AN ENSEMBLE OF DEEP NEURAL NETWORKS FOR OBJECT TRACKING

Xiangzeng Zhou, Lei Xie, Peng Zhang and Yanning Zhang

Shaanxi Provincial Key Laboratory of Speech & Image Information Processing (SAIIP),
School of Computer Science, Northwestern Polytechnical University, Xi'an, 710072, P. R. China

ABSTRACT

Object tracking in complex backgrounds with dramatic appearance variations is a challenging problem in computer vision. We tackle this problem by a novel approach that incorporates a deep learning architecture with an on-line AdaBoost framework. Inspired by its multi-level feature learning ability, a stacked denoising autoencoder (SDAE) is used to learn multi-level feature descriptors from a set of auxiliary images. Each layer of the SDAE, representing a different feature space, is subsequently transformed to a discriminative object/background deep neural network (DNN) classifier by adding a classification layer. By an on-line AdaBoost feature selection framework, the ensemble of the DNN classifiers is then updated on-line to robustly distinguish the target from the background. Experiments on an open tracking benchmark show promising results of the proposed tracker as compared with several state-of-the-art approaches.

Index Terms— Boosting, deep learning, visual tracking, deep neural network, AdaBoost

1. INTRODUCTION AND RELATED WORKS

Object tracking is a fundamental subject in the field of computer vision with a wide range of applications. The chief challenge of object tracking can be attributed to the difficulty in handling the appearance variability of a target object. To deal with this problem, many approaches have been proposed in recent years [1–12].

Early tracking approaches employ static appearance models which are either defined manually or trained initially in the first frame [1, 2]. Unfortunately, these approaches may face difficulties when the tracked targets have heavy intrinsic appearance changes. It has been verified that an adaptive appearance model, evolving with the change of object appearance, brings better performance [3–8]. Many of these approaches adopted a generative model for object representation. In nature, generative approaches were not equipped to distinguish a target from the background [5, 8].

Modeling both the object and the background via discriminative classifiers is an alternative choice in the design of appearance models [7, 9–11]. In this spirit, object tracking has been formulated as a binary classification problem which distinguishes the object of interest from the background via a classifier, e.g., support vector machines (SVMs) [12, 13]. Specifically, the idea of boosting was adopted to select discriminative features for robust tracking [7, 9]. Boosting is a general method for improving the performance of a learning algorithm by combining several "weak" learning algorithms [14]. In ensemble tracking (ET) [9], an ensemble of weak classifiers was trained on-line to distinguish between the target and the background and a strong classifier was successfully obtained using AdaBoost [15]. Extended from [16], Grabner *et al.* [7] have proposed an on-line AdaBoost feature selection algorithm to address a real-time tracking task by updating an ensemble of weak classifiers while tracking. This approach was able to cope with appearance changes of the target. Each weak hypothesis was generated according to a different type of hand-crafted feature.

Hand-crafted feature descriptors may lead to unrecoverable information loss and which feature is suitable for tracking in different scenarios is still an unsolved problem. Recently, automatic feature extraction using deep learning techniques brings significant performance improvements in many areas. As a typical deep learning architecture, deep neural networks (DNNs) have shown superior performance in tasks such as speech recognition [17] and natural language processing [18]. The deep architecture performs as an automatic feature extractor [19], and each network layer can be considered as a different feature space of the raw data. In more recent years, deep learning techniques have been successfully applied in several computer vision tasks, such as image classification [20], object detection [21, 22] and object tracking [23]. Recently, Wang *et al.* [23] proposed a novel deep learning tracker (DLT) for robust visual tracking. DLT combines the philosophies behind both generative and discriminative trackers by automatically learning a deep compact image representation using a stacked denoising autoencoder (SDAE) [24]. Promising results have been reported by this deep learning approach.

Inspired by the success of on-line AdaBoost and deep learning, in this paper, we propose a novel object tracking approach which combines a family of deep neural network classifiers using on-line AdaBoost. Firstly, we build an off-line unsupervised generic image feature extractor using an SDAE with a considerable amount of natural images. Each layer of the SDAE serves as a different level of feature space, which

is subsequently transformed to a discriminative DNN classifier by adding an additional classification layer. Secondly, we propose an on-line AdaBoost feature selection framework to combine these layered classifiers for on-line updating. Particle filter is used for tracking and the confidence of each particle is provided by the boosting system. Experimental results on an open tracking benchmark [25] show that our tracker is superior to several state-of-the-art trackers and significant performance gain is achieved as compared with the recent DLT tracker [23].

