Feature Distilled Tracking

Guibo Zhu, Jinqiao Wang, *Member, IEEE*, Peisong Wang, Yi Wu, and Hanqing Lu, *Senior Member, IEEE*

***Abstract*—Feature extraction and representation is one of the most important components for fast, accurate, and robust visual tracking. Very deep convolutional neural networks (CNNs) provide effective tools for feature extraction with good generalization ability. However, extracting features using very deep CNN models needs high performance hardware due to its large computation complexity, which prohibits its extensions in real-time applications. To alleviate this problem, we aim at obtaining small and fast-to-execute shallow models based on model compression for visual tracking. Specifically, we propose a small feature distilled network (FDN) for tracking by imitating the intermediate representations of a much deeper network. The FDN extracts rich visual features with higher speed than the original deeper network. To further speed-up, we introduce a shift-and-stitch method to reduce the arithmetic operations, while preserving the spatial resolution of the distilled feature maps unchanged. Finally, a scale adaptive discriminative correlation filter is learned on the distilled feature for visual tracking to handle scale variation of the target. Comprehensive experimental results on object tracking benchmark datasets show that the proposed approach achieves 5× speed-up with competitive performance to the state-of-the-art deep trackers.**

***Index Terms*—Correlation filter, model compression, visual tracking.**

I. Introduction

VISUAL tracking is a fundamental problem in computer vision with a wide range of applications, such as visual surveillance, autonomous driving, and virtual/augmented reality [1]. It refers to the task of estimating the trajectory of the target whose initial location is only given in the first frame of an image sequence. Despite significant progress in recent decades, this problem is still very challenging due to large appearance changes caused by several factors, e.g., occlusions, scale variation, and background clutter.

Deep convolutional neural network (CNN) has demonstrated impressive performance in various visual tasks, such as image classification [2], [3] and detection [4]–[6]. The essential factor to success is to learn rich invariant features coming from multiple nonlinear transformations based on various deep learning architectures and large training datasets. Some of recent works [7], [8] exploit that visual tracking can not only benefit from the prior knowledge with spatial-temporal structure in video sequence, but also take advantage of rich feature with deep learning. There are many works [4], [9], showing that features extracted by very deep or wide networks have good representative and generalization ability. That is because the pretrained CNN models provide a good prior knowledge trained from other image dataset [10]. It depends on the well-known theoretical guarantee to the representation capacity of neural networks to a certain degree [11]. Although very deep or wide networks usually appear as best-performing systems, they are time-consuming and require large memory with multitudinous parameters in the training and inference processes. Therefore, they are not appropriate for applications with limited computing resources. To handle the problem, many model compression methods have been proposed to reduce the computational cost and the model size [12]–[15]. These model compression methods aim to obtain a fast and compact model for approximating the structure function and the hierarchical representation learned by a slower, larger deep model. In other words, there are several techniques for enabling machines to learn from other machines, e.g., distillation [14] and privileged information [16]. In this paper, we investigate the key problem of how to exploit the rich features extracting by a small mimic student CNN learned from a larger and deeper teacher CNN while preserving the discriminative and representative power of the original teacher CNN for online visual tracking.

Recently, discriminative correlation filters (DCF) based approaches [17]–[19] have gained much attention due to its high efficiency and robustness for the tracking problem. DCF-based methods mainly learn an optimal correlation filter from a limited number of training samples, in which circular structure is utilized to achieve the performance of dense sampling. Based on the convolution theorem, the filter is not only learned fast but also used tractable for quick inference in predicting the target state. In addition, the training samples can be represented by low-level hand-crafted features or deep features extracted by pretrained CNN using a large set of auxiliary datasets. Previous work [18], [19] has shown that, there exists a strong positive correlation between the feature and DCF tracker, i.e., the stronger the feature's generalization ability is, the better the DCF tracker performs. Actually, we utilize a teacher–student paradigm, where VGG-19 [4] is used to guide the learning of the student network for tracking. Considering the universal approximation theorem [11], CNNs can approximate a wide variety of functions. Therefore, we use the small

Manuscript received June 1, 2016; revised February 17, 2017; accepted November 12, 2017. This work was supported by the National Natural Science Foundation of China under Grant 61702510, Grant 61773375, Grant 61370036, Grant 61772277, and Grant 61772527. This paper was recommended by Associate Editor M. Shin. *(Corresponding authors: Jinqiao Wang; Yi Wu.)*

G. Zhu is with the Research Center for Brain-Inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: guibo.zhu@ia.ac.cn).

J. Wang, P. Wang, and H. Lu are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: jqwang@nlpr.ia.ac.cn; peisong.wang@nlpr.ia.ac.cn; luhq@nlpr.ia.ac.cn).

Y. Wu is with the School of Technology, Nanjing Audit University, Nanjing 211815, China, and also with the Department of Medicine, Indiana University School of Medicine, Indianapolis, IN 46202 USA (e-mail: ywu.china@gmail.com).

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

Digital Object Identifier 10.1109/TCYB.2017.2776977

CNNs to approximate the mapping function from images to the intermediate layers of VGG-19.

To utilize the good generalization ability of deep features from large convolutional networks and high efficiency of DCFs in accelerating the whole tracking system, we propose to learn a feature distilled network (FDN) for faster extracting representative features while distilling the knowledge from a larger convolutional network. Then the extracted features from FDN is embedded in an DCF-based framework with scale estimation and occlusion judging for robust and fast tracking.

The outline of the remaining part of this paper is as follows. We discuss related work about visual tracking and model compression in Section II. Section III introduces the proposed FDN that transfers knowledge from a large teacher network. Then we demonstrate the details of feature distilled tracker in Section IV. Experimental results are given in Section V. Section VI concludes this paper.

II. Related Work

Due to the importance of the tracking problem, it has been studied extensively by many researchers over the years. Since a comprehensive review of the tracking methods is beyond the scope of this paper, we recommend the reader to [20] and [21] for a survey, and also to [22]–[24] for some empirical comparisons. In this section, we first focus on some representative trackers in the view of feature representation: tracking with hand-crafted features and tracking with deep features, and then introduce the model compression techniques.

A. Tracking With Hand-Crafted Features

Traditional object tracking algorithms mainly focus on appearance modeling, which can be categorized roughly into generative and discriminative methods [21]. Generative tracking approaches commonly learn an object reference model to locate the object by searching for the most similar image region. For example, dictionary-based trackers construct dictionaries of local or global object templates and use the integration of these templates to represent object candidates in next frame. A common method is to depict the candidates with sparse representation using $\ell_{p,q}$-norm minimization [25]–[31]. The object candidates are often generated by particle filtering [32] which is appropriate for motion translation and scale variation. Then an observation model is used to give higher confident score to candidates that are represented by the templates with less construction error. Mei *et al.* [25] was to represent the candidates sparsely using ℓ_1 norm minimization. Zhong *et al.* [26] proposed a sparsity-based collaborative model (SCM) by exploring holistic and local information. Wang *et al.* [27] described an online robust dictionary learning method with non-negative constraints for adaptively capturing the distinctive aspect of the object appearance. Xing *et al.* [28] aimed to automatically learn good templates online for constructing a multilifespan dictionary model with short-term, medium-term, and long-term dictionaries which are used to leverage the stability and plasticity residing online update in visual tracking. Zhang *et al.* [29] modeled particles as linear combinations of dictionary templates and mined the structure relationship between these particles. Ma *et al.* [30] learned multiple linear and nonlinear subspaces with key samples so as to model the local distributions of object appearances. Zhang *et al.* [31] proposed a context-aware exclusive sparse tracker by representing the particles with dictionary templates which included contextual information for alleviating the model drift problem. Zhang *et al.* [33] designed a simple two-layer convolutional networks without training for visual tracking.

