Visual Tracking by Structurally Optimizing Pre-trained CNN

Chang Liu, Peng Liu, Wei Zhao, and Xianglong Tang

***Abstract*—In this paper, we propose a novel channel pruning method for convolutional neural network (CNN)-based trackers. Pre-trained CNNs are widely used in visual tracking to obtain high-level representations of targets. However, most pre-trained CNNs are trained for other tasks (e.g., VGGNet is trained for image classification), and they require a considerable amount of time to generate features. First, we introduce a dimensionality reduction method considering the information amount and tracking errors to obtain good low-dimensionality features from the last convolutional layer for tracking. Then, a backward channel selection method is proposed to select representative channels layer by layer. In this process, we aim to minimize the target changes and maximize the loss of the background or other objects. Finally, we reconstruct the neural network weights to reduce the information loss of the target with one-shot learning. Experimental results on challenging benchmarks show that the proposed channel pruning method can enhance the tracking performance and reduce the computational requirements.**

***Index Terms*—Visual tracking, channel pruning, dimensionality reduction, one-shot learning.**

I. Introduction

VISUAL tracking, which plays a significant role in computer vision, is an important research area that has attracted considerable attention in recent years [1]–[7]. Owing to improved performance, visual trackers have been employed in various applications, such as video surveillance and autonomous driving. Nevertheless, visual tracking remains challenging because the only knowledge about the target is the bounding box in the initial frame, and the tracking process involves several issues, such as scale variation, occlusion, deformation, and illumination variation.

Tracking methods include generative [8] and discriminative [9] methods. Generative trackers focus on modeling the target and searching for the most similar patch in the next frame. Discriminative trackers learn a classifier for discriminating the target from the background. Correlation-filter-based methods [10], [11] are the most well-known and effective discriminative methods owing to their high accuracy and speed. Deep-learning-based tracking methods [12], [13] have rapidly developed in recent years. Convolutional neural networks (CNN) are commonly used for feature extraction. CNN features show better discriminant ability than hand-crafted ones [14], [15].

Owing to the lack of a large amount of labeled tracking videos, pre-trained CNNs, such as AlexNet [16] and VGGNet [17], are widely used in a number of trackers. However, such networks are trained for other tasks. For example, VGG is trained to classify numerous types of objects. It is inconsistent with tracking tasks in which an instance of a single type of object is to be tracked in each video [18]. Consequently, trackers with CNN features are prone to drifting to other similar objects when the target changes [19]. Another problem is that pre-trained CNNs are generally large-scale and with massive parameters. Moreover, CNN feature extraction is a time-consuming task that requires considerable computational resources.

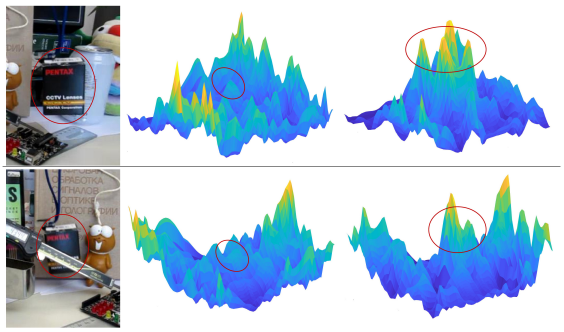

Fig. 1. Features extracted from pre-trained and channel-pruned networks. The first and second rows are from the first and 300th frames, respectively. The first column shows the input patches sampled from the original images. The second and third columns show the features obtained from layer Conv4-3 of pre-trained and channel-pruned VGG16 networks, respectively. For visualization, the summation of all feature maps in that layer is calculated by element, and then the summation map is normalized to represent the feature. The target regions are labeled with circles. After channel pruning, the network can generate better representations of the target.

Recently, deep compression methods [20] have become increasingly popular. They aim to compress deep neural networks for embedded systems with limited hardware resources, such as mobile applications. By using network pruning, network quantization, and coding, compressed networks can achieve considerable speedup and high energy efficiency without a significant loss of accuracy. Shallower layers in CNNs have greater redundancy, whereas deeper layers have important information to discriminate different object types and are more difficult to compress [21]. Accordingly, common deep compression methods focus on reducing the redundancy in shallow layers and ensuring that the information in deep layers remains unchanged. However, in visual tracking problems, only a single target should be tracked in a video. The deep layers in CNNs are redundant, and the redundancy features will cause trackers to drift. Hence, it is necessary to compress CNNs for visual tracking.

This work was supported by the National Natural Science Foundation of China under Grant 61671175, 61370162 and 61672190. *(Corresponding author: Wei Zhao.)*

The authors are with the Department of Computer Science, Harbin Institute of Technology, Heilongjiang 150001, China (e-mail: magicallc@126.com; pengliu@hit.edu.cn; zhaowei@hit.edu.cn; tangxl@hit.edu.cn).

A significant challenge in compressing CNNs for visual tracking is that only one labeled sample is given. Common deep compression methods use numerous samples to compress the network and then use all the samples to fine-tune the compressed model to ensure the representation ability. However, the use of only one sample can easily lead to over-fitting when the network is compressed. When the target changes, the compressed model may not recognize it.

The objective of this study is to optimize the structure of pre-trained CNNs for visual tracking by means of a channel pruning method [21]. We aim to achieve the best CNN performance and transform the original network into a visual tracking problem without extra fine-tuning. A lightweight CNN is obtained by compressing a pre-trained CNN. The compressed CNN can generate better features for a specific tracking target more efficiently than the pre-trained CNN. In deep layers, different channels reflect different semantics of different objects. For visual tracking, the semantic information of the target is a valuable feature, whereas that of the other objects is noise. Hence, pruning of deep layers is useful for discriminating the target from objects of other types. In shallow layers, channels contain detailed edge or texture information of all objects. Too many channels would be redundant because some channel features can be represented by other channel features. Hence, pruning of shallow layers is useful for discriminating the target from similar objects. Thus, good features can be learnt by optimizing the structure of CNNs. The reconstruction of layer outputs is the most significant task in deep compression. If the amount of information in the output is small, the network layer output can be easily pruned and reconstructed. In visual tracking, channels in deep layers contain considerable information for describing other objects besides the target. Reconstructing such information is unnecessary and detrimental to visual tracking. Therefore, in contrast to the compression strategy of common deep compression methods, which compress the networks layer by layer forward, we conduct the procedure backward.

First, we reduce the dimensionality of the last output layer for tracking. The last output layer is directly used for tracking. Even with only one sample, we can select feature maps favorable for tracking a specific target. Thus, the target is easier to track and we do not need to reconstruct the information of the unfavorable feature maps in the next compression procedure. The compression ratio can be increased and the learning problem with one sample is simplified. Moreover, after selecting good channels for tracking in the last layer, we can gradually increase the favorable information in the procedure of pruning the front layers. Then, channel selection is processed layer by layer. Our objective is not only to preserve the target information in deep layers but also to reduce the information of the background and other objects. The process is conducted backward, and the information loss for reconstruction is minimized. After obtaining the reduced network, we reconstruct the remaining weights with one sample to reduce the information loss of the target in the channel selection process and enhance the robustness in case the target changes. This procedure is run forward, and the recovery error can be compensated when reconstructing the following layers. Finally, the kernelized correlation filter (KCF) tracker is used to demonstrate the effect of the features extracted from the pruned network. Figure 1 shows the effect of channel pruning. The first column shows the image patch sampled at the position of the target. The second column shows the features extracted from layer Conv4-3 of the pre-trained VGG16 network. The third column shows the features obtained from the compressed VGG16 network. For visualization, the summation of all feature maps in that layer is calculated by element, and then the summation map is normalized to represent the feature. The first row is from the first frame, and the second row is from the 300th frame. The corresponding target regions are labeled with circles. The target feature is not obvious when extracted from pre-trained networks. However, with channel pruning, favorable channels are selected and the target is easier to track.

Our contributions can be summarized as follows:

–We propose a dimensionality reduction method that selects good feature maps in the last convolutional layer and reduces the dimensionality for efficient tracking.

–We explore a backward channel selection method that drops redundant channels, preserves the target information, and reduces the background information simultaneously, with only one sample.

–We introduce a one-sample weight reconstruction method that reduces the target information loss and preserves the network robustness after compression.

The remainder of this paper is organized as follows. Section II reviews related studies. Section III presents the problem formulation and theoretical details. Section IV presents and analyzes the experimental results of the proposed method. Finally, Section V concludes the paper.

II. Related Work

Recent years have witnessed significant advances in visual tracking. In this section, we briefly introduce the methods related to this study, including the generative and discriminative models in visual tracking. Further, we review the development of deep learning technology in visual tracking as well as deep compression methods.

A. Generative and Discriminative Trackers

Visual tracking methods can be classified into generative and discriminative methods.

Generative methods learn the appearance model of the target and search for the most similar candidate in the next frame. Incremental visual tracking (IVT) [8] learns a low-dimensional subspace representation of the target. Local orderless tracking (LOT) [22] segments the target into superpixels and explores the local orderless representation. Sparse-representation-based methods [23] contribute significantly to the generative domain, as they facilitate the learning of robust representations. In [24], sparse representation has been combined with circulant samples. Generative trackers lack utilization of background information. Their precision is reduced considerably when the target is severely deformed, occluded, or in a cluttered background.

