcal{T}(\boldsymbol{X}, s^*))\right) + R_S \tag{12}$$

where s^* indicates the ground truth scale of the target; F_S is the score predicted by *SPN* and R_S denotes weight decay.

Online Tracking. In the t-th frame, a rectangle image region $\boldsymbol{I}_t$ centered at the last location is cropped from the input image and passed through the *CNN-E* network to obtain $\boldsymbol{X}_t$. The ensemble of base learners take the corresponding feature map $\boldsymbol{X}_t$ as input and predict a heat map as $\hat{\boldsymbol{M}}_t = \frac{1}{|\mathcal{E}|} \sum_{\gamma_i \in \mathcal{E}} \boldsymbol{f}(\boldsymbol{X}_t; \gamma_i)$. The center location of the target is then determined by the location on the heat map with the maximum value. The maximum heat map value then serves as the confidence $conf_t$ of this prediction. To predict the current scale, we crop another image region $\hat{\boldsymbol{I}}_t$ which is centered at the predicted target location and has

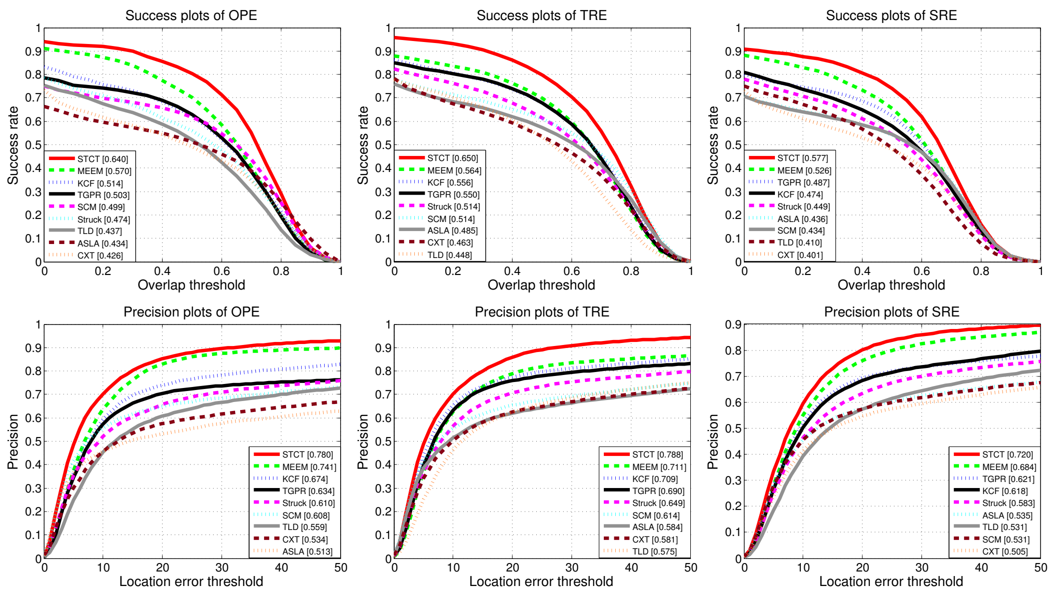

Figure 3. Average success plots and precision plots of nine leading methods in OBT data set for OPE, TRE and SRE evaluations. Trackers are ranked according to the Area Under Curve (AUC) scores.

twice the size of the target bounding box in the last frame. The current scale of the target is predicted by the *SPN* network as $\hat{s} = \arg\max_{s_l \in \mathcal{S}} F_S(\mathcal{T}(\hat{\boldsymbol{X}}_t, s_l))$, where $\hat{\boldsymbol{X}}_t$ denotes the feature map extracted by *CNN-E* from region $\hat{\boldsymbol{I}}_t$.