Discriminative tracking methods [34], [35] have shown to provide good tracking performance. These approaches work by posing the prediction task of target localization in the tracking problem as a binary classification problem or tracking-by-detection problem or regression problem. The classification-based trackers need a set of labeled training samples for incrementally learning a discriminative classifier with the decision boundary to separate the object from background in each frame. However, there is the sampling ambiguity problem arising with such approaches which draw samples in the previous object location predicted by the learned classifier. Slight inaccuracies in the labeled samples will degrade the classifier and gradually result the tracker to drift. There are many algorithms to handle these model update problems resulting from sample ambiguity so as to alleviate model drifts, such as multiple instance learning (MIL) tracker [36], tracker-learning-detection (TLD) tracker [37], structure preserving object tracking [38], and transfer learning with Gaussian processes regression (TGPR) [39]. Babenko *et al.* [36] posed object tracking as an online MIL problem, where the samples are considered within the positive or negative bags which are used to train a Boosting classifier online. Kalal *et al.* [37] proposed the unified TLD framework where there are a short-term tracker, random fern classifier as long-term online detector and P-N learning strategies help each other by exploring the structure of unlabeled data, i.e., the short-term tracker offers the training samples with high confidence to train and update the detector, and the short-term tracker is reinitialized by the detector when it fails. Supancic and Ramanan [40] proposed a self-paced learning framework for long-term tracking. Zhang and van der Maaten [38] proposed a structure preserving model with graphical structure by incorporating spatial constraints to alleviate the model drift problem in the tracking-by-detection framework to some extent. Zhang *et al.* [41] proposed a novel tracking algorithm with multiple experts using entropy minimization (MEEM) for revising the quality of the past training samples. TGPR [39] learned the probability of target appearance with Gaussian processes regression with auxiliary samples remembered from the very early frames and target samples from most recent frames in a semi-supervised fashion. Random projections were introduced to select features fast for visual tracking [42], [43]. On the other hand, Hare *et al.* [35] proposed structure output tracking with kernels (Struck) by exploring the spatial label distribution of the training samples as the intrinsic relative structure, which alleviated the problem of label prediction about noise samples (i.e., label ambiguity). According to the performance in the evaluation by third-party tracking benchmarks [23], the sample ambiguity with correlation filters can

be addressed by regressing the training samples with soft labels of a Gaussian or Laplacian function which is better than binary labels for learning discriminative classifier. Wang *et al.* [44] proposed a hybrid method by designing a novel generative method measured by weighted local cosine similarity and learning discriminative weights for visual tracking. Zhu *et al.* [45] proposed the part context learning to exploit the spatial–temporal hierarchical relationships for visual tracking. Liu *et al.* [46] proposed a vector boosting feature selection for robust visual tracking.

DCFs have been investigated by many researchers in the context of visual tracking. Bolme *et al.* [17] proposed to learn an adaptive correlation filter with the samples of the target appearance by minimizing the output sum of squared error. The convolution theorem is used for fast training and detection in the tracking process. Kernelized correlation filters (KCFs) [19] utilized the circulant structure of adjacent subwindows for dense sampling and used kernelized regression to reinterpret the correlation filter with multichannel features. Danelljan *et al.* [18], [47] introduced color attributes in the view of feature representation and designed an independent and efficient scale estimation filter, respectively, to utilize the color property in colorful tracking sequences and handle the scale variation accurately. Zhu *et al.* [48] proposed collaborative correlation tracking, online template clustering with grid search [49], and saliency proposals [50] for handling the model drift problem. Spatially regularized DCF [51] introduced a spatial regularization term to penalize the correlation filter coefficient depending on their spatial location, and proposed a novel learning algorithm to achieve the optimal filter with spatial constraint. Zhu *et al.* [52] proposed EdgeBox tracking where instance-specific proposals were generated by EdgeBox [53] and the proposals were integrated with online SVM for tracking. Danelljan *et al.* [54] proposed a novel formulation by optimizing the objective loss with both the target appearance model and the training sample quality weights for visual tracking from the perspective of the training set. All of these methods study different intrinsic characteristics of correlation filters for handling the facing major issues and improving the performance of the tracking problem, e.g., circulant structure [19], kernel trick [19], color attributes [47], HOG feature [18], [19], and spatial constraint [51].

B. Tracking With Deep Features

Feature extractor is one of the most components in an object tracking system which includes motion model, feature extractor, observation model, model updater, and ensemble post-processor [55]. Great successes of CNNs on many visual tasks have been witnessed. In the object tracking domain, Wang and Yeung [7] proposed a deep learning tracker by obtaining deep compact and informative image representations with a stacked denoising autoencoder network based on an auxiliary large dataset. Li *et al.* [56] proposed a data-driven model of multiple CNNs without pretraining for visual tracking. Hong *et al.* [57] proposed to learn discriminative map using pretrained CNN. Ma *et al.* [8] proposed hierarchical convolutional features (HCF) extracted from a pretrained CNN by exploring the spatial resolution information. Wang *et al.* [58] proposed online tracking with fully CNN considering the properties of CNN features offline. Nam and Han [59] described a multidomain CNNs which used a large set of videos with tracking labels to train a CNN for extracting universal target representations and then added a domain-specific layer adaptively for each sequence with online hard samples mining. We note that the aforementioned CNN trackers mainly pay attention to the performance of the trackers, few considering the running speed. Whereas, the running speed is a key factor for real-time applications. Visual tracking requires not only accuracy but also speed with device limitation, such as common mobile phones which have neither GPUs nor high computing power. We try to learn a small and fast student network while preserving the comparable tracking performance to satisfy more general cases.

C. Model Compression

In many commercial applications, it is much more crucial to reduce the time and memory cost of running a pattern recognition model in the inference process than in the training process. It is a cost sensitive and balanced problem [60]. For the situations which do not need self-adaptation and self-learning, it may only need to train a model once, then publish it for millions of terminal deployments. In many situations, the terminal is more resource-limited than the server. One useful method for reducing the cost of inference is to use model compression [12].

The main idea of model compression is to use a fast and compact model which requires less memory and runtime in storage and inference to approximate the intrinsic function learned by a slower, larger, but well performing model. Deep models are usually over-parameterized to facilitate convergence during training phase. However, the redundancies are not necessary in inference phase. There are many different learning methods for model compression and applications with the compressed model in various domains. Ba and Caruana [13] trained a shallow model to mimic a more accurate deep model. ShrinkConnect [61] was proposed to reduce the computation of the location-sensitive deep network for cross-modality image synthesis. Hinton *et al.* [14] introduced knowledge distillation (KD) as a model compression framework by distilling the knowledge in ensemble models into a single student model, i.e., imitating the soft output of a larger teacher network or ensemble of networks in a teacher–student paradigm. Romero *et al.* [15] designed to train a deeper but thinner student with the teacher, using not only the predicted label outputs but also the intermediate representations of the teacher as hints to supervise the training process so as to boost the student performance.

Inspired by the observation that there are many redundancies in neural network parameters [62], Denton *et al.* [63] exploited filter clustering and the low-rank approximation by applying singular value decomposition to pretrained CNN model for efficient evaluation. Jaderberg *et al.* [64] also used low-rank decompositions of CNN filter banks with filter reconstruction

optimization and data reconstruction optimization to speed up the evaluation of CNNs for fast scene text classification.

LeCun *et al.* [65] used second derivatives with information-theoretic view to remove redundant parameters from a pretrained network. Han *et al.* [66] proposed to achieve efficient neural networks by first learning important connections, then pruning the unimportant connections, and finally fine-tuning the remaining connections for deploying on embedded systems. Subsequently, Han *et al.* [67] again extended their work to compress the deep neural networks with pruning by learning. It preserved the most important connections, and quantized the network weights with sharing, and then used Huffman coding to give more compression without affecting their classification accuracy. Further, Han *et al.* [68] designed a hardware accelerator called EIE which runs directly on the compressed model for the goal of sequential hardware acceleration and resource savings.

High precision parameters need many memory to store which may not be necessary in deep neural networks. Gong *et al.* [69] investigated different vector quantization techniques to compress the weight parameters of CNNs. Low-precision fixed point quantization-based methods were also investigated by the community. Arora *et al.* [70] proposed to train a sparse network with $+1/0/-1$ weights. Vanhoucke *et al.* [71] proposed to replace 32-bit floating point activations with a fixed-point implementation of 8-bit integer. Another fixed-point networks or quantized networks were proposed by [72] and [73], which also were applied in object recognition [74]. To further address the model storage problem, network binarization was exploited by the researchers [75]–[79]. Chen *et al.* [80] proposed HashedNets which used the hashing trick to compress neural networks, where the connections were mapped into different hash bucket with a low-cost hash function for reducing model size. Soudry *et al.* [76] proposed expectation backpropagation to train a network with binary weights. Courbariaux *et al.* [78] proposed to use binary weights for forward and backward computation while keeping full-precision weights as reference. Their method achieved comparable or even higher precision than full-precision weights on MNIST, CIFAR-10, and SVHN. They further extended their work and binarized both weights and activations in [79]. Rastegari *et al.* [81] proposed XNOR-Net with binary weights and binary inputs which was also been evaluated on large-scale datasets.