Recently, discriminative trackers have shown better performance in terms of accuracy and speed compared to generative trackers [1], [2]. They learn a classifier to discriminate the target from the background. Multiple instance learning has been used in [9], while a structured SVM has been used in [25]. The most prominent methods are correlation-filter-based methods. The fast Fourier transform (FFT) was proposed for target tracking in [26]. Subsequently, high-speed performance was achieved by using circulant samples, training the classifier, and detecting the target with FFT [10], [27]–[29]. High-speed operation allows the use of better features and leads to high accuracy. In [30]–[32], the authors focused on solving the bound effect of correlation filters.

B. Deep Learning in Visual Tracking

Deep learning technologies have been successfully adopted in many domains. A number of significant studies have recently investigated visual tracking. Wang [12] used a stacked denoising autoencoder for online tracking. Hong et al. [33] obtained features using a CNN and tracked the target with an SVM; however, their approach is inefficient. Owing to the high speed of correlation-filter-based trackers, pre-trained CNNs are widely used to extract features [14], [15], [34]–[36]. GOTURN [37] constructs deep regression networks and can track 100 frames per second (FPS). However, its precision is low. Fully convolutional Siamese networks [38] track the target with Siamese networks that can generate consistent features for different sizes of image patches. A convolutional layer is used to realize the correlation operation. The speed achieved by this method in GPU is comparable with that of correlation filters. An end-to-end representation can be learned instead of using pre-trained CNNs [18]. YCNN [39] connects two-flow inputs with full connected layers and gets a network with 'Y' shape. However, re-training CNNs requires a large amount of labeled videos and offline computational resources. CREST [40] uses spatial and temporal residual layers to learn better representations. EAST [41] learns policies for tracking with shallow CNN layers in advance. Gao et al. [42] proposed a novel relative tracker that exploits the relative relationship among image patches. Chi et al. [43] learned dual networks on the basis of different layers of pre-trained CNNs. Many other state-of-the-art deep learning technologies are employed in visual tracking, such as recurrent neural networks (RNN) [44], reinforcement learning [45], [46], attention [47], adversarial learning [48], and region proposal network (RPN) [49].

C. Deep Compression Methods

Deep compression aims to reduce the storage and energy requirements of deep neural networks. This issue has become increasingly important with the rapid development of mobile devices. Networks can be compressed from three perspectives: network pruning, network quantization, and coding [20]. Quantization and coding are mainly used to reduce the network storage. Network pruning focuses on simplifying the network structure. It mainly involves sparse connection [50], tensor factorization [51], and channel pruning [21]. Sparse connection methods deactivate connections between neurons or channels, and they can achieve a high speed-up ratio. However, it is difficult to implement sparse convolutional layers, which require a large number of samples. Tensor factorization methods factorize a convolutional layer into several layers, especially the efficient 1×1 convolutional layer. The network parameters can be reduced, but extra computation overhead is incurred. Channel pruning methods directly remove channels and the corresponding network connections. Channel pruning is easier to implement and achieves greater weight reduction than the other two methods. However, the network output might be damaged. Li et al. [52] pruned filters with smallest sum values. To recover the original accuracy, the model needs to be retrained every time a layer is pruned. Luo et al. [53] pruned filters and channels on the basis of static information computed in the next layer. He et al. [21] used LASSO regression to select channels that minimize the output change of the current layer. This method can achieve similar accuracy compared to the original model without fine-tuning. However, 50000 samples are required to prune the channels without damaging the model output. XNOR-Net [54] proposed to represent each weight parameter and activation with 1-bit. The efficiency was improved greatly, whereas the accuracy degradation was severe. Liu et al. [55] proposed the Bi-Real Net, optimized the training algorithm, and improved the performance of XNOR-Net. Learnet [19] uses a deep compression method in visual tracking. The authors employed tensor factorization to reduce the size of the output space and a one-shot meta-learning method to learn the reduced parameters of a layer. However, they also used a sampling method to generate large numbers of image patch pairs to update the parameters on the basis of gradient descent. Further, they learned additional filter weights for the pupil network to recognize objects of the same type. Thus, they showed that deep compression increases the feasibility of one-shot learning and it can be effective for a specific task. However, the improvement in tracking was insignificant. Therefore, we investigate how to improve the tracking performance and aim to highlight the target and reduce the background information using deep compression technology.

III. Proposed Approach

In this section, we first introduce the kernelized correlation filter (KCF) tracker. The KCF is used to construct a tracking loss to help select channels, and it is implemented for the final tracking. Then, we present the proposed dimensionality reduction method for the last layer, the backward channel selection method, the one-sample weight reconstruction method, and the entire tracking method with channel pruning. Figure 2 shows the framework of the entire channel pruning process.

A. Kernelized Correlation Filters Tracking

Henriques et al. [10] considered object tracking as a kernelized ridge regression problem. With circular samples $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^{\mathrm{T}}$ and corresponding labels $\mathbf{y} = [y_1, \cdots, y_n]$, a ridge classifier $y = f(\mathbf{x})$ is learned. $\mathbf{x}_i, i = 1, .., n$ are all generated circularly from a base $\mathbf{x}$. $y_i, i = 1, ..., n$ are determined by a Gaussian distribution. Here, we briefly introduce

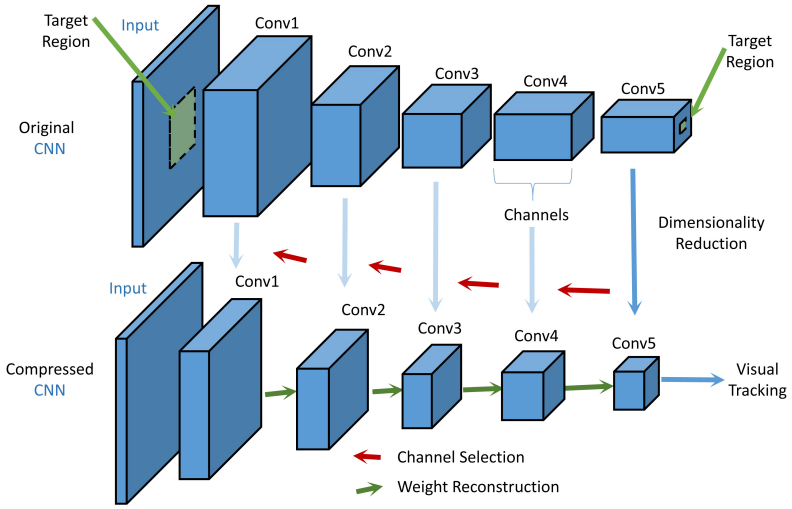

Fig. 2. Framework of the proposed channel pruning method for visual tracking. In the first frame, with the only sample, channel pruning is conducted with dimensionality reduction, channel selection, and weight reconstruction in order. In the next frames, the pruned network generates features for tracking.

the training and detection methods. Detailed information can be found in [10].

With the kernel strategy and the special characteristic of the circular matrix, the regression can be trained efficiently in the Fourier domain:

$$\mathcal{F}\boldsymbol{\alpha} = \frac{\mathcal{F}\mathbf{y}}{\mathcal{F}(\mathbf{k}^{\mathbf{xx}}) + \lambda} \tag{1}$$

where $\mathcal{F}$ denotes the Fourier transformation, $\boldsymbol{\alpha}$ is the kernel parameter of f, $\mathbf{k}^{\mathbf{xx}}$ is the self-kernel correlation of $\mathbf{x}$, and λ is a regularization parameter. The target can be detected in a new sample $\mathbf{z}$:

$$\mathbf{y} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{k}^{\mathbf{xz}}) \cdot \mathcal{F}\boldsymbol{\alpha}) \tag{2}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transformation, and $\mathbf{k}^{\mathbf{xz}}$ is the kernel correlation of $\mathbf{x}$ and $\mathbf{z}$.

In the case of multiple channels, the linear kernel correlation can be calculated as

$$\mathbf{k}^{\mathbf{xz}} = \mathcal{F}^{-1}(\sum_{c}(\mathcal{F}\mathbf{x_c})^* \odot (\mathcal{F}\mathbf{z_c}) \tag{3}$$

where c denotes the c_{th} channel and $*$ is the complex-conjugate transformation.

B. Dimensionality Reduction for Tracking

Shallow layers in CNN commonly describe the texture or edge features of the image, whereas deep layers contain rich semantic features. Deep layers perform better than shallow layers in visual tracking, because the feature maps in deep layers are invariant when the target changes. However, deep layers have more channels than shallow layers. For example, in the commonly used VGG16 network, there are 256 channels in layer Conv4-3 and 512 channels in layer Conv5-3. Hence, the outputs of the layers are 256 and 512 feature maps, respectively. Tracking methods need to calculate the similarity of each feature map. The time consumed is nearly linearly proportional to the number of feature maps. Thus, it is inefficient to track the target with these layers. A common solution is dimensionality reduction, for which principal component analysis (PCA) is the most widely used method. PCA can reduce the duplicate information between feature maps and improve the efficiency of tracking, but it cannot reduce the effect of other objects on the target. The last layer is closely related with tracking, and selecting information related with the tracked target in the last layer facilities the selection of favorable information for tracking in the front layers. Only good target representations should be reconstructed. Hence, we aim to select favorable feature maps and reduce the dimensionality simultaneously. Furthermore, selecting favorable feature maps of the last layer is useful for channel selection and weight reconstruction of the previous layers.