In the next section, we describe the SDAE-based unsupervised image feature descriptor. In Section 3, we present the proposed approach for on-line boosting a family of deep neural networks. Experiments on an open benchmark are shown in Section 4. Finally, we conclude the paper in Section 5.

2. GENERIC IMAGE FEATURE DESCRIPTOR

In this section, we describe the unsupervised generic feature learning scheme, which is based on a stacked denoising autoencoder (SDAE). In an idiomatic manner of a learning deep architecture, we first pretrain the SDAE layer-by-layer and fine-tune the whole SDAE subsequently. This process is carried out off-line using one million images randomly selected from the 80-million tiny images dataset [26].

An SDAE is built by stacking several one-layer neural networks, called denoising autoencoders (DAE), which is trained to denoise the corrupted versions of the raw data [24]. Figure 1(A) shows the architecture of a DAE. Consider there are N_t training samples. Let $\mathbf{x}^i$ denote the original data sample and $\tilde{\mathbf{x}}^i$ be the corrupted version of $\mathbf{x}^i$. For the network, let $\mathbf{W}$ and $\mathbf{W}'$ denote the weights of the encoder and decoder, respectively. Similarly, $\mathbf{b}$ and $\mathbf{b}'$ refer to the bias terms. A denoising autoencoder is learned by solving the following optimization problem with regularized term

$$\min_{\Theta} \sum_{i=1}^{N_t} \|\mathbf{x}^i - \hat{\mathbf{x}}^i\|_2^2 + \eta(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2), \tag{1}$$

where

$$\hat{\mathbf{x}}^i = f(\mathbf{W}'\mathbf{h}^i + \mathbf{b}') \tag{2}$$

$$\mathbf{h}^i = f(\mathbf{W}\tilde{\mathbf{x}}^i + \mathbf{b}) \tag{3}$$

Here $\Theta = (\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}')$ denote the parameters of the network. $f(\cdot)$ is a nonlinear activation function which is typically the logistic sigmoid function. The factor η balances the reconstruction loss and the weight penalty terms, in which $\|\cdot\|_F$ is the Frobenius norm [27]. It has been shown that, by preventing the autoencoder from simply learning the identity mapping, a denoising autoencoder is more effective than the conventional autoencoder in discovering robust features [24]. After the pretraining step, the SDAE is unrolled to form a feedforward neural network, which is fine-tuned using the backpropagation (BP) procedure.

According to the configuration suggested in [23], we use a 4-layer network shown in Fig. 1(B) to achieve our learning. In the proposed tracker, an object in each frame is represented by a 32×32 image patch. Hence, the size of the input layer of SDAE is 1024 corresponding to a vector of 1024 dimensions. To elaborately describe the image structure, a structure with overcomplete filters is deliberatively chosen in the first hidden layer for the purpose of finding an overcomplete basis. Then, the number of units is reduced by half when a new layer is added until there are only 256 hidden units, which serves as the bottleneck of the autoencoder.

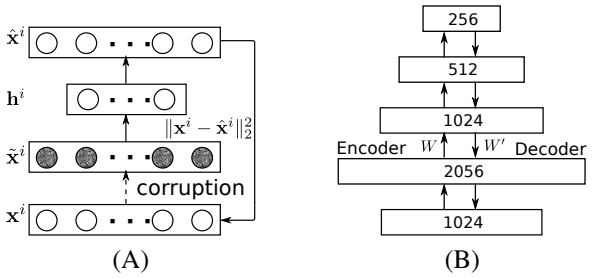

Fig. 1. (A) DAE architecture (B) SDAE architecture

3. TRACKING

In this section, we describe our on-line boosting DNN tracker. The main idea is to formulate the tracking problem as an on-line classification task. In Section 2, we have learned an image feature descriptor using an SDAE. Each layer of the SDAE serves as a different feature space of the raw image data. We transform each layer to a discriminative binary classifier by adding an additional sigmoid classification layer on the top of each layer. In the first frame, the object has been provided by a bounding box and the object region is considered as a positive sample, and the negative samples can be extracted from the surrounding background. These positive and negative samples are used to adapt the DNN classifiers via the BP algorithm in several iterations.