There is another direction for model simplification by designing more elegant and compact model architectures. For example, inspired by inception architecture [82] and residual connections [9], Iandola *et al.* [83] proposed a small CNN architecture called SqueezeNet which has fewer parameters with preserving similar accuracy comparing with AlexNet architecture [84].

Different from the mentioned model compression frameworks which aim to improve the classification accuracy, our algorithm mainly takes advantage of a large public classification dataset (e.g., ImageNet [85]) for learning a small network to mimic a larger and more accurate model so as to obtain effective intermediate representation. The learned intermediate representation as feature is embedded in the DCF-based tracking framework for fast, accurate, and robust tracking. To the best of our knowledge, we are the first to propose a model compression formulation to solve online visual tracking problem.

III. Feature Distilled Network

This section demonstrates the details of our method. We first describes how to learn FDN and use an acceleration method to distill robust and domain-independent convolutional features for visual tracking.

A. Problem Formulation

In the field of machine learning, a softmax function can be used to transform values into class probabilities. The commonly used function is

$$q_i = \frac{\exp(z_i/\tau)}{\sum_j \exp\left(z_j/\tau\right)} \tag{1}$$

where the z_i corresponds to the logit, i.e., the value before softmax activation, τ is temperature parameter which is normally set to 1. The probability distribution gets much softer as the temperature gets higher.

The KD approach [14] seeks to make the output of a student network imitating to the soft output of a larger teacher network or an ensemble of teacher networks. For this purpose, using a higher temperature to generate the softened outputs provides more information due to the information entropy. The information is about the relative similarity of the input to classes, which can not be represented by only the one value with the highest probability. More precisely, they raise the temperature of the final softmax until the teacher network produces a set of target outputs which are soft enough. Then the same temperature is used to train the small student network using the "distilled" output of the teacher network as supervision. To further boost the accuracy, they also train the student network to produce the correct labels. The training process is to optimize the following loss function:

$$\mathcal{L}_{\mathrm{KD}}(W_S) = \mathcal{H}(y_T, y_S) + \lambda \mathcal{H}\left(q_T^\tau, q_S^\tau\right) \tag{2}$$

where y_T and y_S are the supervised classification labels of the teacher networks and the student networks, respectively, $\mathcal{H}$ refers to the cross-entropy and λ is a tunable parameter to balance both cross-entropies. The terms q_T^τ and q_S^τ are the soft output of the teacher network and the student network, respectively, using the same temperature τ, and W_S denotes the model parameter matrix of the student network which needs to be optimized. Please refer to [15] for more details.

B. Feature Distilled Network

In contrast to conventional KD that minimizes the cross-entropies between the soft output of the student network and the teacher network, the object of our FDN is to lower the discrepancy between the outputs (responses) of hidden layers, i.e., the convolutional feature maps, of the teacher network and the student network. The student network is trained to mimic the teacher network's feature space. We imitate the output of hidden layers because the feature maps of these layers provide

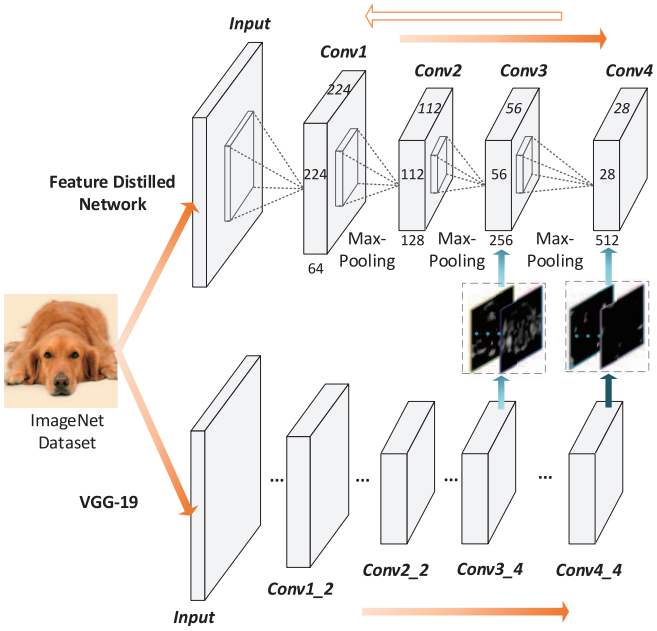


Fig. 1. Illustration of training the FDN. Here, there are many ellipsis layers in VGG-19 than FDN's which have been not shown for brevity.

more useful feature representation for the DCF tracker. We formulate the loss function of teacher–student model compression as a regression problem with a training set $\mathcal{X} = \{\mathbf{x}_i, \mathbf{u}_i\}_{i=1}^{N}$

$$\mathcal{L}(\mathcal{X}) = \frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{u_i} - g(\mathbf{x}_i; \mathbf{W}_S)\|^2 \tag{3}$$

where $\mathbf{x}_i$ and $\mathbf{u}_i$ denote the ith training image and its corresponding chosen features (e.g., conv4_4 features in VGG-19 [4]), respectively. $\mathbf{u}_i$ is extracted from a well-trained large neural network, which is the teacher. In other word, while distilling with a CNN or its ensemble, $\mathbf{u}_i$ is chosen from the intermediate layers of the CNN or CNNs. $\mathbf{W}_S$ indicates a set of parameters of the student network and $g(\cdot)$ represents a general nonlinear transformation function from the input image to the features. Equation (3) is the loss function of learning student network that can be optimized by the stochastic gradient descent with standard back-propagation algorithm [84].

1) Architecture: The architecture of our FDN is shown in Fig. 1. It has four convolutional layers and three pooling layers and receives the same input image as VGG-19 which has 16 convolutional layers, five pooling layers, three fully connected layers, and a softmax layer [4]. We drop the fully connected layers and softmax layer because the object of visual tracking aims to locate targets precisely rather than to infer their semantic classes (target or background) where the fully connected layers mainly encode the semantic information. Specifically, the first convolutional layer consists of 64 filters with the receptive field of 7×7, the second includes 128 filters of size 5×5, the third has 256 filters of size 3×3, and the last layer owns 512 filters of size 3×3.

The proposed student network architecture is not only substantially smaller than the teacher network VGG-19 [4], but also preserves the fine structure of the teacher network so as to obtain good generality for different visual tasks (e.g., object tracking and detection). There are some important advantages with our network for visual tracking. First, a deep CNN has some good hierarchies of features from coarse to fine in spatial resolution [8]. Second, even if the target objects in visual tracking are commonly small, the input size may be sufficient considering to the context region in an appropriate tracking framework, e.g., DCF [17], [19]. Finally, a smaller network is more efficient in visual tracking problem, where online update and inference of object appearance model are necessary. Although the tracking algorithm may be more accurate with performing larger networks, it becomes slower significantly.

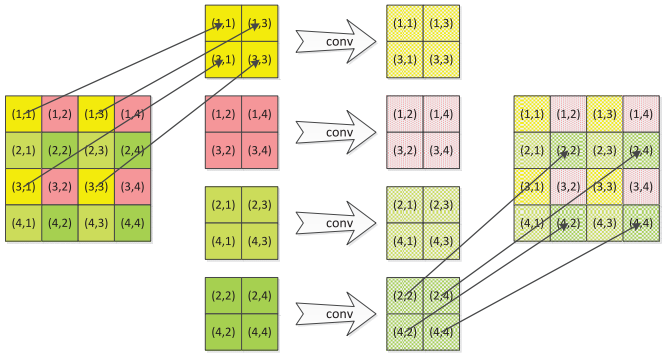


Fig. 2. Illustration of the multipooled features for finer output.

C. Faster Feature Distilled Network

Once the small student network is learned, the network filters in each layer of FDN can be used as a feature extractor for the DCF tracker. To obtain feature maps for the DCF tracker, a typical method is to alter the window size to the size of 224×224 and extract output features of the chosen layer. Then the feature maps are resized proportional to the searching window [e.g., $(1/4^2)$ of the original searching window]. But for the task of tracking, the target search region is usually much smaller than 224×224. Scaling it up to the size 224×224 may affect the effectiveness of the learned filters, since the network filters are learned for images of "appropriate scales." On the other hand, if we use a smaller input image, e.g., of size 112×112, the output resolution of the last layer will be too coarse for the DCF tracker. Therefore, it is ideal to reduce the size of the input image while keeping sufficient resolution of the output layer for good tracking performance.

To alleviate the above problems, we adapt the shift-and-stitch method in object detection [86] and segmentation [87], to our fast feature distilled network, denoted as faster FDN. Note that the resolution discrepancy between feature maps is on account of strides in the convolutional layers as well as pooling layers. Consider a 2×2 pooling layer for example, the output resolution is $1/4$ of the previous layer. With the goal of getting the dense output of the same size as the previous layer, we process four groups of down-sampled feature maps using shared weights and then stitch results together to get the final high resolution outputs, as shown in shown in Fig. 2.