For visual tracking, most feature maps have a small amount of information because the deep layers represent the semantic features of objects but few objects appear. These feature maps should be dropped, as they are likely to be noise. Matrix norms can be used to express the information amounts of feature maps, such as ℓ_0, ℓ_1, and ℓ_2 [52]. In the case of a large number of experiments, we choose the ℓ_2 norm to express information amounts, because it is the smoothest and most robust candidate:

$$In(\mathbf{x}_c) = ||\mathbf{x}_c||_2^2 \tag{4}$$

where $\mathbf{x}_c$ denotes the feature map of channel c.

Some feature maps with a large amount of information are unfavorable for tracking. There are two typical cases. First, the background shows larger feature values, whereas the feature values of the target are very small. Second, the background and the target have similar feature values. Hence, we evaluate each feature map independently and construct the following tracking error to represent the discriminant ability with the KCF:

$$E(\mathbf{x}_c) = ||\mathcal{F}^{-1}(\frac{\mathcal{F}\mathbf{y}\mathcal{F}(\mathbf{k}^{\mathbf{x_c^o x_c^b}})}{\mathcal{F}(\mathbf{k}^{\mathbf{x_c^o x_c^o}}) + \lambda})||_2^2 \tag{5}$$

where $\mathbf{x_c^o}$ and $\mathbf{x_c^b}$ are the target features and background features in $\mathbf{x}_c$, respectively.

Equation (5) represents training of a ridge regression with the target feature and detection of the target in the background. The smaller information amounts of the detection response lead to better ability of discriminating the target from the background. This metric is sensitive to noise. Hence, we gather the information amounts and tracking error together with PCA. Let α denote the channel selection parameter, where $\alpha_c \in \{0, 1\}$ and $\alpha_c = 0$ implied that channel c is dropped. It can be concluded that the proposed dimensionality reduction minimizes the following loss:

$$L(\alpha) = L_{PCA}(\mathbf{x}\ diag(\alpha)) + \sum_c (1-\alpha_c) In(\mathbf{x}_c) + \alpha_c E(\mathbf{x}_c) \quad (6)$$

where $L_{PCA}(\mathbf{x}) = ||\mathbf{x} - \mathbf{x}DD^T||_2^2$ is the PCA loss and D is the PCA projection matrix. It is noted that only feature maps with $\alpha_c = 1$ need to be transformed with PCA, because the values in the c_{th}column in $\mathbf{x}\ diag(\alpha)$ are zeros if $\alpha_c = 0$.

For the task of reducing M feature maps to d feature maps, we solve Equation (6) in two steps. First, d' best feature maps are selected considering the information amounts and tracking error. Then, the selected d' feature maps are further reduced to d feature maps with PCA. d' is determined as the value with the least loss among several reasonable candidates.

C. Channel Selection for Tracking

Convolutional neural networks have two special characteristics. One characteristic is location invariance. The target is located at the same positions of feature maps from different convolution layers, for which pre-trained CNNs can be directly used for tracking. The other characteristic is that deeper layers have wider receptive fields than shallower layers. The features in deep layers are the local information aggregation of features in shallow layers. Accordingly, we propose that selecting favorable information for tracking in shallow layers enables us to obtain that in deep layers. Channel selection can select favorable channels and improve the efficiency of feature extraction. It will not affect the target locating process if the target information is preserved.

We first propose the channel selection method for a single layer and then generalize it to cases of multiple layers.

In deep compression, the objective is to maintain the output after channel selection. For visual tracking, we aim to minimize the information loss of the target and the output of the background.

Considering a single convolutional layer, the $c \times W \times H$ input tensor $\mathbf{X}$ is applied with $n \times c \times k_h \times k_w$ convolutional filters $\mathbf{W}$ to generate an $n \times W \times H$ output tensor $\mathbf{Y}$, where W and H are the sizes of feature maps, c and n are the numbers of input channels and output channels, and k_h and k_w are the sizes of the convolutional filters, respectively. To realize convolution in the form of matrix multiplication, we consider extracting a sample at every pixel from each input feature map, to get $c \times W \times H$ individual samples of size $k_h \times k_w$. Then, these samples and filters are reshaped as column vectors. Let $N = W \times H$ and $k = k_h \times k_w$; the convolution operation can be represented as

$$\mathbf{Y} = \sum_{i=1}^{c} {\mathbf{W}_i}^T \mathbf{X}_i + \mathbf{b}\mathbf{1}^T \quad (7)$$

where $\mathbf{Y}$ is an $n \times N$ matrix representing the n output feature maps, $\mathbf{X}_i$ is a $k \times N$ matrix denoting the samples from the i_{th} input feature map, and $\mathbf{W}_i$ is a $k \times n$ matrix expressing n convolutional filters corresponding to the i_{th} input channel. Further, $\mathbf{b}$ is an $n \times 1$ column vector representing the bias for each output channel and $\mathbf{1}$ is an $N \times 1$ column vector whose values are all 1.

Then, the features in the target and background can be separated easily owing to the CNN characteristic of weight sharing:

$$\mathbf{Y}^p = \sum_{i=1}^{c} {\mathbf{W}_i}^T \mathbf{X}_i^p + \mathbf{b}\mathbf{1}_p^T, p \in \{O, B\} \quad (8)$$

where $\mathbf{X}^p$ and $\mathbf{Y}^p$ with $p = O$ denote the input and output values in the target region of the original feature maps $\mathbf{X}$ and $\mathbf{Y}$, respectively, while $p = B$ denotes the background.

Let $\beta = [\beta_1, \beta_2, ..., \beta_c]^T$ be the channel selection parameter; the channels with $\beta_i = 0$ will be pruned and those with $\beta_i \neq 0$ will be retained. To prune the channels from c to c', the channel pruning method for visual tracking can be formulated as minimization of the following loss function:

$$\begin{aligned} L_s(\beta, c' | \mathbf{X}, \mathbf{Y}, \mathbf{W}) = & \left|\left| \mathbf{Y}^O - \sum_{i=1}^{c} \beta_i {\mathbf{W}_i}^T \mathbf{X}_i^O - \mathbf{b}\mathbf{1}_O^T \right|\right|_F^2 \\ & + \lambda_B \left|\left| \sum_{i=1}^{c} \beta_i {\mathbf{W}_i}^T \mathbf{X}_i^B + \mathbf{b}\mathbf{1}_B^T \right|\right|_F^2 \\ & s.t.\ ||\beta||_0 \leq c' \end{aligned} \quad (9)$$

where $||\cdot||_F$ denotes the Frobenius norm, and λ_B is a parameter for achieving a tradeoff between the information loss of the target and the output of the background.

For simplicity, we reshape $\mathbf{Y}^O - \mathbf{b}\mathbf{1}_O^T$ as a row vector $\mathbf{A}$, ${\mathbf{W}_i}^T \mathbf{X}_i^O$ as a row vector $\mathbf{C}_i$, and ${\mathbf{W}_i}^T \mathbf{X}_i^B$ as a row vector $\mathbf{D}_i$. Further, for $i = 1, ..., c$, $\mathbf{C}_i$ and $\mathbf{D}_i$ can be merged as matrixes $\mathbf{C}$ and $\mathbf{D}$, respectively. $\mathbf{b}\mathbf{1}_B^T$ is reshaped as a row vector $\mathbf{B}$. Then, Equation (9) can be simplified as

$$\begin{aligned} & \min_{\beta} \left|\left| \mathbf{A} - \beta^T \mathbf{C} \right|\right|_F^2 + \lambda_B \left|\left| \beta^T \mathbf{D} + \mathbf{B} \right|\right|_F^2 \\ & s.t.\ ||\beta||_0 \leq c' \end{aligned} \quad (10)$$

Equation (10) is classified as an ℓ_0 minimization problem that is NP-hard. Hence, we solve the approximate ℓ_1 minimization problem instead.

$$\min_{\beta} \frac{1}{2} \left|\left| \mathbf{A} - \beta^T \mathbf{C} \right|\right|_F^2 + \frac{\lambda_B}{2} \left|\left| \beta^T \mathbf{D} + \mathbf{B} \right|\right|_F^2 + \lambda_\beta \left|\left| \beta \right|\right|_1 \quad (11)$$

Then, we introduce a dual variable θ, replace β with θ in the ℓ_1 item, and constrain $\theta = \beta$. Equation (11) can be solved with the augmented Lagrangian [56],

$$\begin{aligned} L(\beta, \theta) = & \frac{1}{2} \left|\left| \mathbf{A} - \beta^T \mathbf{C} \right|\right|_F^2 + \frac{\lambda_B}{2} \left|\left| \beta^T \mathbf{D} + \mathbf{B} \right|\right|_F^2 + \\ & \lambda_\beta \left|\left| \theta \right|\right|_1 + (\beta - \theta)^T S + \frac{\mu}{2} \left|\left| \beta - \theta \right|\right|_2^2 \end{aligned} \quad (12)$$

where S is the Lagrange multiplier and μ is the Lagrange parameter. The augmented Lagrangian can be iteratively optimized in three steps.