After that, an on-line boosting feature selection mechanism is used to combine the well-fitted binary classifiers. The tracking is carried out based on a particle filter. In each frame, a set of particles, corresponding to the object candidates, are drawn from an importance distribution. A confidence map of each image region of an object candidate is evaluated by the on-line boosting DNNs system. The region with the maximal confidence is then taken as the new location of the object. Once the object has been detected in current frame, the on-line boosting system is updated to the possible appearance variation. Similarly, the current object region is regarded as a positive sample and the surrounding regions as negative samples. The whole tracking and updating procedure is repeated once a new frame arrives.

3.1. Particle Filter based Tracking

Our tracking process is formulated within a Bayesian framework in which a particle filter [28] is used. Let $\mathbf{x}_t$ and $\mathbf{y}_t$ denote the hidden state and observation variables at time t, respectively. The state here is represented by the affine transformation parameters which correspond to translation, scale, aspect ratio, rotation, and skewness. Up to time $t-1$, the

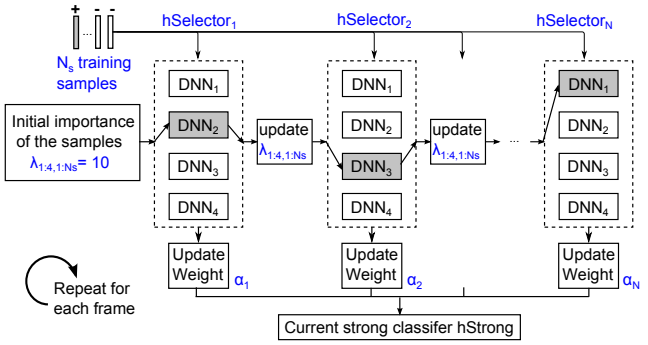

Fig. 2. On-line boosting DNNs.

predicting posterior distribution of $\mathbf{x}_t$ given all available observations $\mathbf{y}_{1:t-1} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{t-1}\}$ is computed as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}. \quad (4)$$

Inference to time t, the observation $\mathbf{y}_t$ is available and the posterior distribution of $\mathbf{x}_t$ is updated using the Bayesian formulation

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}. \quad (5)$$

The job of particle filter is to approximate the posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ by a finite set of N samples $\{\mathbf{x}_t^i\}_{i=1,...,N}$, called *particles*, associated with normalized importance weights w_t^i. The particles $\mathbf{x}_t^i$ are drawn from an importance distribution $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$ and the corresponding weights are updated using

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})}. \quad (6)$$

The importance distribution $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$ is often simplified with first-order Markov assumption to $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, and consequently the weights updating becomes $w_t^i = w_{t-1}^i p(\mathbf{y}_t|\mathbf{x}_t^i)$. In our approach, the motion model $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is modeled independently by a normal distribution and the observation model $p(\mathbf{y}_t|\mathbf{x}_t^i)$ is estimated by an on-line boosting classifier whose output is a kind of confidence measure.

3.2. On-line Boosting DNNs

Given a learned image feature descriptor in Section 2, each layer of the SDAE serves as a different level of feature. We build several discriminative binary DNN classifiers by adding a sigmoid layer on the top of each layer of the SDAE, and combine these classifiers using the on-line boosting framework [7]. Specifically, we present a modified version of on-line AdaBoosting to be more applicable to the DNN classifiers. The modifications include voting weight updating, sample importance weight updating and DNN classifiers updating with weighted samples, as described in Algorithm 1. For the requirement of on-line DNN updating, we carry out the boosting learning process in a batch manner.

The framework of the proposed on-line boosting DNN framework is shown in Fig. 2, and the corresponding algorithm pseudo-code is presented in Algorithm 1. Let $\mathcal{X}_t = \{\mathbf{x}_t^1, ..., \mathbf{x}_t^{N_s}\}$ denote the N_s training samples associated with labels $\mathcal{Y}_t = \{0, 1\}$ at frame t. At the first frame, following the concept of *selector* in [7], a fixed set of N selectors $\{h_1^{sel}, ..., h_N^{sel}\}$ is initialized randomly. Once a set of training samples $\langle \mathcal{X}_t, \mathcal{Y}_t \rangle$ arrive at frame t, selectors are updated with respect to the importance weights λ of current samples. In order to update the DNN classifiers with weighted samples, the feedforward error is weighted by multiplying with λ before the BP process. This is done for all samples over each DNN classifier in a batch manner. After that, the error $e_{n,m,i}$ is estimated from the weights of correctly and wrongly classified examples ($\lambda_{n,m,i}^{corr}$ and $\lambda_{n,m,i}^{wrong}$), where $e_{n,m,i}$ is the error of the i-th sample over the m-th DNN classifier in the n-th selector. Then, the selector h_n^{sel} selects one DNN classifier with the maximal error reducing margin in average. Subsequently, the voting weight α_n and the importance weight λ are updated, and then the algorithm moves on to the next selector. This process is repeated for all selectors. Finally, a final strong classifier h^{strong} is obtained by a linear combination of selectors to serve as a confidence measure for the particle filter.