This operation can be performed more efficiently by modifying the convolutional filters. We drop the last pooling layer

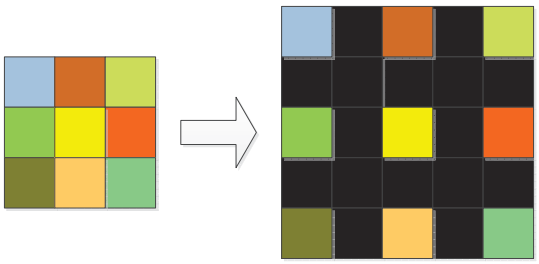


Fig. 3. Illustration of the filter modification for finer output. Different colors refer to different values and black color means the value is zero.

of our learned student network and modify the last convolutional filters by introducing zeros to increase their length (2×), as illustrated in Fig. 3. By using larger filters and smaller input images, our method can not only speed up the feature extraction process but also improve the tracking accuracy.

IV. Feature Distilled Tracking

In this section, we first present DCFs. Then, we discuss how to extract deep features in the proposed FDN and apply the extracted features in tracking. Finally, we introduce the modules of scale estimation and occlusion judge for handling the scale variation and the occlusion problem in tracking.

A. Discriminative Correlation Filter

DCF-based tracking has attracted much attention due to its high efficiency and robustness. The goal of standard correlation trackers [17]–[19] is to learn a multichannel correlation filter $\mathbf{f}$ from a set of training samples $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{t}$. Each training sample $\mathbf{x}_k$ with same spatial size $M \times N$ consists of a d-dimensional feature representation extracted from an image region. In other words, a d-dimensional feature vector $\mathbf{x}_k(m,n) \in \mathbb{R}^d$ is generated at each spatial location $(m,n) \in \{0,1,\ldots,M-1\} \times \{0,1,\ldots,N-1\}$. We denote $\mathbf{x}_k^l$ as the lth feature layer of $\mathbf{x}_k$, where $l \in \{1,\ldots,d\}$. The desired output y_k satisfies a scalar value of some label distribution, e.g., Gaussian, corresponding to each location in the sample $\mathbf{x}_k$. A correlation filter $\mathbf{f}$ is then obtained by optimizing the following minimization problem with L_2-loss:

$$\varepsilon = \sum_{k=1}^{t} \alpha_k \left\| \sum_{l=1}^{d} \mathbf{x}_k^l * \mathbf{f}^l - \mathbf{y}_k \right\|^2 + \lambda \sum_{l=1}^{d} \left\| \mathbf{f}^l \right\|^2. \tag{4}$$

Here, $\alpha_k \geq 0$ denotes the weight of the ith training sample and $\lambda \geq 0$ is the impact of the regularization term. Equation (4) can be treated as a ridge regression problem. Based on Parseval's theorem, the optimization of (4) can be transformed into the Fourier domain for fast training. Here, $\mathbf{x}_k$ can be the distilled feature extracted from the student network.

Then the correlation filter $\hat{\mathbf{f}}$ is applied to estimate the target state in the next frame. For predicting the new object state in next frame, a sliding-window-like manner is necessary by evaluating the correlation scores on all cyclic shifts of a test sample $\mathbf{z}$. Let $\mathbf{z}$ denotes a $M \times N \times d$ feature map extracted from an image region with size $M \times N$, d is the number of feature channels. With the convolution theorem and circulant structure [19], the correlation scores $S(\mathbf{z})$ at all locations in the image region can be computed efficiently

$$S(z) = \mathcal{F}^{-1}\left\{ \sum_{l=1}^{d} \hat{\mathbf{z}}^l \cdot \hat{\mathbf{f}}^l \right\} \tag{5}$$

where $\cdot$ represents point-wise multiplication, the hat denotes the FFT of a function, and $\mathcal{F}^{-1}$ denotes the inverse FFT.

B. Model Update

An optimal filter for locating can be learned and updated by minimizing the output error over all the historical tracked results [17], [19]. According to (4) and common ridge regression optimization method, the numerator $\hat{\mathbf{a}}$ and denominator $\hat{\mathbf{b}}$ of the appearance filter $\hat{\mathbf{f}}$ can be separately learned and updated as follows:

$$\hat{\mathbf{f}} = \frac{\hat{\mathbf{a}}^t}{<\hat{\mathbf{x}}^t, \hat{\mathbf{x}}^t> + \lambda} = \frac{\hat{\mathbf{a}}^t}{\hat{\mathbf{b}}^t + \lambda} \tag{6}$$

$$\hat{\mathbf{a}}^t = (1-\beta) * \hat{\mathbf{a}}^{t-1} + \beta * <\hat{\mathbf{y}}, \hat{\mathbf{x}}^t> \tag{7}$$

$$\hat{\mathbf{b}}^t = (1-\beta) * \hat{\mathbf{b}}^{t-1} + \beta * \sum_{i=1}^{\mathbf{d}} <\hat{\mathbf{x}}_i^t, \hat{\mathbf{x}}_i^t> \tag{8}$$

where t denotes the frame index and β is the learning rate. Actually, while the object is not heavily occluded, this update strategy works well because it can capture the appearance variation slowly.

When the object is heavily occluded or occluded for a long time, some incorrect update for the appearance model may result to the model drift. To alleviate the problem, we simply utilize an occlusion judging method via the correlation score and adaptively adjust the learning rate β in a common manner. If the object appearance is judged as occluded, the learning rate β is reduced; otherwise, keep the initialization value. With the correlation score $\mathbb{T}_o$ and the lower confidence bound $\mathcal{T}$, the learning rate β is adjusted as follows:

$$\beta = \begin{cases} 0.1 * \beta_{\text{init}}, & \text{if } \mathbb{T}_o < \mathcal{T} \\ \beta_{\text{init}}, & \text{otherwise} \end{cases} \tag{9}$$

where β_{init} is the initialization value of the learning rate β.

C. Scale Estimation

Since scale variation is a critical problem in the tracking domain, we need a scale estimation module to boost the tracking performance. A effective method [18] is to learn a independent scale filter to estimate the object scale variation. However, there are two problems about this method which need to be considered in the DCF framework. First, if the object center is predicted accurately, the scale method is good; otherwise, the accumulated location error will be larger and larger, eventually the model drift. Second, online update of scale filter needs to consider the occlusion problem.

To solve the first problem, we provide more fine-grained object location through modifying the learned FDN with shift-and-stitch method mentioned in Section III-C. To alleviate the second problem, we adopt the correlation score of the predicted object location $\mathbb{T}_o$ to evaluate whether to update the scale filter. If $\mathbb{T}_o < \mathcal{T}$, we will reduce the learning rate

of the scale filter similar to Section IV-B to reduce the risk of incorrect update. While we utilize the two strategies, the performance of our FDN has a large gain which can verify the effectiveness of the proposed strategies.

V. Experiments

Here, we perform a comprehensive evaluation of the proposed method on public challenging benchmark datasets, Object Tracking Benchmark (OTB-50) [23] and OTB-100 [88] and compare our proposed FDT tracker with state-of-the-art methods. The datasets are gathered from many public sequences, which pose challenging situations, such as heavy occlusions, deformation, out-of-view, motion blur, illumination changes, scale variation, in-plane and out-of-plane rotations, background clutter, and low resolution. Our tracker is implemented in MATLAB using MatConvNet toolbox [89] and runs on an Intel i7-4770 3.50 GHz CPU or NVIDIA GeForce GTX Titan GPU, CUDA toolkit 6.5.

A. Evaluation Methodology

To evaluate the performance of our proposed approach, we follow the adopted protocol [23], which included two evaluation metrics based on [23] and [36]: distance precision (DP), and overlap precision (OP). DP is calculated as the relative number of frames in the sequence, where the center location error (CLE) is smaller than a certain threshold. We take use of the common DP values at a threshold of 20 pixels. OP is defined as the percentage of frames, where the bounding box overlap exceeds a threshold $t \in [0, 1]$ between the estimated bounding box and the ground-truth bounding box. If the overlap ratio of bounding boxes exceeds 0.5 in one frame, it is considered to be successful in tracking. We compute OP as the average successful overlap rate at some threshold with in all sequences. The results are summarized over both the datasets OTB-50 and OTB-100. We present them with precision and success plots [23]. Specifically, we take area under curve (AUC) as the measure of success plot to rank trackers, where success plot is the average of the success rates corresponding to the sampled overlap threshold.