Online Update. To avoid updating using contaminated training samples, online update is conducted only if the confidence of the location prediction is higher than a predefined threshold θ. The base learners in the ensemble set $\mathcal{E}$ and candidate set $\mathcal{C}$ are updated respectively with the following loss function using SGD

$$L_{\mathcal{E}}(\boldsymbol{M}_t, \mathcal{E}) = \|\boldsymbol{M}_t - \frac{1}{|\mathcal{E}|}\sum_{\boldsymbol{\gamma}_i \in \mathcal{E}} \boldsymbol{f}(\hat{\boldsymbol{X}}_t; \boldsymbol{\gamma}_i)\|_2^2,$$
$$L_{\mathcal{C}}(\boldsymbol{M}_t, \boldsymbol{\gamma}_j \in \mathcal{C}) = \|\boldsymbol{M}_t - \boldsymbol{f}(\hat{\boldsymbol{X}}_t; \boldsymbol{\gamma}_j) - \eta\frac{1}{|\mathcal{E}|}\sum_{\boldsymbol{\gamma}_i \in \mathcal{E}} \boldsymbol{f}(\hat{\boldsymbol{X}}_t; \boldsymbol{\gamma}_i)\|_2^2, \tag{13}$$

where $\boldsymbol{\gamma}_i \in \mathcal{E}$ is fixed when updating $\boldsymbol{\gamma}_j \in \mathcal{C}$; $\boldsymbol{M}_t$ denotes the heat map of Gaussian distribution centered at the estimated location. If the current training error $L_{\mathcal{E}}$ of the ensemble is higher than a threshold ε and the candidate set $\mathcal{C}$ is not empty, we sample without replacement one base learner parameter from the candidate set $\mathcal{C}$ and add it to the ensemble set $\mathcal{E}$. In our experiment, we adopt the following sampling probability density

$$p(\boldsymbol{\gamma}_j) = \delta(\hat{\boldsymbol{\gamma}}^* - \boldsymbol{\gamma}_j), \quad \boldsymbol{\gamma}_j \in \mathcal{C}, \tag{14}$$

where $\delta(\cdot)$ is the Dirac delta function; $\hat{\boldsymbol{\gamma}}^* \in \mathcal{C}$ denotes the optimal parameter with the smallest training error $L_{\mathcal{C}}$. Meanwhile, the *SPN* network is also updated via (12) using sample $\hat{\boldsymbol{X}}_t$ and the predicted scale $\hat{s}$ as ground truth, where the training sample $\hat{\boldsymbol{X}}_t$ is resized into a fixed spatial size (256×256 in our experiments) to fit the input size of *SPN*.

5. Experiments

Implementation Details The proposed tracker is implemented in MATLAB with Caffe framework [16], and runs at 2.5 fps on a PC with a 3.4GHz CPU and a TITAN GPU. The source code is publicly available[1]. Both the *CNN-A* and the *SPN* networks are trained online using SGD with learning rates of $5e{-}7$ and $1e{-}10$, respectively. It takes 50 iterations to initialize in the first frame and 2 iterations for the following each updating step. We use $n_s = 11$ scale parameters for scale transformation with a coverage of $[0.82, 1.21]$ of the original scale. The threshold θ and ε for online update are set to 0.2 and 0.4, respectively. The parameter η in (13) is set to 0.2. A Bernoulli distribution $B(0.5)$ is employed to randomly generate binary masks (Section 3.3). We fix all the parameters throughout the experiments.

5.1. Evaluation on OTB Data Set

Data Set and Evaluation Settings. The OTB data set [37] includes 50 sequences tagged with 11 attributes which covers various challenging factors. We evaluate the proposed STCT tracker against 29 state-of-the-art trackers[2] and three recently proposed trackers: MEEM [40], TGPR [8] and KCF [12]. Three different experiments are conducted as in [37], including one pass evaluation (OPE), temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). Among others, TRE randomizes the starting frame of the evaluation, and SRE randomizes the initial bounding boxes by perturbation. As additional evaluations to OPE, TRE and SRE can better demonstrate the robustness of the

[1]`http://ice.dlut.edu.cn/lu/index.html`.

[2]We use the results of the 29 trackers reported in [37] for fair comparison.