Step 1: Update β:

$$\begin{aligned}\beta = \underset{\beta}{argmin}\frac{1}{2}\left|\left|\mathbf{A}-\beta^T\mathbf{C}\right|\right|_F^2+\frac{\lambda_B}{2}\left|\left|\beta^T\mathbf{D}+\mathbf{B}\right|\right|_F^2+\\ \beta^T S+\frac{\mu}{2}\left|\left|\beta-\theta\right|\right|_2^2\end{aligned} \tag{13}$$

A closed-form solution is obtained by letting the derivatives be zero:

$$\beta=(\mathbf{C}\mathbf{C}^T+\lambda_B\mathbf{D}\mathbf{D}^T+\mu)^{-1}(\mathbf{C}\mathbf{A}^T-\lambda_B\mathbf{D}\mathbf{B}^T-S+\mu\theta) \tag{14}$$

Step 2: Update θ:

$$\begin{aligned}\theta &= \underset{\theta}{argmin}\,\lambda_\beta\left|\left|\theta\right|\right|_1-\theta^T S+\frac{\mu}{2}\left|\left|\beta-\theta\right|\right|_2^2\\ &= \underset{\theta}{argmin}\,\frac{\lambda_\beta}{\mu}\left|\left|\theta\right|\right|_1+\frac{1}{2}\left|\left|\theta-\beta-\frac{1}{\mu}S\right|\right|_2^2\end{aligned} \tag{15}$$

There is an analytic solution

$$\theta=\delta(\frac{\lambda_\beta}{\mu},\beta+\frac{1}{\mu}S) \tag{16}$$

where $\delta(\epsilon,x)=sign(x)max(0,|x|-\epsilon)$ and δ denotes the shrinkage operator. As λ_β increases, $||\theta||_0$ decreases. Therefore, we can tune λ_β with a fixed number of selected channels c' in each iteration, or fix λ_β to achieve an adaptive c'.

Step 3: Update Multiplier S: The Lagrange multipliers are updated as

$$S^{i+1}=S^i+\mu^i(\beta-\theta) \tag{17}$$

where i is the number of iterations and $\mu^{i+1}=\min(\mu_{max},\rho\mu^i)$. μ_{max} is the predefined maximum value of μ, and $\rho>1$.

We define the iteration process to converge when the difference of the loss in Equation (12) between two successive iterations is extremely small. As the iteration proceeds, $|\beta|_0$ will decrease. With the obtained β after convergence, the channels for which $\beta_c\neq 0$ are selected, and the others are pruned. As the learned β does not consist of ones and zeros, the parameter of the remaining filters should be tuned. This will be discussed in next subsection.

D. One-sample Weight Reconstruction

As mentioned above, we retain the weights of the remaining connections after pruning, but the learned β does not consist of ones and zeros. It is necessary to learn new weights to preserve the output of each layer. Let c_l' denote the remaining number of channels after pruning for convolutional layer l, which means that there are c_l' values in β^l that are not zero. $\mathbf{W}$ consists of $c_l'\times c_{l+1}'$ filters, with k parameters each. Each input sample can supply N equations for $\mathbf{W}$.

A simple method is to apply the weights of β to $\mathbf{W}$. The filters in each channel get the same weight. However, this method has a large reconstruction error, and the output error will be accumulated in deep layers in the network. Therefore, we propose the one-sample weight reconstruction method.

We believe that each pre-trained filter reflects a useful feature and the structure of each filter is significant. Hence, we learn a weight for each filter. Thus, the filters are weighted with their structures preserved. $c_l'\times c_{l+1}'$ parameters γ are learned to reconstruct the output with the remaining channels. The weight reconstruction is processed forward layer by layer. For each layer, the problem loss function can be formulated as

$$L_r(\gamma|\mathbf{X},\mathbf{Y},\mathbf{W})=\sum_{j=1}^{c_{l+1}'}\left|\left|{\mathbf{Y'}}_j^O-\sum_{i=1}^{c_l'}\gamma_{ij}{\mathbf{W}_{ij}}^T\mathbf{X}_i^O\right|\right|_2^2 \tag{18}$$

where ${\mathbf{Y}'}_j$ is from the feature maps of the original model in order to reduce the accumulated error in the entire pruning process.

Then, the problem can be rearranged as c_{l+1}' independent least-squares problems:

$$\min_{\gamma_j}\left|\left|{\mathbf{Y'}}_j^O-\gamma_j^T\mathbf{Y}_j^O\right|\right|_2^2 \tag{19}$$

where each row of $\mathbf{Y}_j^O$ is $\mathbf{Y}_{ij}^O={\mathbf{W}_{ij}}^T\mathbf{X}_i^O$ for each i, and it is from the feature maps of pruned model.

To avoid over-fitting the single sample, an ℓ_2 regularization included in Equation (19). γ can be obtained by

$$\gamma_j=(\mathbf{Y}_j^O(\mathbf{Y}_j^O)^T+\lambda_W)^{-1}(\mathbf{Y}_j^O({\mathbf{Y'}}_j^O)^T) \tag{20}$$

where λ_W is the regularization parameter.

E. Whole Model Pruning and Tracking

To prune multiple convolutional layers, we apply the above-mentioned single layer channel selection method layer by layer. The process is conducted backward, because only the channels remaining after pruning in the next layer should be considered when we prune the channels in the current layer. Considering a CNN with n convolutional layers, after the dimensionality reduction method reduces the dimensionality of layer n from c_n to c_n', the channel selection process can be formulated as follows:

$$\begin{cases}\min_{\beta^l} L_s(\beta^l,c_l'|\mathbf{F}^l,\mathbf{F}_s^{l+1},\mathbf{W}_s^{l+1})\\ \mathbf{F}_s^l=Prune_F(\mathbf{F}^l,\beta^l) & l=n-1,...,1\\ \mathbf{W}_s^l=Prune_W(\mathbf{W}^l,\beta^l)\end{cases} \tag{21}$$

where L_s is the loss function in Equation (9), c_l' is the expected remaining number of channels of layer l, and β^l is the corresponding optimization parameter. $\mathbf{F}^l$ and $\mathbf{W}^l$ are the original feature maps and filter weights of layer l, respectively. $Prune_F$ and $Prune_W$ are the pruning operations that prune the channels with $\beta_i^l=0$ for the feature maps and filter weights, respectively. $\mathbf{F}_s^l$ and $\mathbf{W}_s^l$ are the remaining feature maps and filter weights after pruning layer l, respectively. For pruning each layer, we minimize the target information loss and the background information in the feature maps of the next layer. The procedure is conducted backward; thus, favorable information for tracking the target can be selected gradually. If we prune the channels forward, more unfavorable information would disturb the tracking process.

After pruning all the layers, we reconstruct the filter weights for the compressed network. The process can be formulated as follows:

$$\begin{cases}\min_{\gamma^l} L_r(\gamma^l|\mathbf{F}_N^{l-1},\mathbf{F}_s^l,\mathbf{W}_s^l)\\ \mathbf{W}_N^l=Weight(\mathbf{W}_s^l,\gamma^l) & l=1,...,n-1\\ \mathbf{F}_N^l=Conv(\mathbf{F}_N^{l-1},\mathbf{W}_N^l)\end{cases} \tag{22}$$

where L_r is the loss function in Equation (18), $\mathbf{F}_N^l$ denotes the feature maps generated by layer l of the compressed network, and $\mathbf{F}_N^0 = \mathbf{F}^0$ is the input image. $\mathbf{W}_N^l$ are the reconstructed filter weights. $Weight$ is the weighting operation whereby each small filter in $\mathbf{W}_s^l$ is weighted by the corresponding γ^l. $Conv$ is the convolution operation. This procedure is conducted forward. It is noted that $\mathbf{F}_s^l$ is from the original network; hence, the forward weight reconstruction can compensate the error in shallow layers with information in deep layers, and the accumulated error can be accounted for.

In visual tracking, for each video image sequence, the initial target position and scale are labeled. A sample $\mathbf{X}_0$ that includes the target and its background can be obtained. We prune channels of a pre-trained CNN model M with the sample. The channel pruning is conducted in the first frame for each video. The pruned model is fixed in the next tracking process. The target is determined in the first frame, and we can prune the CNN model to be suitable for this target.