B. Details and Parameters

Similar to the latest DCF-based tracker with HCF [8], we also employ the deep convolutional features. In addition, the parameters related to the DCF formulation are the same as [8], where the search region ratio is 1.8 larger than the target size, the initialization learning rate η is 0.025, and λ of (4) is 0.0001. In (9), $\mathcal{T} = 0.15$. Different from [8], which explores the hierarchies of features with different spatial resolutions, we mainly study the convolutional features imitating the output of the *conv4-4* hidden layer in VGG-19 [4]. First, it has good generalization ability in many applications, such as, pedestrian and edge detection [90] and tracking [8]. It can also be applied to person reidentification, object classification, or segmentation. Second, empirical experiments on approximating the output of the *conv5-4* hidden layer are not good on the tracking benchmark. This higher level features may need some more advanced deep network to estimation. Therefore the estimation with small network has a little big bias which results to the degrade of performance.

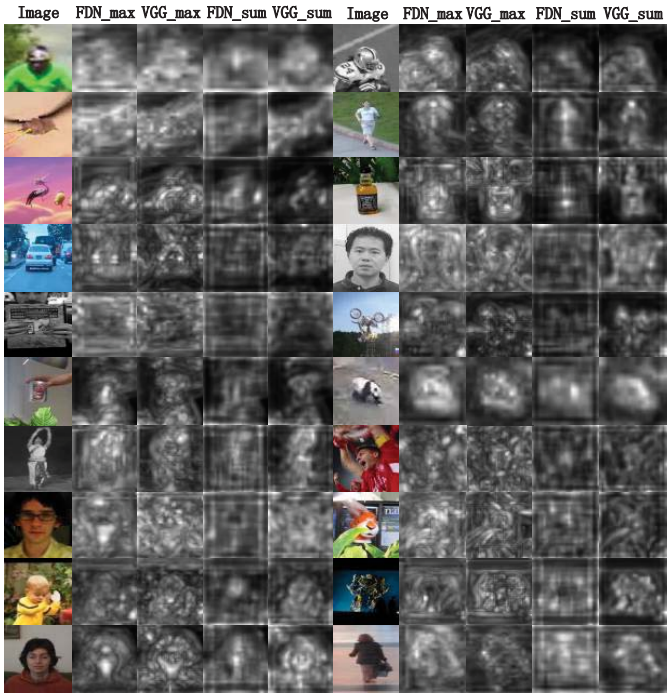


Fig. 4. Visualization of the output *conv4-4* features between VGG-19 and FDN. The displaying images come from OTB-100 dataset [88]. The first column is the input image. The second column is taking the *max* operation among the channel direction of the mimic output features extracted by FDN. The third column is taking the *max* operation among the channel direction of the intermediate *conv4-4* output features extracted by VGG-19. The forth and the fifth are adapting the *sum* operation along the channel direction from FDN and VGG-19 network, respectively. The output features are all normalized by dividing the maximum value for visualization.

C. Feature Visualization Between VGG-19 and FDN

To compare the output features of the hidden layers of the FDN and the corresponding layers of the VGG-19, we perform our experiments on public tracking sequences from OTB-100 dataset [88]. For all sequences, we crop the context region of the object based on the ground truth in the first frame and extract the corresponding intermediate features from the network architecture of VGG-19 or FDN. Each context region of object is resized to the same size, i.e., 64×64 for elegance while the spatial resolution of extracted features is also transformed to the same size, i.e., 64×64. The results are shown in Fig. 4. The first column is the input image. The second column is taking the max operation along the channel direction of the mimic output features extracted by FDN. The third column is taking the max operation along the channel direction of the intermediate *conv4-4* output features extracted by VGG-19. The forth and the fifth are adapting the *sum* operation along the channel direction from FDN and VGG-19 network, respectively. As shown in Fig. 4, we find that there are similar regions between the output of the hidden layers of the FDN and the corresponding layers of the VGG-19, especially these brighter areas.

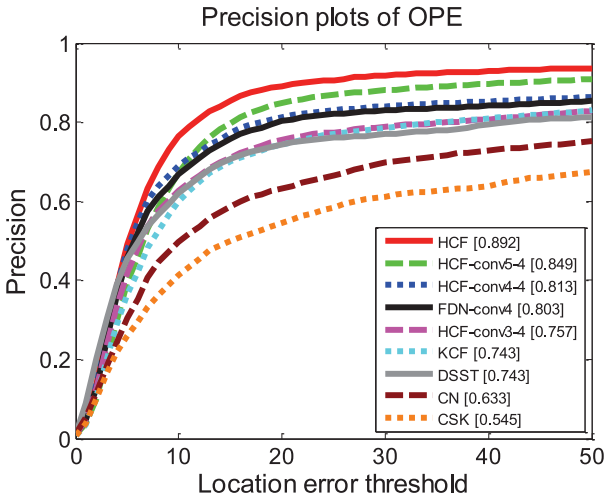


Fig. 5. Comparison of different features in the DCF formulation. Their performance differences mainly come from the used features. HCF [8]: use *conv5-4*, *conv4-4*, and *conv3-4* deep features of VGG-19; HCF-conv5-4: only use *conv5-4* deep feature of VGG-19; HCF-conv4-4: only use *conv4-4* deep feature of VGG-19; HCF-conv3-4: only use *conv3-4* deep feature of VGG-19; FDN-conv4: only use *conv4* feature extracted from our pretrained FDN. KCF and DSST: HOG feature; CN: color naming feature; and CSK: gray feature. Our *conv4* feature of FDN achieves the comparable performance with HCF-conv4-4 via VGG-19 network but much faster running speed.

TABLE I
COMPARISON OF STANDARD FDN (224 × 224 AND WITH-POOL) AND FASTER FDN (112 × 112 AND NO-POOL). WP: WITH-POOL; NP: NO-POOL. NO-POOL MEANS TO USE SHIFT-AND-STITCH METHOD IN SECTION III-C. THE RESULTS IN THE BRACKET (⋆) DENOTE THE CORRESPONDING OUTPUT OF THE TRACKERS WITH SCALE ESTIMATION AND OCCLUSION HANDLING. EVALUATION PROTOCOL: DP RATE (%)

Input	With-pool	No-pool	FPS	Speed-up
224x224	80.2 (79.7)	80.0 (79.8)	12 (WP)	1
112x112	79.4 (83.2)	80.4 (**84.6**)	43 (NP)	3×

TABLE II
COMPARISON OF STANDARD FDN (224 × 224 AND NO-POOL) AND FASTER FDN (112 × 112 AND NO-POOL) FOR DIFFERENT SEQUENCES. STAN.: STANDARD FDN; FAST.: FASTER FDN; DIFF.: TRACKING PERFORMANCE DIFFERENCE BETWEEN FEATURES GENERATED BY STANDARD FDN AND FASTER FDN

FDN	CarS.	Coke	Coup.	Dudek	Iron.	Lemm.	Moto.
Stan.	75	**95.2**	**82.9**	88.7	**57.8**	27.5	10.4
Fast.	**86.5**	88	67.1	**95.9**	13.9	**53.4**	**17.1**
Diff.	11.5	-7.2	-15.8	7.2	-43.9	25.9	6.7
FDN	Shak.	Skii.	Subw.	Sylv.	Tiger1	Tiger2	Walk.
Stan.	1.9	19.8	24.6	**99**	**76**	**62.2**	**100**
Fast.	**80.8**	**100**	**100**	79.4	65	52.6	71.4
Diff.	78.9	80.2	75.4	-19.6	-11	-9.6	-28.6

D. Baseline Comparison

We evaluate the impact of the feature extracted from FDN and compare it with the traditional features in the standard DCF formulation, shown in Fig. 5. CSK [91] is based on gray feature. The extension version of CSK is KCF [19] which is based on HOG variant feature [92]. DSST [18] is also based on HOG feature [93]. CN [47] explores the color property via color naming features [94]. These traditional features (e.g., gray and HOG) achieved good progress in DCF formulation before the recent deep convolutional features are applied in the tracking domain [8], [57], [58]. As shown in Fig. 5, the *conv4* output feature extracted from FDN or the *conv4-4* output feature extracted from VGG-Net both achieve better performance in precision plots than HOG in the KCF formulation due to the richer information encoded in the deep convolutional feature. Notice that the performance of the *conv4* feature of FDN and the *conv4-4* feature is similar. In Fig. 5, there is an obvious gap between the black curve of FDN-conv4 and the turquoise curve of KCF in precision plots. Even while the threshold is 10, the precision of FDN-conv4 is 0.667 while KCF is 0.599. With scale estimation and occlusion handling, feature distilled tracking (FDT) can achieve 0.846 while DSST is 0.743 in Fig. 6. It verifies the good generalization ability of our proposed FDN. Therefore, we consider that the distilled features is much better than HOG features.