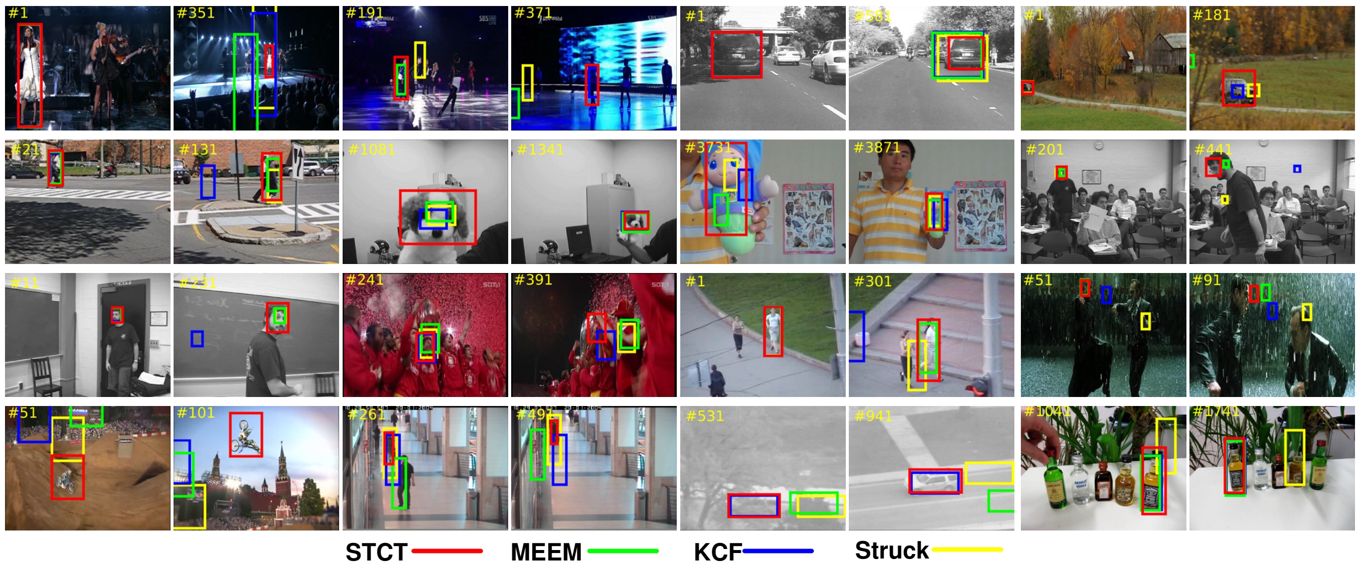

Figure 4. Qualitative results of the proposed STCT tracker on a subset of challenging sequences: *Singer1*, *Skating1*, *Car4*, *CarScale*, *Couple*, *Dog1*, *Doll*, *Freeman3*, *Freeman1*, *Soccer*, *Jogging-2*, ,*Matrix*, *MotorRolling*, *Walking2*, *Suv* and *Liquor*.

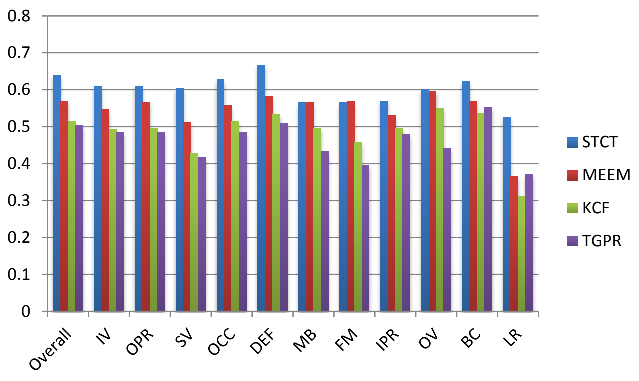

Figure 5. Average AUC scores of the success plots of the four leading trackers under different attributes of test sequences in OPE, including: illumination variation (IV), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-view (OV), background cluttered (BC) and low resolution (LR).

evaluated trackers.

We use the precision plot and the success plot to evaluate all the trackers. The precision plot demonstrates the percentage of frames where the distance between the predicted target location and the ground truth location is within a given threshold. Whereas the success plot illustrates the percentage of frames where the overlap ratio between the predicted bounding box and the ground truth bounding box is higher than a threshold $\tau \in [0, 1]$. The area under curve (AUC) is used to rank the tracking algorithms in each plot.

Evaluation Results. Figure 3 demonstrates the average precision plots and success plots on all the 50 sequences of the top nine trackers, including MEEM [40], TGPR [8], KCF [12], SCM [42], Struck [11], TLD [17], ASLA [15], CXT [6], and the proposed STCT tracker. Our method achieves the highest performance in terms of both evaluation metrics and outperforms the second best tracker (MEEM) with a considerable margin. Note that, among all three experiments, SRE is most challenging since the trackers are initialized with inaccurate target locations. Though all the evaluated trackers achieve lower performance in SRE, our method can still compare favorably against the other trackers, which demonstrates the robustness of our method. Qualitative tracking results on some challenging sequences are shown in Figure 4.