The feature maps in each layer of M can be calculated with the sample $\mathbf{X}_0$. Then, we carry out dimensionality reduction, channel selection, and weight reconstruction in order. After obtaining the pruned CNN model, we can extract new good features to train the KCF model. When a new frame arrives, the features are extracted with the pruned CNN model and the position of the target can be predicted by the KCF model. The scale of the target is estimated with the method proposed in [27], and the KCF model is updated. The tracking algorithm with channel pruning is summarized in Algorithm 1.

Algorithm 1 CPT: Channel Pruning for Tracking algorithm

Require:
The original CNN model $\mathbf{M}$ with n convolutional layers, the first input sample $\mathbf{X}_0$, and a new arrived sample $\mathbf{X}_t$

Ensure:
The pruned CNN model $\mathbf{M}'$, and target prediction x_t^{target};

if $t = 0$ **then**
  Execute $\mathbf{M}$ with input sample $\mathbf{X}_0$, and obtain the feature maps in each layer;
  Set current layer $l = n$;
  Learn α and PCA parameters with Equation (6);
  repeat
    Select channels in layer l by iteratively solving β, θ, S with Equation (14), (16), (17); $l = l - 1$.
  until $l = 0$
  repeat
    Learn γ to reconstruct filter weights in layer l with Equation (20); $l = l + 1$.
  until $l = n$
  Preserve the pruned model $\mathbf{M}'$, and generate new features to train a KCF model.
end if
Extract features $\mathbf{z}$ with $\mathbf{M}'$ for $\mathbf{X}_t$;
Generate target response $\mathbf{y}$ with KCF in Equation (2);
x_t^{target} is located where $\mathbf{y}$ is the maximum.
Predict the scale of x_t^{target}, and update the KCF model.
return $\mathbf{M}'$, x_t^{target}.

IV. Experimental Results and Analysis

To evaluate the proposed tracking with channel pruning method, we first detail the implementation. Then, we analyze the effectiveness and sensitivity of different components. Finally, the proposed method is compared with state-of-the-art methods and the results are analyzed. Our algorithm is implemented in MATLAB and runs in both CPU and GPU environments. The CPU environment is Intel Core i7 (3.6 GHz) with 16 GB RAM. The GPU environment is GTX 1080ti.

A. Experimental Setup

CNN Network: We employ the most commonly used network, VGG16, as an example. VGG16 has 13 convolutional layers. The 7th convolutional layer (Conv3-3), 10th convolutional layer (Conv4-3), and 13th convolutional layer (Conv5-3) are commonly and directly used for tracking. Here, we use Conv4-3 as the last layer, which outputs feature maps for tracking. Conv4-3 has 512 channels, and the output feature maps are reduced to 64 channels for tracking with PCA if the pruned networks output more than 64 channels.

Tracking Implementation: To evaluate the effectiveness of different components of our method, we use the simple the kernelized correlation filter (KCF) tracker with a linear kernel without scale adaptation. The sample region size is 9 times the target area. The regularization parameter $\lambda = 0.0001$, and the model update rate is 0.1. For state-of-the-art comparison, we run SAMF [27], which is an improved version of KCF with seven scales. Here, we define the scales as $1.01^s, s \in \{-3, -2, -1, 0, 1, 2, 3\}$.

Datasets and Evaluation Metrics: OTB [1] is one of the most popular visual tracking benchmarks; it includes 100 videos with numerous comparisons of various state-of-the-art trackers. There are two metrics for accuracy: distance precision (DP) and overlap precision (OP). To evaluate DP, the distance between the predicted target position and the true position is calculated, and a frame is judged to be successfully tracked if the distance is less than a threshold of 20 pixels. DP is the success rate of all frames from all videos. To evaluate OP, the overlap rate between the predicted target region and the true target region is calculated. A success rate curve can be plotted with different thresholds. The area under the curve (AUC) is used to represent OP. To evaluate the efficiency, frames per second (FPS) is used to evaluate the tracker and floating-point operations per second (FLOPs) is used to evaluate the CNN network. The number of parameters of the networks is also concerned.

B. Component Evaluation

Our channel pruning method consists of dimensionality reduction, channel selection, and weight reconstruction.

Dimensionality reduction is conducted in the last layer of CNN networks. Here, we reduce Conv4-3 of VGG16 and keep the previous layers unchanged. Our dimensionality reduction method is compared with the original PCA method, and the tracking results are shown in Figure 3. DP-PCA

and OP-PCA denote the tracking performance with PCA dimensionality reduction in terms of DP and OP, respectively. Without any dimensionality reduction, the original 512 feature maps produce a DP of 0.781 and an OP of 0.539. The best dimensionality with PCA is 128, where DP is 0.789 and OP is 0.548. DP-CP and OP-CP represent the performance of the proposed dimensionality reduction method. Our method achieves a maximum DP of 0.793 with a dimensionality of 128 and a maximum OP of 0.550 with a dimensionality 64. Our method can reduce the last layer to fewer dimensionalities with better precision compared to PCA.

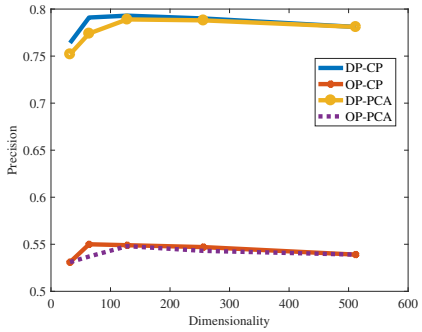

Fig. 3. Performance on OTB100 of KCF with different dimensionality reduction methods.

The proposed dimensionality reduction method involves information amounts, tracking error, and different values of the selected dimensionality d'. The tracking results of different configurations are listed in Table I. In addition to d', the effects of different information amount metrics, such as ℓ_0, ℓ_1, and ℓ_2, are shown. Our method has the best performance when $d' = 64$. Thus, the proposed selection method based on information amounts and tracking error is effective. ℓ_2 is the best representation for the information amounts.

TABLE I
TRACKING RESULTS OF DIFFERENT DIMENSIONALITY REDUCTION CONFIGURATIONS IN OTB100.

Configuration	d'	OP	DP
ℓ_2	512	0.537	0.774
ℓ_2	256	0.547	0.785
ℓ_2	128	0.547	0.787
ℓ_2	64	**0.55**	**0.791**
ℓ_2	32	0.531	0.764
ℓ_1	64	0.546	0.788
ℓ_0	64	0.543	0.784
without tracking error	64	0.540	0.779

We then evaluate the single layer pruning method with the fixed dimensionality reduction method. As discussed in the previous section, the pruning ratio can be adaptive with fixed λ_β. The comparisons with different λ_β for different convolutional layers are shown in Figure 4. It can be seen that the pruning ratio increases with λ_β. Deep layers are easier to compress than shallow layers. This is consistent with the fact that most feature maps of deep layers have little information about the target. It is noted that a fixed λ_β contributes to different pruning ratios for different layers and different targets. Fixed λ_β is adaptive to different layers, but it is not robust for different videos. Hence, we employ a fixed pruning ratio. A small λ_β is pre-set and gradually increased until the converged selection results reach the fixed pruning ratio.

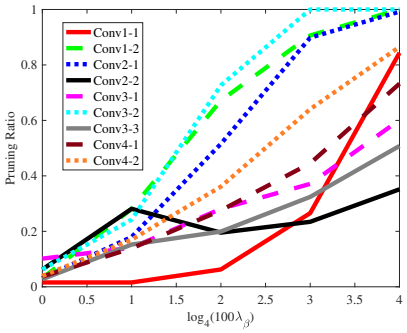

Fig. 4. Pruning ratio with different λ_β for different convolutional layers.

Pruning different layers will have different effects on the tracking results. Compression is effective in classification problems because large amounts of samples are trained so that the classifier is robust against different object states. However, tracking is sensitive to minor changes. Once the tracker fails in a frame, it would probably fail in the following tracking process. Figure 5 shows tracking results with different pruning rates for different single convolutional layers in 10 representative videos. In the 10 videos, the targets change quickly and with few scale variations. The robustness of features is significant in these videos. To prove the effectiveness of the proposed pruning method, two simple selection strategies are compared. One is the order first method named $firstK$, which selects the first k channels and prunes the left channels. The other is the max response method named $maxK$, which selects channels that have a high absolute sum. CPr denotes the proposed pruning method with weight reconstruction, whereas CP represents the version without weight reconstruction.

From Figure 5, channel pruning will not degrade the performance severely and it even has a positive effect in some cases. There is considerable scope for compression in pre-trained CNN networks for visual tracking problems. In general, the $firstK$ method shows the worst performance. It is significant to select favorable channels. Our methods have stable performance in different layers with different pruning ratios. The $maxK$ method and the $firstK$ method are unstable and show worse performance than our methods in most cases.