Algorithm 1 On-line Boosting DNNs

Init: $\lambda_{n,m,i}^{corr} = 1, \lambda_{n,m,i}^{wrong} = 1, \lambda_{n,m,i} = 10, \hat{e}_{0,m} = 0.1$
for $n = 1, 2, ..., N$ **do**
  for $m = 1, 2, ..., M$ **do**
    $\mathrm{DNN}_{n,m}$ = update($\mathrm{DNN}_{n,m}, \langle \mathcal{X}_t, \mathcal{Y}_t \rangle, \lambda$)
    $e_{n,m,i} = |\mathrm{DNN}_{n,m}(\mathbf{x}_t^i) - \mathbf{y}_t^i|$ $(e_{n,m,i} \in [0, 1])$
    if $e_{n,m,i} < 0.5$ **then**
      $\lambda_{n,m,i}^{corr} = \lambda_{n,m,i}^{corr} + \lambda_{n,m,i} \times e_{n,m,i}$
      $\lambda_{n,m,i}^{wrong} = \lambda_{n,m,i}^{wrong} + \lambda_{n,m,i} \times (1 - e_{n,m,i})$
    else
      $\lambda_{n,m,i}^{wrong} = \lambda_{n,m,i}^{wrong} + \lambda_{n,m,i} \times e_{n,m,i}$
      $\lambda_{n,m,i}^{corr} = \lambda_{n,m,i}^{corr} + \lambda_{n,m,i} \times (1 - e_{n,m,i})$
    end if
    $e_{n,m,i} = \frac{\lambda_{n,m,i}^{wrong}}{\lambda_{n,m,i}^{wrong} + \lambda_{n,m,i}^{corr}}$
  end for
  $\hat{e}_{n,m} = \frac{1}{N_s} \sum_{i=1}^{N_s} e_{n,m,i}$
  $m^* = \arg\max_m (\hat{e}_{n-1,m} - \hat{e}_{n,m})$
  $h_n^{sel} = \mathrm{DNN}_{m^*}$
  $\alpha = \exp\left(\frac{\hat{e}_{n-1,m^*} - \hat{e}_{n,m^*}}{\hat{e}_{n-1,m^*}}\right)$
  $\lambda_{n,m,i} = \lambda_{n,m,i} \times e_{n,m,i}$
end for

4. EXPERIMENTS

4.1. Experimental Settings

We compare our proposed approach with some state-of-art trackers over 8 video sequences on a standard benchmark [25]. For off-line training of the SDAE, we follow the parameter configuration of the recent DLT tracker [23]. The momentum parameter is set to 0.9. The mini-batch size is set to 100 for off-line training and to 10 for on-line updating. For the on-line tracking system, we draw 1000 particles for the particle filter. The number N of selectors is set to 5 and the number M of DNN classifiers is 4.

Table 1. Comparison of 7 trackers on 8 video sequences. The first number denotes the success rate in percentage and the number in brackets denotes the central-pixel error in pixels.