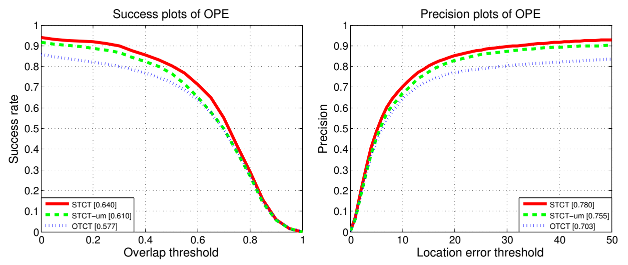

Figure 6. Average success plot and precision plot of the proposed tracker STCT against baseline methods.

To facilitate more detailed analysis, we further report the performance (success scores) of the top 3 trackers on different attributes in Figure 5. Our method can well handle various challenging factors and consistently outperform the other two trackers in almost all the attributes.

To gain more insights on the effectiveness of the proposed sequential CNN training method, we also compare the proposed STCT tracker with two baseline methods: STCT-um and OTCT, where STCT-um denotes a variant of the proposed method that does not exploit the proposed con-

Table 1. The average ranks of accuracy and robustness under baseline and region noise experiments in VOT2014. The first, second and third best methods are highlighted in **red**, **blue** and **green** colors, respectively

Trackers	baseline			region_noise			Overall Rank
	Acc. Rank	Rob. Rank	Average	Acc. Rank	Rob. Rank	Average	
STCT	**6.26**	**6.34**	**6.30**	**6.22**	**5.87**	**6.05**	**6.17**
PLT_14	7.50	**5.38**	**6.44**	7.64	**4.81**	**6.23**	**6.33**
DGT	7.02	**6.42**	**6.72**	**6.42**	**6.90**	**6.66**	**6.69**
DSST	6.86	8.28	7.57	**6.72**	8.29	7.51	7.54
SAMF	**6.58**	7.67	8.76	6.82	8.43	7.63	7.65
KCF	**6.46**	8.98	7.72	7.22	8.88	8.05	7.89
eASMS	8.34	7.98	8.16	7.86	7.83	7.85	8.00
MCT	8.64	8.36	8.50	9.00	8.42	8.71	8.61
MatFlow	10.20	7.12	8.66	9.98	9.15	9.57	9.11
VTDMG	9.42	9.52	8.47	9.22	8.70	8.96	9.21

volution with mask layer, and OTCT represents the tracking method that trains the *CNN-A* (Section 4) network using conventional training method (Training the whole network jointly by SGD). The success plot and precision plot in Figure 6 demonstrate that the proposed sequential CNN training method can significantly improve the tracking performance and that the proposed convolution with mask layer can further reduce over-training.

5.2. Evaluation on VOT2014 Data Set

Data Set and Evaluation Settings. The VOT2014 data set [18] contains 25 video sequences with various real-life visual phenomena. According to the evaluation protocol, the evaluated tracker is re-initialized whenever a tracking failure (overlap between estimated and ground truth target bounding box equals zero) is detected. Following [18], we conduct two experiments: a baseline evaluation, where trackers are initialized with ground truth target location; a noise evaluation, where the initial target location is perturbed with random noises. Two metrics are used to rank all the trackers: accuracy and robustness, which measure the overlap ratio with ground truth bounding box and the probability of tracking failure, respectively. We evaluate the proposed STCT tracker against all the trackers submitted to VOT2014 challenge [18]. Readers are referred to [18] for more details about the compared trackers.