In shallow layers, pruning channels will degrade the performance because shallow layers have fewer channels than deep layers and the error may be magnified in the forward-propagation process. The CP method sometimes performs worse than the $maxK$ method. Because we prune the information that can be reconstructed, such information may be lost without weight reconstruction. It can be seen that our method with weight reconstruction (CPr) always performs the best in shallow layers. Information loss will increase as the network becomes deep without weight reconstruction. Hence, weight reconstruction is necessary in shallow layers. Pruning channels

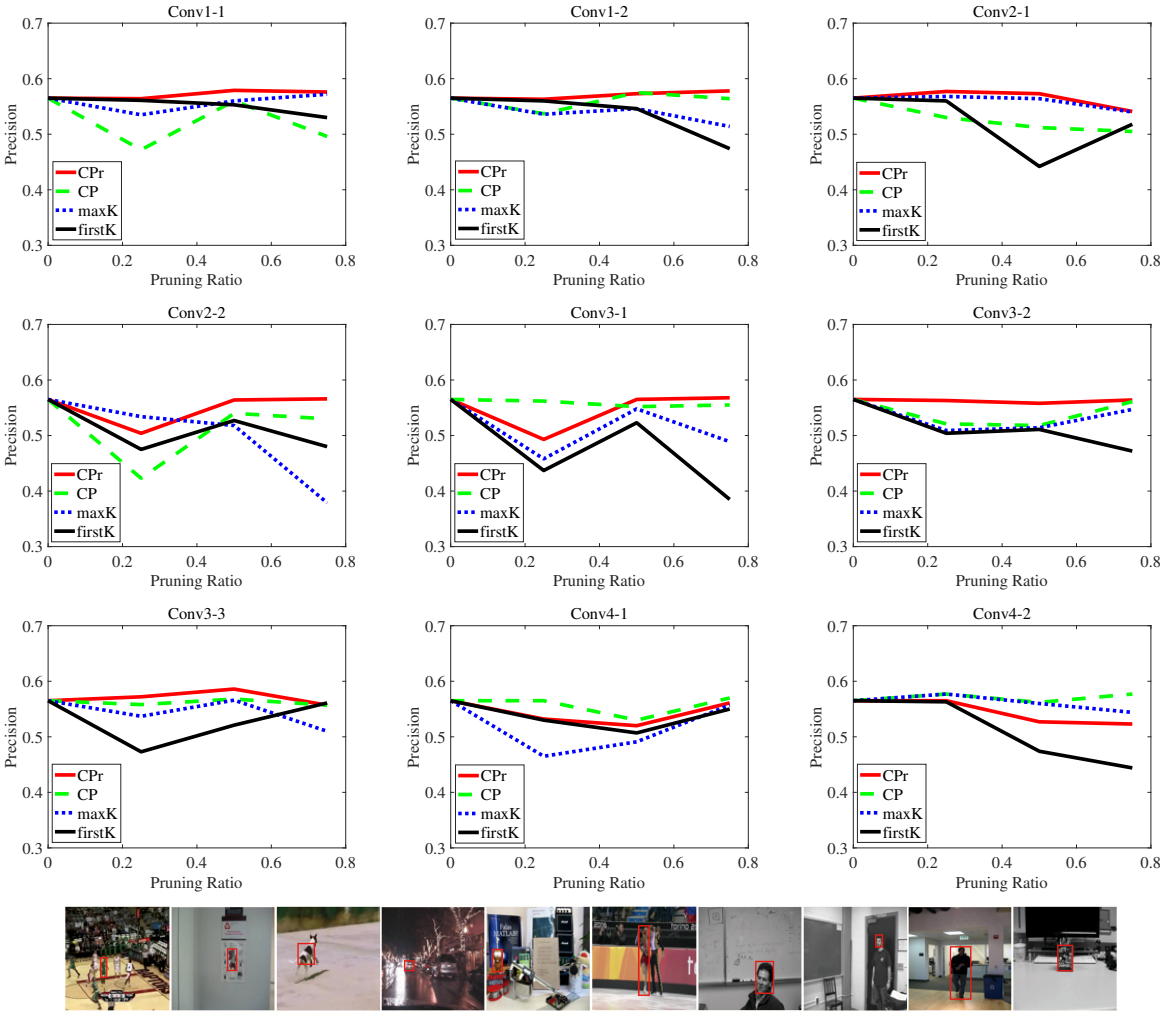

Fig. 5. Tracking results using OP plots with different pruning rates for different single convolutional layers in 10 representative videos. The first frames of the 10 videos are shown at the bottom.

in shallow layers depends on the weight reconstruction. It is not easy to ensure that the pruned information is reconstructed well and is suitable for any case with only one sample. It is suggested that a few channels in shallow layers be pruned.

In deep layers, our methods have some advantages. Our method without weight reconstruction (CP) performs better than the one with weight reconstruction (CPr). There are many channels in deep layers, most of which have little information. Hence, more noise can be pruned in deep layers when the pruning ratio is not too large. Accordingly, we can prune more channels in deep layers, especially channels with little information.

To verify the ability of strengthening the target with the proposed compression method, we use the information ratio of the target to the local image patch to represent the discriminative ability of features. Figure 6 shows plots of the real-time target information ratio for different videos. RGB denotes the original image patch, and CNN represents the features from pre-trained CNN with PCA. CPd denotes the proposed method with dimensionality reduction only. CP improves CPd with channel selection, and CPr improves CP with weight reconstruction. Dog is a video where the background has a color similar to that of the target, but is stable and the target has deformation. $CarDark$ is a video where the background is similar to the target and highly unstable. Our method increases the average target information ratio by around two times compared to the original pre-trained CNN. Figure 6 shows that our method can improve the target representation ability and is resistant to target deformation and background clutter. We can see that the target information ratio is improved considerably in the first several frames. This is because our method is trained with the first frame. Although the advantage will reduce with time, our method still outperforms the original pre-trained CNN after long-time tracking, and the proposed weight reconstruction method can improve the stability significantly.

The Alexnet is also tested to confirm the effectiveness of proposed method. Alexnet has five convolutional layers. The fifth convolutional layer is the last convolutional layer for dimensionality reduction, and the first and third layers are common. However, the convolutional operations in the second and fourth convolutional layers are calculated with two parts. In such cases, the two parts are thought as two inputs of a same convolutional layer, and their pruning losses

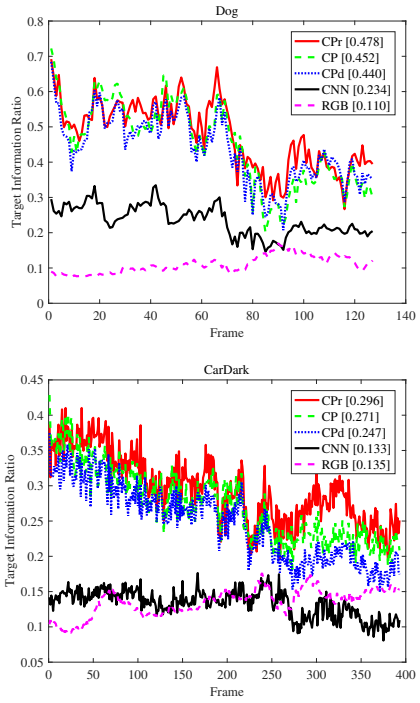

Fig. 6. Plots of target information ratio for the frames of the videos Dog and $CarDark$. The legend states the mean information ratio of all frames.

are simply added up. Figure 7 presents the tracking results of Alexnet. "alexnet" indicates the tracking results with the original Alexnet. "CPT alexnet" indicates optimized Alexnet with proposed dimensionality reduction, and "CPT alexnet1" indicates optimized Alexnet with proposed pruning methods. Figure 7 shows the proposed method can optimize the structure of Alexnet, and generate better features for tracking.

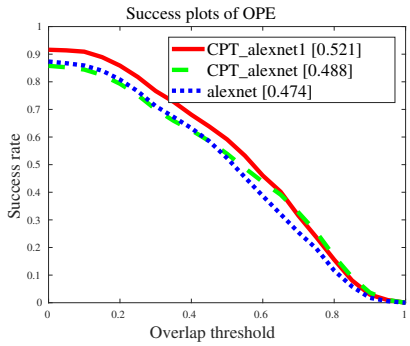

Fig. 7. Tracking results with the Alexnet in ten videos mentioned in Fig. 5.

The efficiency of our method is evaluated. In tracking, the initial time for each sequence is not considered by tracking speed. Because the initial operation is conducted once, whereas the tracking frames would be hundreds or thousands. Here, the average time cost by network compression is about one minute for GPU. Table II compares different pruning configurations in terms of several metrics. The configurations are the pruning ratio for different layers. "0.75/Conv4-3" implies that the layer Conv4-3 is pruned by 75%, and "0.25/all" means that all layers are pruned by 25%. "Base" denotes the original network without any pruning. The FLOPs and Params columns indicate the percentage of remaining CNN FLOPs and parameters, respectively. The feature and tracking columns indicate the speed of feature extraction and tracking in terms of FPS in both CPU and GPU environments. It is noted that only the CNN processing is conducted in GPU, whereas other parts of tracking are conducted in CPU. Pruning deep layers can reduce additional parameters, because there are more channels in deep layers. The reduced FLOPs are similar for different layers, because the FLOPs are influenced by the size of feature maps, and feature maps in shallow layers have a larger size. As the pruning ratio of all layers increases, the numbers of FLOPs and parameters decrease rapidly. The CPU speed can be improved significantly with channel pruning, because the feature extraction requires too much time. In the GPU environment, the improvement has a bottleneck. In addition to processing the image with the CNN, feature extraction in tracking also involves reshaping the image patch, interpolation, regularization, and some other calculations. Pruning channels can lead to a speedup of 7 times or more in CPU and 2 times in GPU for visual tracking.