E. Comparison of Standard FDN and Faster FDN

We evaluate the effectiveness of our faster FDN architecture as well. The results are shown in Table I. The evaluation protocol is following the metric of distance precision (DP) rate (%) center location error (CLE) used in [23]. Here, DP rate (%) is the relative number of frames in the sequence, where the CLE of the target and the ground truth is smaller than a certain threshold (e.g., 20 pixels). Note that we only use the feature maps of *conv4* as features. In addition, we also test the scale estimation and occlusion handling techniques as described in Section IV-C. Table I shows that our faster FDN is about three times faster than standard FDN while its performance is better than the standard FDNs, especially after adding the scale estimation module and occlusion handling strategy. We argue that, for the task of tracking, scaling the general small target window image up to the size of 224×224 may affect the effective of the learned filters. Because the filters are optimized for images of appropriate scales. So we use the faster FDN architecture in the following experiments. For deeper understanding why scale estimation and occlusion handling produces bigger performance gain on faster FDN features than on standard FDN features, we compare the results among all sequences in the tracking benchmark OTB-50 and give the values of several sequences which have significant differences based on the metric of the DP. As shown in Table II, we find that in some sequences the performance of features extracted from standard FDN is better than the faster FDN's, (e.g., *Coke, Couple, Ironman, Sylvester, Tiger1, Tiger2,* and *Walking2*), while the faster FDN's is better in some other sequences (e.g., *CarScale, Dudek, Lemming, MotorRolling, Shaking, Skiing,* and *Subway*). Although there are better performance in different sequences for features with standard FDN and faster FDN as shown in Tables I and II, faster FDN achieves better overall performance than standard FDN in the tracking benchmark OTB-50. Object tracking is an accurate target state estimation problem, where a minimal error or deviation may result in wide divergence, especially in the discrepant deep features extracted from different networks.

AQ2

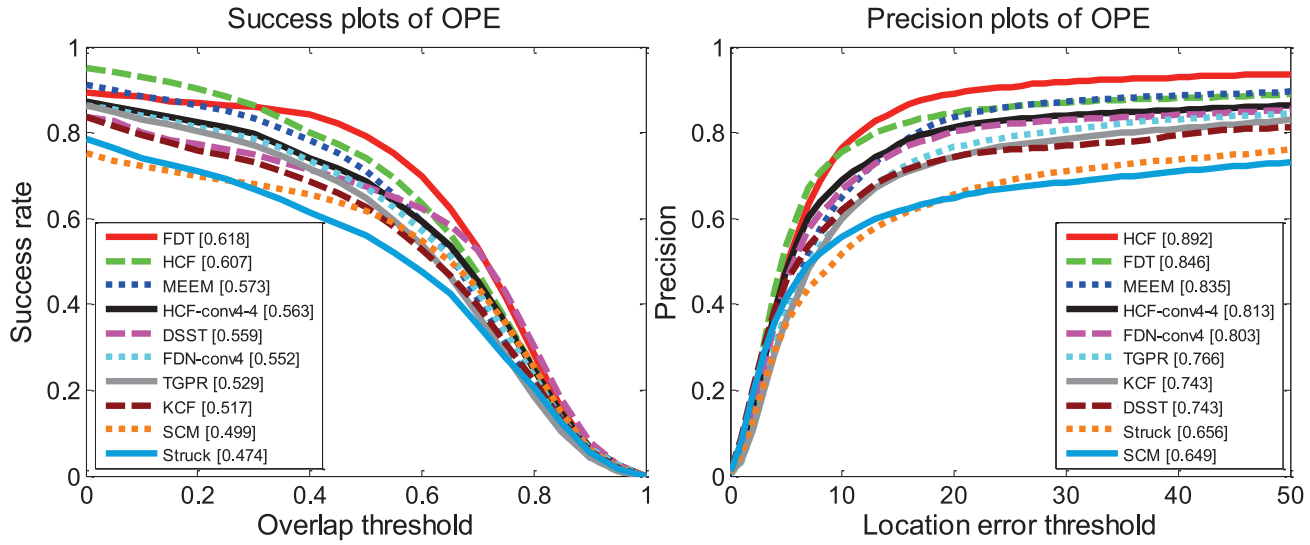


Fig. 6. Precision and success plots of total performance evaluation for the 50 videos in the data set [23] (best-viewed in high-resolution). The mean precision scores for each tracker are reported in the legends. The best method with the highest performance is marked as *red*. In both of the two metrics based on CLE and overlap rate, our method (FDT and FDT-conv4) perform favorably better than the state-of-the-art tracking approaches.

TABLE III
OVERALL TRACKING PERFORMANCE AT OTB-50 BENCHMARK WITH DIFFERENT FDN ARCHITECTURES. DP: DISTANCE PRECISION RATE; OS: MEAN OVERLAP SUCCESS RATE WITH ALL SEQUENCES IN THE BENCHMARK

Network	DP(%)	OS(%)	FPS
FDN	80.4	71.8	7.81
FDN-Double	81.3	74.2	3.02
FDN-Half	77.3	69.5	9.38
FDN-Shallower	71.3	65.4	8.53
FDN-Deeper	79.9	71.9	5.31

TABLE IV
VARIATIONS OF FDN NETWORK ARCHITECTURES IN EXPERIMENTS

	FDN	FDN-Double	FDN-Half	FDN-Shallower	FDN-Deeper
conv1 pool1	7x7x64 2x2	7x7x**128** 2x2	7x7x**32** 2x2	7x7x64 **4x4**	7x7x64 2x2
conv2 pool2	5x5x128 2x2	5x5x**256** 2x2	5x5x**64** 2x2	5x5x128 2x2	5x5x128 2x2
conv3 pool3	3x3x256 2x2	3x3x**512** 2x2	3x3x**128** 2x2	3x3x512	3x3x256
conv4 pool4	3x3x512	3x3x512	3x3x512		3x3x256 2x2
conv5					3x3x512

F. Exploring Different Feature Distilled Network Architectures

To evaluate the effectiveness of the network structure on the tracking performance, we investigate four types of neural network architectures in the views of filter variation (i.e., FDN-Double and FDN-Half) and layer variation (i.e., FDN-Shallower and FDN-Deeper), respectively. The architectures are shown in Table IV. In the view of filter number, FDN-Double has twice filters comparing with FDN while FDN-Half has half filters of FDN except the last layer which keeps the same filter number for preserving the output features consistent. In the number of layer, FDN-Shallower has removed one convolutional layer and one activation layer than FDN while FDN-Deeper has added one convolutional layer and one activation layer than FDN. From Table III, we find that although FDN-Double has better performance than FDN, the running speed of FDN-Double has reduced much more. Compared with FDN, FDN-Half has a three percentage loss in performance than FDN. FDN-Shallower has much worse performance than FDN. As shown in Table III, we find that FDN is a more appropriate choice as our network architecture considering of the tracking performance and the running speed.

G. Time Complexity

The time complexity of a CNN is related to the input image size and number of filters. Note that the input image size and network structure for FDN is much smaller than the VGG-19 teacher network. The proposed method has at most five layers according to the adopted feature extracted layer, whereas the VGG-19 implements at most 16 convolutional layers for extracting feature in *conv5-4* layer. Meanwhile, we modified the pretrained network so that the input image size changes from 224×224 to 112×112 without performance loss in the tracking domain. Hence, FDN has much lower computational cost. The VGG-19 based HCF requires 250 ms to process an image on an Intel Core i7 CPU, while FDN only takes 40 ms, which is six times faster. The FDN costs only 8 ms on the Titan GPU. As shown in Table V, our FDN tracker based on *conv4* can run at around 42 FPS in GPU mode which is 5.19 faster than HCF tracker based on *conv4-4* feature. The traditional features in DCF have much higher speed than deep features-based trackers, including FDN features. Our motivation is to speed up deep trackers while preserving their high performance. Therefore, we only give the comparisons between *conv4* features of FDN and different VGG-19 intermediate features. In Table V, we also have compared our method with HCF-conv4-4 in CPU-mode, shown in the last two rows.