Evaluation Results. Due to limited space, we only present the average accuracy and robustness rank of top ten compared trackers in Table 1. In both the baseline and region noise experiments, the proposed STCT tracker achieves the highest accuracy score with a relatively small number of tracking failure. Note that the DSST tracker [4] also explicitly estimates the scale of the target using correlation filters learned from HOG features. In contrast, our method proposes to transfer pre-trained deep features for online tracking, which enables more accurate and robust tracking. Figure 7 shows the accuracy-robustness plots of the top 16

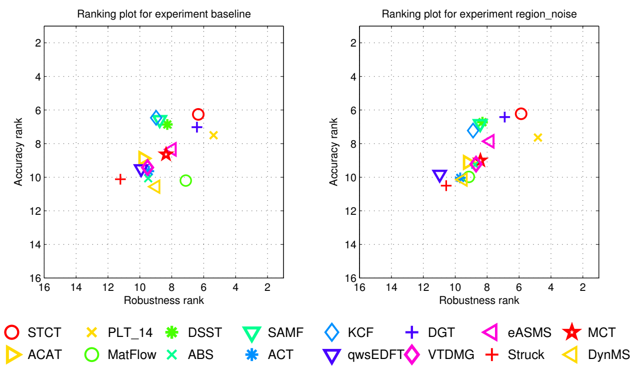

Figure 7. The robustness-accuracy ranking plots of 16 leading tracking methods under baseline and region noise experiments in VOT2014 data set. The better trackers are located at the upper-right corner.

trackers in the VOT2014 data set. Our method located at the upper-right corner achieves more favorable performance in terms of both accuracy and robustness.

6. Conclusion

In this paper, we present a sequential training method for CNNs to effectively transfer pre-trained deep features for online applications. Different from prior approaches, our method regard the online training process for CNNs as sequentially learning an optimal ensemble of base learners, such that the learned features are not highly correlated with each other. A convolution with mask layer is proposed to further reduce over-fitting. Applied to visual tracking, the proposed training method significantly improves tracking performance and compares favorably against stat-of-the-art methods in two challenging tracking benchmark data sets.

Acknowledgments. This work is supported by the Natural Science Foundation of China (61528101, 61472060), the SenseTime Group Limited, the Hong Kong Innovation and Technology Support Program (ITS/221/13FP), and the Research Grants Council of Hong Kong (CUHK 417011, CUHK 419412, CUHK 14207814, CUHK 14203015).

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8):1619–1632, 2011.

[2] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 26(1):63–84, 1998.

[3] X. Chu, W. Ouyang, W. Yang, and X. Wang. Multi-task recurrent neural network for immediacy prediction. In *ICCV*, 2015.

[4] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011.

[7] J. H. Friedman and B. E. Popescu. Importance sampled learning ensembles. *JMLR*, 94305, 2003.

[8] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*. 2014.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[10] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[11] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015.

[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, 2012.

[14] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

[15] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014.

[17] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 34(7):1409–1422, 2012.

[18] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernandez, A. Lukežič, A. Dimitriev, et al. The visual object tracking vot2014 challenge results. In *ECCV Workshops*, 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *ICCV*, 2011.

[21] H. Li, Y. Li, and F. M. Porikli. Robust online visual tracking with a single convolutional neural network. In *ACCV*, 2014.

[22] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[23] X. Mei and H. Ling. Robust visual tracking using l1 minimization. In *CVPR*, 2009.

[24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[25] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015.

[26] W. Ouyang, X. Zeng, and X. Wang. Learning mutual visibility relationship for pedestrian detection with a deep model. *IJCV*, pages 1–14, 2016.

[27] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[28] J. Shao, K. Kang, C. C. Loy, and X. Wang. Deeply learned attributes for crowded scene understanding. In *CVPR*, 2015.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[30] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015.

[31] D. Wang and H. Lu. Visual tracking via probability continuous outlier model. In *CVPR*, 2014.

[32] D. Wang, H. Lu, and M. Yang. Robust visual tracking via least soft-threshold squares. *TCSVT*, PP(99):1–1, 2015.

[33] D. Wang, H. Lu, and M.-H. Yang. Online object tracking with sparse prototypes. *TIP*, 22(1):314–325, 2013.

[34] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015.

[35] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[36] N. Wang and D. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[37] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[38] F. Yang, H. Lu, and M.-H. Yang. Robust superpixel tracking. *TIP*, 23(4):1639–1651, 2014.

[39] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *CVPR*, 2016.

[40] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *ECCV*. 2014.

[41] L. Zhang, H. Lu, D. Du, and L. Liu. Sparse hashing tracking. *TIP*, 25(2):840–849, 2016.

[42] W. Zhong, H. Lu, and M. Yang. Robust object tracking via sparse collaborative appearance model. *TIP*, 23(5):2356–2368, 2014.