TABLE II
EFFICIENCY COMPARISONS OF DIFFERENT PRUNING CONFIGURATIONS.

Configuration Ratio/Layer	FLOPs Ratio	Params Ratio	Feature(FPS) (CPU/GPU)	Tracking(FPS) (CPU/GPU)
Base	1	1	5.08/74	2.3/25
0.5/Conv4-3	0.934	0.846	5.29/102	2.59/29
0.75/Conv4-3	0.901	0.768	5.34/144	2.74/34
0.875/Conv4-3	0.884	0.73	5.61/112	2.84/40
0.5/Conv4-2	0.81	0.556	5.89/199	3.01/50
0.5/Conv4-1	0.785	0.498	5.89/183	3.04/48
0.5/Conv3-3	0.785	0.614	5.98/170	3.05/48
0.5/Conv3-2	0.752	0.652	6/158	3.1/46
0.5/Conv3-1	0.785	0.672	5.8/129	3.12/45
0.5/Conv2-2	0.785	0.7	5.96/153	2.98/45
0.5/Conv2-1	0.785	0.715	5.91/170	3.07/45
0.5/Conv1-2	0.785	0.722	6.13/162	3.18/49
0.5/Conv1-1	0.815	0.727	6.29/140	3.11/38
0.25/all	0.502	0.418	7.97/194	4.17/48
0.5/all	0.227	0.192	14.06/183	7.02/52
0.75/all	0.06	0.053	32.86/186	15.2/52

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

The proposed visual tracking method with channel pruning, called CNNcompress, is compared with nine state-of-the-art trackers, namely DeepSRDCF [57], HCF [14], CNNSVM [33], CSRDCF [32], SRDCF [30], LCT [58], SAMF [27], DSST [28], and Struck [25]. For state-of-the-art performance, we prune several deep layers of VGG16 and keep the shallow layers unchanged. Unlike common deep compression methods that focus on compressing the shallow layers for classification problems, we solve the visual tracking problem where the performance is sensitive to feature variations and information loss. With only one sample, pruning all layers can lead to good performance in most videos but not in difficult videos. As discussed above, pruning the last several layers in pre-trained CNN can lead to sufficient improvement in terms of both efficiency and precision. Specifically, the Conv4-3 layer is reduced to 64 dimensions, the Conv4-2 and Conv4-1 layers are pruned by 25%, and the Conv4-1 layer undergoes weight reconstruction.

We report the overall performance for one-pass evaluation (OPE) in all 100 videos, as shown in Figure 8. Our method

yields the best DP score of 0.855 and the second best OP score of 0.617. The DP score represents location precision whereas the OP score takes scale estimation into account. We also test SAMF with CNN features and obtain a DP score of 0.815 and an OP score of 0.567. With channel pruning, we obtain better features compared to the original CNN. Therefore, our tracker locates the target precisely. Specifically, we achieve an improvement of 4% in terms of DP and 5% in terms of OP. However, we only use a single layer Conv4-3 whose resolution is low to track, and it results in inaccurate scale estimation. Nevertheless, we still achieve a state-of-the-art overlap success rate. This indicates the power of good features in our methods. DeepSRDCF, HCF, and CNNSVM are based on pre-trained CNNs, and the others are based on hand-crafted ones. CNN-based methods have better location precision than hand-crafted ones because CNN features have better target semantics and are robust against target deformation. The hand-crafted features have advantages in scale estimation, because they usually have high-resolution representations. DeepSRDCF uses multiple CNN layers and hand-crafted features simultaneously; hence, it has good scale estimation and good location precision. Among the trackers, DeepSRDCF, HCF, LCT, SAMF, and DSST all use improved KCF methods. Our method is based on the SAMF method, and we achieve promising performance with good features generated by the pruned CNN.

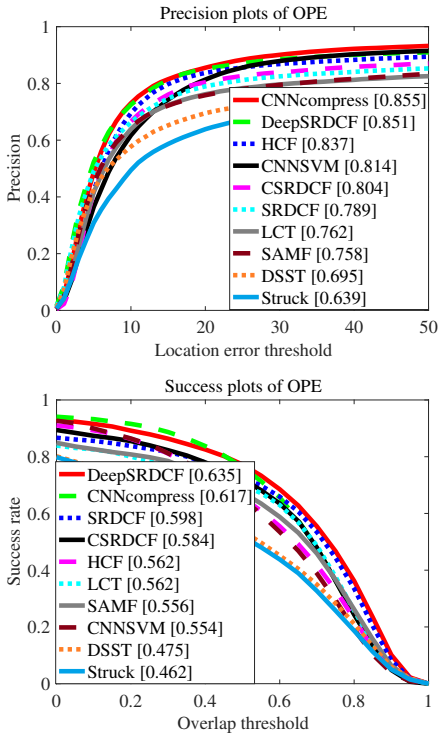

Fig. 8. Distance precision and overlap success plots of OPE in OTB100. The legends show the average DP scores at a location error threshold of 20 and the OP AUC scores.

The 100 videos in OTB100 are annotated with 11 attributes. The attributes represent the challenges appearing in the labeled videos, and each video is labeled with several challenges, such as illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), low resolution (LR), fast motion (FM), out of view (OV), background clutter (BC), in-plane rotation (IPR), and out-of-plane rotation (OPR). The tracking results of the 11 challenges are shown in Figure 9. Our tracker performs the best under seven challenges, namely scale variation, deformation, motion blur, low resolution, fast motion, illumination variation, and out-of-plane rotation. Deep CNN features are invariant when the target changes; thus, the CNN-based trackers DeepSRDCF, HCF, CNNSVM, and our CNNcompress have good locating ability. However, KCF-based methods will result in decay of the features. The feature values away from the center of the image patch will decrease. The tracker will fail if the target deviates from the center of the image patch. Our method prunes channels with little target information. This can reduce the feature values in backgrounds. The target feature values are relatively increased. Hence, the target is highlighted and is easy to track in fast motion and low resolution cases. The feature invariance of deep CNN layers has a side effect in visual tracking because the target is a special instance of a category. The discriminative power against similar objects of the same category is poor. With channel pruning, only the target information in the first frame is preserved, and the feature invariance for objects of the same category is reduced. The pruned network may not recognize the target after drastic change, but updating the KCF tracker can compensate the loss. Structurally optimizing the networks makes our tracker find the target easily, and the center positions of different target postures are judged precisely. In the occlusion and out-of-view cases, DeepSRDCF performs better because a large search area is used. HCF performs the best in the background clutter environment. HCF uses three CNN layers and trains three KCF trackers to achieve fine segmentation. Overall, the results demonstrate that channel pruning can improve the representative and discriminative ability for a specific target.

To further analyze the effectiveness of channel pruning for visual tracking, we compare the four CNN-based trackers in 10 challenging sequence. The tracking results of the trackers are shown in Fig. 10. In $Basketball$, $Bolt2$, and $Freeman1$, all trackers can find the target correctly. However, our tracker gives the most precise prediction. In $Biker$, fast motion and deformation occur at the same time; our tracker and DeepSRDCF still track the target successfully. In Box and $Lemming$, occlusion occurs several times; nevertheless, our CNNcompress always shows good tracking performance. DeepSRDCF has a large search region; hence, it can rediscover the target after a short failure interval. In $Human9$ and $Walking2$, DeepSRDCF drifts to the background and similar objects, respectively. Our tracker has a lower probability of drifting. In $Ironman$, our tracker fails to track the target because the background is similar to the target and both the target and the background change quickly. In $Singer2$, DeepSRDCF, HCF, and CNNSVM cannot follow the target. In this case, the features generated from the CNN in the target are smaller than those in the background, and the trackers drift to the background in the first several frames. Owing to the proposed channel pruning method, the pruned networks can generate discriminative features and our tracker follows the