TABLE V
COMPARISON BETWEEN DIFFERENT NETWORKS ON OTB-50. REPORTED SPEEDS INCLUDE FEATURE COMPUTATION, PREDICTION TIME, AND MODEL UPDATE TIME

Algo.	Feature	DP(%)	OP(%)	FPS	Speed-up
HCF	*conv4-4*	81.2	66.6	8.18	1 (GPU)
Ours	*conv4*	80.4	65.7	**42.49**	**5.19**×(GPU)
HCF	*conv3-4*	75.5	60.7	12.22	1.49×(GPU)
HCF	*conv5-4*	**84.9**	**66.6**	7.08	0.87×(GPU)
HCF	*conv4-4*	81.2	66.6	1.47	1(CPU)
Ours	*conv4*	80.4	65.7	**7.81**	**5.31**×(CPU)

H. OTB-50

We compare our method with nine different state-of-the-art trackers. The trackers used for comparison are: HCF [8], MEEM [41], DSST [18], TGPR [39], KCF [19], ALSA [95], Struck [35], SCM [26], and TLD [37]. Their source codes or binary codes are released by the authors publicly and the default parameters are used suggested by their papers. All algorithms are evaluated in terms of the same initial positions in the first frame in [23]. For handling scale variation, we introduced scale estimation and simple occlusion handling strategies of Section IV-C into the proposed FDN tracker, denoted as FDT. In [8], there are comprehensive comparisons showing that *conv3-4*, *conv4-4*, and *conv5-4* features of VGG-19 are better than some other deep features (e.g., AlexNet features [84]) in the DCF framework. Therefore, we only give the comparison results of VGG-19 features.

Fig. 6 shows precision and success plots which contain the mean distance and OP over all the 50 sequences. The values in the legend are the mean precision score and AUC, respectively. As shown in Fig. 6, our tracker FDT is better than the other trackers in success plots and achieve a comparable performance with the others in precision plots.

1) Attribute-Based Evaluation: Many factors can affect the tracking performance. In the recent benchmark evaluation [23], the sequences are annotated with 11 different attributes, which are named as: occlusion, deformation, illumination variation, fast motion, motion blur, out-of-plane rotation, scale variation, background clutter, out-of-view, low resolution, and in-plane rotation. Fig. 7 shows the success plots of several different attributes.

As shown in Fig. 7, FDT obtained the top performance compared to the state-of-the-art trackers in illumination variation, scale variation, out-of-plane rotation, and in-plane rotation. This is mainly because the *conv4* deep feature extracted from our proposed FDN has good generalization ability. The two main differences between FDN-conv4 to FDT are the scale estimation and occlusion handling. In most of the attribute cases, the performance of FDT is better than FDN-conv4. The increase in tracking performance not only attributes to scale estimation, but also depends on the effective feature representation which has strong coupling relationship with the scale variation of the target and occlusion handling. Therefore, it is helpful to add the scale estimation module and occlusion handling for boosting the tracking performance.

2) Parameter Analysis: The goal of the parameter $\mathcal{T}$ in (9) is to account for both the structure factors and appearance variations of the target object, as well as retraining hard negatives (distracters) in the background. In order to evaluate the effect of $\mathcal{T}$, we perform the experiments to compare different update thresholds. As shown in Fig. 8, we can see that the update threshold affects the performance of our trackers heavily because it determines the learning rate and the tradeoff between adaptivity and stability of the tracker.

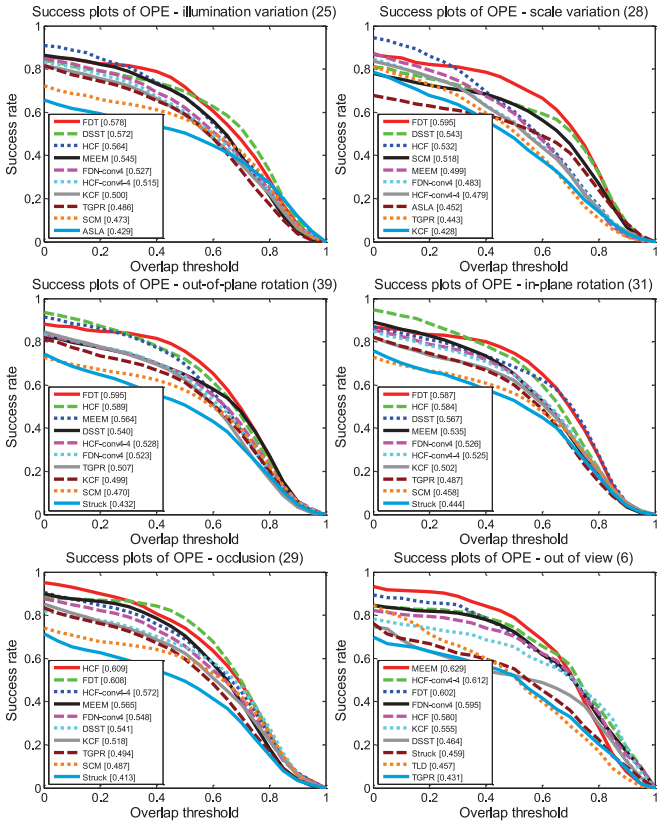


Fig. 7. Success plots of different attributes (best-viewed on high-resolution display). The valued appearing in the title denotes the number of videos associated with the respective attributes. The proposed methods in this paper perform favorably against state-of-the-art algorithms.

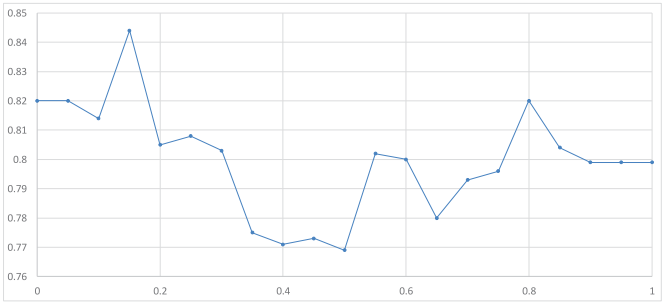


Fig. 8. Overall performance comparison with different update thresholds. The values of x-axis are different update threshold for the parameter $\mathcal{T}$. The values of y-axis represent the accuracy of DP.

I. OTB-100

We also perform experiments on the OTB-100 dataset [88] containing 100 videos. Our approach is compared with the six state-of-the-art trackers. The quantitative comparisons of DP

TABLE VI
COMPARISONS WITH STATE-OF-THE-ART TRACKERS ON OTB-50 AND OTB-100 SEQUENCES. DP: DISTANCE PRECISION RATE; OS: MEAN OVERLAP SUCCESS RATE; CLE: CENTER LOCATION ERROR. TO KEEP THINGS FAIR, THESE TRACKERS ARE ALL RUNNING IN THE CPU-MODE. FPS: FRAMES PER SECOND. THE FIRST AND SECOND BEST VALUES ARE HIGHLIGHTED BY BOLD AND UNDERLINE

	Ours	HCF	KCF	Struck	SCM	Meem	TGPR
DP(50)	84.6	**89.1**	74.1	65.6	64.9	83.0	70.5
DP(100)	80.6	**83.7**	69.2	63.5	57.2	78.1	64.3
OS(50)	**76.8**	74.0	62.2	55.9	61.6	69.6	62.8
OS(100)	**68.8**	65.5	54.8	51.6	51.2	62.2	53.5
CLE(50)	20.0	**15.7**	35.5	50.6	54.1	20.9	51.3
CLE(100)	27.7	**22.8**	45.0	47.1	61.6	27.7	55.5
FPS	7.8	1.5	**250**	12	0.4	22	0.7

rate at 20 pixels, overlap success rate at 0.5, CLEs in Table VI. Among the existing methods, our tracker achieves the top-1 or top-2 performance. Moreover, our tracker is around five times faster than HCF analysis in Section V-G. According to our empirical experiments, their speeds in CPU-mode are as shown in Table VI.

AQ3

VI. CONCLUSION

In this paper, we propose a feature distilled network (FDN) for visual tracking. FDN not only preserves the fine structure of the teacher network in feature space with less computational burden but also obtains the convolutional features for accurate and robust tracking. Moreover, the features are embedded in a scale adaptive DCF formulation for robust and fast tracking. Extensive experiments show that our tracker obtains a comparable performance in accuracy while it is faster than the state-of-the-art deep methods on the challenging tracking benchmark datasets.

REFERENCES

[1] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 3, pp. 334–352, Aug. 2004.

[2] C. Zhang, J. Cheng, and Q. Tian, "Structured weak semantic space construction for visual categorization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2017.2728060.