	Sylvester	Coke	Woman	Girl	Car4	David3	David	Bolt
CT [29]	82.8(8.6)	9.3(40.5)	17.0(115.0)	17.8(18.9)	27.5(86.0)	34.9(88.7)	42.7(10.5)	0.6(363.8)
DLT [23]	84.1(7.0)	59.8(28.4)	21.1(144.1)	95.2(3.0)	**100**(2.6)	33.3(104.8)	82.0(6.6)	2.3(388.1)
IVT [5]	67.6(34.2)	13.1(83.0)	19.6(174.1)	18.6(22.5)	**100**(**2.1**)	63.5(52.0)	79.4(**4.8**)	1.4(397.0)
L1T [30]	42.8(26.2)	20.0(50.4)	20.9(128.7)	**97.0**(**2.8**)	30.0(77.0)	45.6(90.0)	69.2(14.0)	1.1(408.4)
MIL [3]	54.6(15.2)	11.7(46.7)	20.0(127.3)	29.4(13.7)	27.6(50.8)	**68.3**(**29.7**)	22.9(16.9)	1.1(393.5)
MTT [31]	82.3(7.6)	61.5(30.0)	21.1(136.8)	93.2(4.3)	31.1(22.3)	10.3(341.3)	28.9(33.1)	1.1(408.6)
Ours	**91.9**(**6.6**)	**69.1**(**26.6**)	**86.4**(**5.5**)	89.0(3.2)	**100**(3.3)	54.8(66.0)	**84.7**(6.2)	**24.5**(**111.6**)
CT [29]	1132-OPR	22-IV	95-DEF	98-OPR	186-IV	83-OCC	406-IV	3-DEF
DLT [23]	SUCCESS	177-FM	119-OCC	SUCCESS	SUCCESS	82-OCC	SUCCESS	9-DEF
IVT [5]	911-OPR	39-OCC	110-OCC	91-OPR	SUCCESS	190-OCC	94-DEF	6-DEF
L1T [30]	597-OPR	82-OPR	118-OCC	SUCCESS	199-SV	118-OPR	SUCCESS	5-DEF
MIL [3]	717-IV	70-FM	119-OCC	98-OPR	199-SV	100-DEF	SUCCESS	5-DEF
MTT [31]	1138-OPR	190-OCC	119-OCC	SUCCESS	207-SV	27-DEF	153-OPR	5-DEF
Ours	SUCCESS	177-FM	SUCCESS	SUCCESS	SUCCESS	82-OCC	SUCCESS	119-DEF

4.2. Quantitative and Qualitative Performance

For quantitative comparison, we use two standard metrics: success rate and central-pixel error. Both the tracking result and the labeled groundtruth are represented by a bounding box, respectively. In each frame, once the overlap percentage of the two bounding boxes is larger than 50% against the entire union box, the tracking is considered as a successful case. The central-pixel error is defined as the Euclidean distance between the centers of the two bounding boxes. Table 1 summarizes the performances of the 7 trackers over 8 typical video sequences in the top half. The best results are highlighted in bold font. It's clearly observed that our proposed approach is superior to other methods on 6 video sequences. For the other 2 video sequences, our method is also among the best three methods. Figure 3 shows some tracking results.

We have analyzed the robustness of the 7 trackers against some common problems defined in [25]: illumination variation (IV), partial or full occlusion (OCC), non-rigid object deformation (DEF), fast motion (FM) and out-of-plane rotation (OPR). Each entity in the bottom half of Table 1 means that a tracker fails in which frame due to what reason. The SUCCESS means the tracker finished with tracking success, but may suffer several unsuccessful cases in the middle. It's clearly observed that our tracker and the DLT tracker win the most and second most successful tracking, respectively. In the *woman* sequence, our tracker doesn't drift whilst most trackers fail due to severe occlusions. The *Bolt* is a challenging sequence on which most trackers fail in early frames due to severe deformation. However, our tracker continues tracking till frame 119. In the *David3* sequence, our tracker fails because of full occlusion. Fast motion (FM) also causes our tracker's failure as shown in the *Coke* sequence.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel object tracker by integrating the deep learning technique with the on-line boosting framework. We first use a stacked denoising autoencoder (SDAE) to learn a multi-level image feature descriptor. A family of discriminative DNN classifiers is then built from the different layers of the SDAE. After that, these classifiers are combined with an alternative version of the on-line boosting to facilitate the object/background classification. Promising results have been achieved. We believe that the key to the success of our approach lies in two-fold. First, the deep learning architecture can automatically learn useful generic image features in different levels. Second, which level of feature is most suitable for appearance modeling is further automatically decided by boosting. In the future, we plan to use a convolutional neural network (CNN) which recently has shown better capability of extracting image features [20]. Currently, without aiding with GPU our tracker suffers a little time-consuming DNN updating, and this would be our next work to reduce computation cost with GPU. We also plan to investigate how the depth of the deep architecture affects the tracking performance.

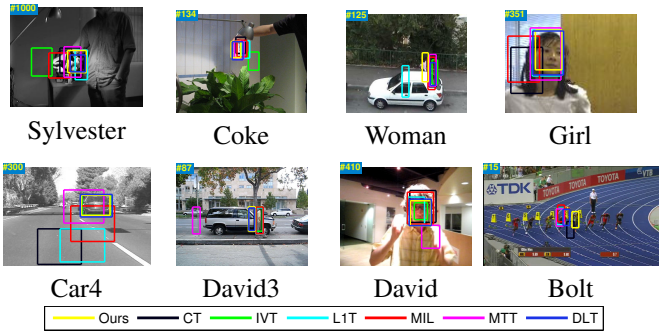

Fig. 3. Comparison of 7 trackers on several key frames in terms of the bounding box.

6. ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61175018, 61301194), the Fok Ying Tung Education Foundation (131059) and the doctoral program of High Education of China (No. 20126102120055) approved by the Ministry of Education, China.

References

[1] Amit Adam, Ehud Rivlin, and Ilan Shimshoni, "Robust fragments-based tracking using the integral histogram," in *CVPR '06*, 2006, vol. 1, pp. 798–805.

[2] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *CVPR '00*, 2000, vol. 2, pp. 142–149.

[3] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie, "Visual tracking with online multiple instance learning," in *CVPR '09*, 2009, pp. 983–990.

[4] Junseok Kwon and Kyoung Mu Lee, "Visual tracking decomposition," in *CVPR '10*, 2010, pp. 1269–1276.

[5] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang, "Incremental learning for robust visual tracking," *IJCV '08*, vol. 77, pp. 125–141, 2008.

[6] Helmut Grabner, Christian Leistner, and Horst Bischof, "Semi-supervised on-line boosting for robust tracking," in *ECCV '08*, 2008, pp. 234–247.

[7] Helmut Grabner, Michael Grabner, and Horst Bischof, "Real-time tracking via on-line boosting.," in *BMVC '06*, 2006, vol. 1, p. 6.

[8] David Ross, Jongwoo Lim, and Ming-Hsuan Yang, "Adaptive probabilistic visual tracking with incremental subspace update," in *ECCV '04*, 2004, pp. 470–482.

[9] Shai Avidan, "Ensemble tracking," *PAMI '07*, vol. 29, pp. 261–271, 2007.

[10] Jianyu Wang, Xilin Chen, and Wen Gao, "Online selecting discriminative tracking features using particle filter," in *CVPR '05*, 2005, vol. 2, pp. 1037–1042.

[11] Robert T Collins, Yanxi Liu, and Marius Leordeanu, "Online selection of discriminative tracking features," *PAMI '05*, vol. 27, pp. 1631–1643, 2005.

[12] Oliver Williams, Andrew Blake, and Roberto Cipolla, "Sparse bayesian learning for efficient visual tracking," *PAMI '05*, vol. 27, pp. 1292–1304, 2005.

[13] Shai Avidan, "Support vector tracking," *PAMI '04*, vol. 26, pp. 1064–1072, 2004.

[14] Nikunj C Oza, "Online bagging and boosting," in *IEEE international conference on Systems, Man and Cybernetics*, 2005, vol. 3, pp. 2340–2345.

[15] Yoav Freund, Robert E Schapire, et al., "Experiments with a new boosting algorithm," in *ICML '96*, 1996, vol. 96, pp. 148–156.

[16] Omar Javed, Saad Ali, and Mubarak Shah, "Online detection and classification of moving objects using progressively improving detectors," in *CVPR '05*, 2005, vol. 1, pp. 696–701.

[17] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 20, pp. 30–42, 2012.

[18] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML '08*, 2008, pp. 160–167.

[19] Yoshua Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, pp. 1–127, 2009.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114.

[21] Brody Huval, Adam Coates, and Andrew Ng, "Deep learning for class-generic object detection," *arXiv preprint arXiv:1312.6885*, 2013.

[22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv preprint arXiv:1311.2524*, 2013.

[23] Naiyan Wang and Dit-Yan Yeung, "Learning a deep compact image representation for visual tracking," in *NIPS '13*, 2013, pp. 809–817.

[24] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 9999, pp. 3371–3408, 2010.

[25] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *CVPR '13*, 2013.

[26] Antonio Torralba, Robert Fergus, and William T Freeman, "80 million tiny images: A large data set for non-parametric object and scene recognition," *PAMI '08*, vol. 30, pp. 1958–1970, 2008.

[27] Gene H. Golub and Charles F. Van Loan, *Matrix Computations (3rd Ed.)*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[28] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. on Signal Processing*, vol. 50, pp. 174–188, 2002.

[29] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang, "Real-time compressive tracking," in *ECCV '12*, pp. 864–877. 2012.

[30] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie, "Robust object tracking with online multiple instance learning," *PAMI '11*, vol. 33, pp. 1619–1632, 2011.

[31] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja, "Robust visual tracking via multi-task sparse learning," in *CVPR '12*, 2012, pp. 2042–2049.