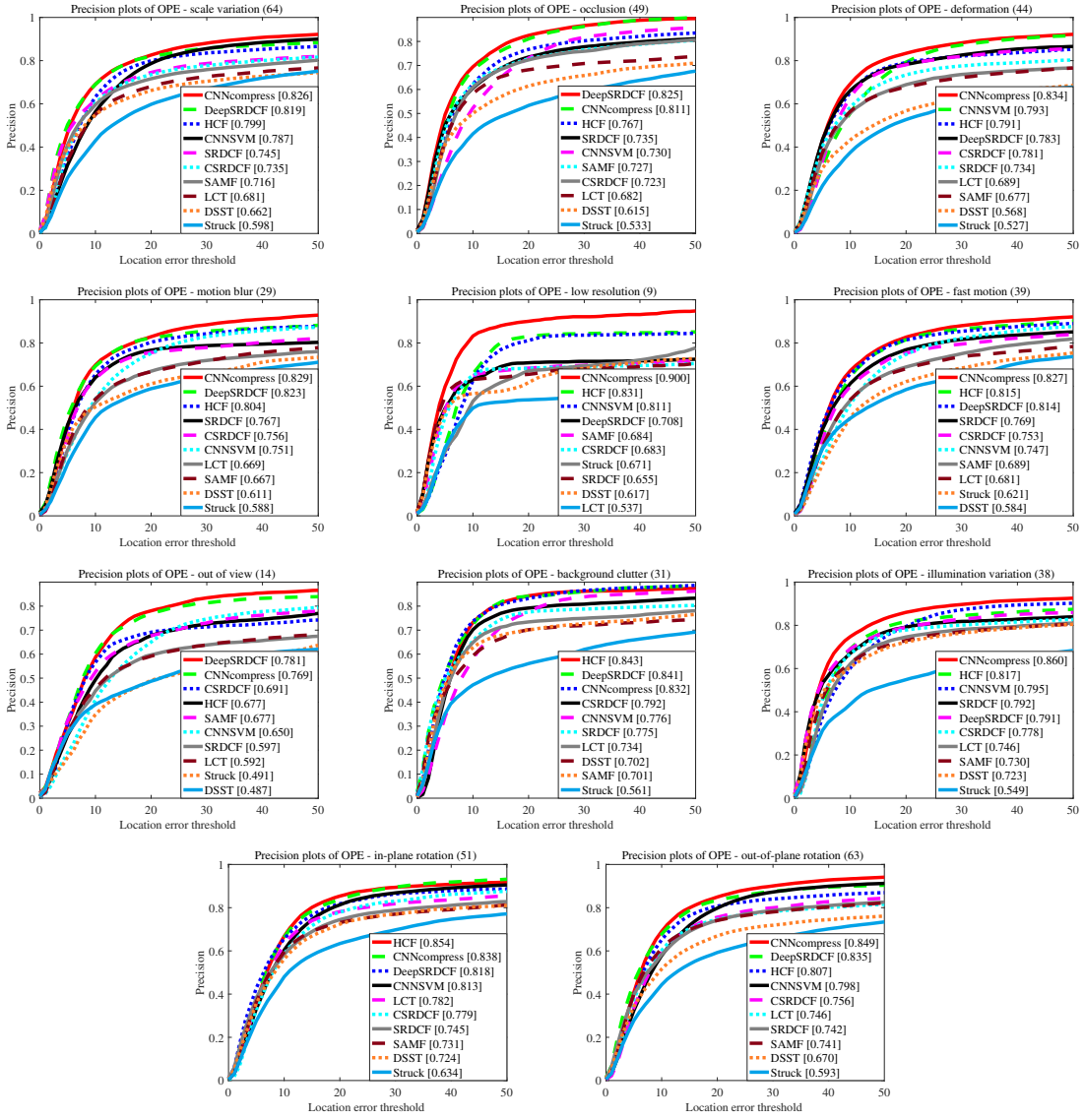

Fig. 9. Distance precision plots of different challenges in OTB100.

target successfully. Our tracker is more robust than the other three trackers. Common CNN-based trackers are weak in scale estimation. This will result in the background information being updated into the model, and the error is magnified with time. However, with the proposed channel pruning method, the background information around the target is low. Although our tracker predicts a bigger bounding box than the actual target, the extracted background features have small values and will not affect the tracker.

V. Conclusion

In this paper, we presented a channel pruning method for visual tracking. We aimed to apply pre-trained CNNs to a visual tracking problem. A dimensionality reduction method, backward channel selection method, and one-sample weight reconstruction method were proposed to customize the channel pruning method for visual tracking. We found that pruning deep layers is easy and beneficial for tracking in most cases, whereas pruning shallow layers is difficult with only one sample. Promising performance could be achieved with features from one layer in a compressed CNN and a simple scale strategy involving KCF implementation. By optimizing the structure of pre-trained CNNs, we can improve the tracking performance by 5% in terms of precision and achieve a speed-up of 2 times for GPU and 7 times for CPU. In the future, we will further explore multiple layer fusion in CNN and one-shot learning in visual tracking.

References

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

[2] M. Kristan, J. Matas, and A. Leonardis, "The visual object tracking vot2015 challenge results," in *IEEE Int. Conf. Comput. Vis. workshops*, 2015, pp. 1–23.

[3] L. C. Zajc, A. Lukezic, A. Leonardis, and M. Kristan, "Beyond standard benchmarks: Parameterizing performance evaluation in visual object tracking," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3343–3351.

Fig. 10. Tracking results of 4 CNN-based trackers in 10 challenging video sequences. The 10 videos are $Basketball$, $Biker$, Box, $Human9$, $Bolt2$, $Ironman$, $Lemming$, $Singer2$, $Freeman1$, and $Walking2$, from top to bottom, and from left to right.

[4] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1134–1143.

[5] S. Li and D. Yeung, "Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models," in *Proc. AAAI*, 2017, pp. 4140–4146.

[6] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 445–461.

[7] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark." *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5630–5644, 2015.

[8] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, 2008.

[9] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* IEEE, 2009, pp. 983–990.

[10] 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, 2015.

[11] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6931–6939.

[12] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 809–817.

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

[14] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3074–3082.

[15] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M. H. Yang, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4303–4311.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.

[18] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5000–5008.

[19] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 523–531.

[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. ICLR*, 2016.

[21] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *IEEE Int. Conf. Comput. Vis.*, 2017.

[22] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *Int. J. Comput. Vis.*, vol. 111, no. 2, pp. 213–228, 2015.

[23] T. Zhang, S. Liu, N. Ahuja, M. H. Yang, and B. Ghanem, "Robust visual tracking via consistent low-rank sparse learning," *Int. J. Comput. Vis.*, vol. 111, no. 2, pp. 171–190, 2015.

[24] T. Zhang, A. Bibi, and B. Ghanem, "In defense of sparse tracking: Circulant sparse tracker," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3880–3888.

[25] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, 2016.

[26] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* IEEE, 2010, pp. 2544–2550.

[27] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. Eur. Conf. Comput. Vis.*, 2014.

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

[29] S. Wang, D. Wang, and H. Lu, "Tracking with static and dynamic structured correlation filters," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 28, no. 10, pp. 2861–2869, 2018.

[30] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *IEEE Int. Conf. Comput. Vis.*, 2015.

[31] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *IEEE Int. Conf. Comput. Vis.*, 2017.

[32] A. Lukežič, T. Vojíř, L. Čehovin, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

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

[34] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016.

[35] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

[36] M. Wang, Y. Liu, and Z. Huang, "Large margin object tracking with

circulant feature maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.
[37] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 749–765.
[38] 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. Vis. Workshops*, 2016, pp. 850–865.
[39] K. Chen and W. Tao, "Once for all: A two-flow convolutional neural network for visual tracking," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 28, no. 12, pp. 3377–3386, 2018.
[40] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. H. Lau, and M. Yang, "CREST: convolutional residual learning for visual tracking," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2574–2583.
[41] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 105–114.
[42] J. Gao, T. Zhang, X. Yang, and C. Xu, "Deep relative tracking," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1845–1858, 2017.
[43] Z. Chi, H. Li, H. Lu, and M. Yang, "Dual deep network for visual tracking," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 2005–2015, 2017.
[44] Z. Cui, S. Xiao, J. Feng, and S. Yan, "Recurrently target-attending tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1449–1458.
[45] J. S. S. III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 322–331.
[46] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-driven visual object tracking with deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2239–2252, 2018.
[47] Q. Wang, Z. Teng, J. Xing, , and J. Gao, "Recurrently target-attending tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
[48] Y. Song, C. Ma, X. Wu, , and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
[49] B. Li, W. Wu, Z. Zhu, and J. Yan, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
[50] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1379–1387.
[51] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, 2016.
[52] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. ICLR*, 2017.
[53] J. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5068–5076.
[54] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
[55] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 747–763.
[56] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foun. Tren. Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[57] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *IEEE Int. Conf. Comput. Vis. workshops*, 2015.
[58] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5388–5396.

Chang Liu is a Ph. D. candidate at the College of Computer Science and Technology, Harbin Institute of Technology. He received his bachelor's degree of Computer Science and Technology from Harbin Institute of Technology in 2014. His research interest covers computer vision and pattern recognition.

Peng Liu is an associate professor at the College of Computer Science and Technology, Harbin Institute of Technology. He received his Ph. D. degree of microelectronics and solid state electronics from Harbin Institute of Technology in 2007. His research interest covers image processing, computer vision, and pattern recognition.

Wei Zhao is an associate professor at the College of Computer Science and Technology, Harbin Institute of Technology. She received her Ph. D. degree of computer application technology from Harbin Institute of Technology in 2006. Her research interest covers computer vision and pattern recognition. Corresponding author of this paper.

Xianglong Tang is a professor at the College of Computer Science and Technology, Harbin Institute of Technology. He received his Ph. D. degree of computer application technology from Harbin Institute of Technology in 1995. His research interest covers pattern recognition.