[3] C. Zhang, J. Cheng, C. Li, and Q. Tian, "Image-specific classification with local and global discriminations," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2017.2748952.

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

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 580–587.

[6] C. Zhang, J. Cheng, and Q. Tian, "Incremental codebook adaptation for visual representation and categorization," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2726079.

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

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

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.

[10] D. Erhan *et al.*, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Jan. 2010.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[12] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. ACM SIGKDD Conf. Knowl. Disc. Data Min.*, Philadelphia, PA, USA, 2006, pp. 535–541.

[13] J. J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 2654–2662.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[15] A. Romero *et al.*, "FitNets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2014.

[16] V. Vapnik and R. Izmailov, "Learning using privileged information: Similarity control and knowledge transfer," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 2023–2049, 2015.

[17] 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. (CVPR)*, San Francisco, CA, USA, 2010, pp. 2544–2550.

[18] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, Nottingham, U.K., Sep. 2014, pp. 1–11, doi: 10.5244/C.28.65.

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

[20] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surveys*, vol. 38, no. 4, p. 13, 2006.

[21] X. Li *et al.*, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, p. 58, 2013.

[22] Y. Pang and H. Ling, "Finding the best from the second bests—Inhibiting subjective bias in evaluation of visual tracking algorithms," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Sydney, NSW, Australia, 2013, pp. 2784–2791.

[23] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Portland, OR, USA, 2013, pp. 2411–2418.

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

[25] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Minimum error bounded efficient l1 tracker with occlusion detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Colorado Springs, CO, USA, 2011, pp. 1257–1264.

[26] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, 2012, pp. 1838–1845.

[27] N. Wang, J. Wang, and D.-Y. Yeung, "Online robust non-negative dictionary learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, 2013, pp. 657–664.

[28] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, "Robust object tracking with online multi-lifespan dictionary learning," in *Proc. Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, 2013, pp. 665–672.

[29] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via structured multi-task sparse learning," *Int. J. Comput. Vis.*, vol. 101, no. 2, pp. 367–383, 2013.

[30] L. Ma *et al.*, "Local subspace collaborative tracking," in *Proc. Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 4301–4309.

[31] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja, "Robust visual tracking via exclusive context modeling," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 51–63, Jan. 2016.

[32] N. Widynski, S. Dubuisso, and I. Bloch, "Integration of fuzzy spatial information in tracking based on particle filtering," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 3, pp. 635–649, Jun. 2011.

[33] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust visual tracking via convolutional networks without training," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1779–1792, Apr. 2016.

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

[35] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Barcelona, Spain, 2011, pp. 263–270.

[36] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[37] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.

[38] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Portland, OR, USA, 2013, pp. 1838–1845.

[39] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Zürich, Switzerland, 2014, pp. 188–203.

[40] J. S. Supancic and D. Ramanan, "Self-paced learning for long-term tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, 2013, pp. 2379–2386.

[41] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 188–203.

[42] K. Zhang, L. Zhang, and M.-H. Yang, "Fast compressive tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2002–2015, Oct. 2014.

[43] H. Song, "Robust visual tracking via online informative feature selection," *Electron. Lett.*, vol. 50, no. 25, pp. 1931–1933, Dec. 2014.

[44] D. Wang, H. Lu, and C. Bo, "Visual tracking via weighted local cosine similarity," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1838–1850, Sep. 2015.

[45] G. Zhu, J. Wang, C. Zhao, and H. Lu, "Weighted part context learning for visual tracking," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5140–5151, Dec. 2015.

[46] Q. Liu, J. Yang, K. Zhang, and Y. Wu, "Adaptive compressive tracking via online vector boosting feature selection," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4289–4301, Dec. 2017.

[47] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 1090–1097.

[48] G. Zhu, J. Wang, Y. Wu, and H. Lu, "Collaborative correlation tracking," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2015, pp. 184.1–184.12.

[49] G. Zhu, J. Wang, and H. Lu, "Clustering based ensemble correlation tracking," *Comput. Vis. Image Understand.*, vol. 153, pp. 55–63, Dec. 2016.

[50] G. Zhu, J. Wang, Y. Wu, X. Zhang, H. Lu, "Mc-HOG correlation tracking with saliency proposal," in *Proc. 30th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 3690–3696.

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

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

[53] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 391–405.

[54] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 1430–1438.

[55] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proc. Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 3101–3109.

[56] H. Li, Y. Li, and F. Porikli, "DeepTrack: Learning discriminative feature representations by convolutional neural networks for visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, doi: 10.5244/C.28.56.

[57] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 597–606.

[58] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 3119–3127.

[59] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4293–4302.

[60] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[61] H. Van Nguyen, K. Zhou, and R. Vemulapalli, "Cross-domain synthesis of medical images using efficient location-sensitive deep network," in *Proc. Int. Conf. Med. Image Comput. Comput. Assisted Intervent.*, 2015, pp. 677–684.

[62] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[63] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 1269–1277.

[64] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Convolutional neural networks with low rank expansions," *Proc. Brit. Mach. Vis. Conf.*, 2014, doi: 10.5244/C.28.88.

[65] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Proc. NIPS*, vol. 2. 1989, pp. 598–605.

[66] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 1135–1143.

[67] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[68] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. 43rd Int. Symp. Comput. Archit.*, Seoul, South Korea, 2016, pp. 243–254.

[69] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014.

[70] S. Arora, A. Bhaskara, R. Ge, and T. Ma, "Provable bounds for learning some deep representations," in *Proc. ICML*, Beijing, China, 2014, pp. 584–592.

[71] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learn. Unsupervised Feature Learn. NIPS Workshop*, vol. 1, 2011, p. 4.

[72] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights $+1$, 0, and -1," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Belfast, U.K., 2014, pp. 1–6.

[73] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," *CoRR*, vol. abs/1510.03009, 2015.

[74] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brisbane, QLD, Australia, 2015, pp. 1131–1135.

[75] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *CoRR*, vol. abs/1412.7024, 2014.

[76] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 963–971.

[77] S. K. Esser, R. Appuswamy, P. A. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 1117–1125.

[78] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[79] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[80] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. ICML*, Lille, France, 2015, pp. 2285–2294.

[81] 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.

[82] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 2818–2826.

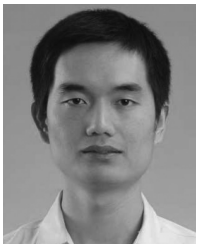
[83] F. N. Iandola *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size," *CoRR*, vol. abs/1602.07360, 2016.

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

[85] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[86] P. Sermanet *et al.*, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.

[87] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation, in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 3431–3440.
[88] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.
[89] A. Vedaldi and K. Lenc, "MatconvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 689–692.
[90] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 82–90.
[91] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, 2012, pp. 702–715.
[92] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[93] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[94] J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning color names for real-world applications," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1512–1523, Jul. 2009.
[95] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, 2012, pp. 1822–1829.



Guibo Zhu received the B.E. degree from Wuhan University, Wuhan, China, in 2009, the M.Sc. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2013, and the Ph.D. degree from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, in 2016.

He is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include computer vision, pattern recognition, machine learning, and brain-inspired learning.

Jinqiao Wang (M'09) received the B.E. degree from the Hebei University of Technology, Tianjin, China, in 2001, the M.S. degree from Tianjin University, Tianjin, in 2004, and the Ph.D. degree in pattern recognition and intelligence systems from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China, in 2008.

He is currently a Professor with the Chinese Academy of Sciences. His current research interests include pattern recognition and machine learning, image and video processing, mobile multimedia, and intelligent video surveillance.

Peisong Wang received the B.E. degree from Shandong University, Jinan, China, in 2013. He is currently pursuing the Ph.D. degree with the Image and Video Analysis Group, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China.

His current research interests include machine learning and computer vision.

Yi Wu received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009.

He is a Runze Professor with Nanjing Audit University, Nanjing, China. From 2010 to 2012, he was a Post-Doctoral Fellow with Temple University, Philadelphia, PA, USA. From 2012 to 2014, he was a Post-Doctoral Fellow with the University of California at Merced, USA. His current research interests include computer vision, medical image analysis, and machine learning.

Hanqing Lu (SM'06) received the B.E. and M.E. degrees from the Harbin Institute of Technology, Harbin, China, in 1982 and 1985, respectively, and the Ph.D. degree from the Huazhong University of Sciences and Technology, Wuhan, China, in 1992.

He is currently a Professor of the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image and video analysis, medical image processing, and object recognition.