Convolutional Neural Net Bagging for Online Visual Tracking

Hanxi Li[a,b], Yi Li[d,b,*], Fatih, Porikli[b,c]

[a] *Jiangxi Normal University, 99 Ziyang Road, Nanchang, Jiangxi, China*
[b] *Computer Vision Research Group, NICTA, Canberra, Australia*
[c] *College of Computer Science and Engineering, Australian National University*
[d] *Toyota Research Institute - North America, Ann Arbor, MI 48105*

Abstract

Recently, Convolutional Neural Nets (CNNs) have been successfully applied to online visual tracking. However, a major problem is that such models may be inevitably over-fitted due to two main factors. The first one is the label noise because the online training of any models relies solely on the detection of the previous frames. The second one is the model uncertainty due to the randomized training strategy. In this work, we cope with noisy labels and the model uncertainty within the framework of *bagging* (Bootstrap aggregating), resulting in efficient and effective visual tracking. Instead of using multiple models in a bag, we design a single multitask CNN for learning effective feature representations of the target object. In our model, each task has the same structure and shares the same set of convolutional features, but is trained using different random samples generated for different tasks. A significant advantage is that the bagging overhead for our model is minimal, and no extra efforts are needed to handle the outputs of different tasks as done in those multi-lifespan models. Experiments demonstrate that our CNN tracker outperforms the state-of-the-art methods on three recent benchmarks (over 80 video sequences), which illustrates the superiority of the feature representations learned by our purely online bagging framework.

Keywords:
visual tracking, deep learning, ensemble learning

*Corresponding author

Email addresses: `hanxili@nicta.com.au` (Hanxi Li), `yili@nicta.com.au` (Yi Li), `fatih.porikli@nicta.com.au` (Fatih, Porikli)

1. Introduction

Tracking-by-detection approaches prevail in recent years. These methods usually rely on predefined heuristics and construct a set of positive (objects) and negative (background) samples from the estimated object location. Often these samples have binary labels, which leads to a few positive samples and a large negative training set.

Since Convolutional Neural Networks (CNNs) have been successfully adopted for object detection, it is not surprising to witness a surge of deep learning methods for visual tracking [1, 2]. However, although online training has proven a huge benefits [3, 4, 5], the immediate adoption of CNN for *online* visual tracking is not straightforward.

To begin with, CNN requires a large number of training samples, which is often not available in visual tracking. Moreover, CNN tends to overfit to the most recent observation, *e.g.*, most recent instance dominating the model, which may result in the drift problem. Besides, CNN training is computationally intensive for online visual tracking. The slow updating speed could prevent the CNN model from being practical. Li *et al.* [6, 7] handled these problems by either using an ensemble of CNNs or a single CNN with different sampling methods for positive and negative classes. Most recently, as an enhanced version of [7], [8] achieves high tracking accuracies by exploiting color information and label uncertainties.

The underlying reason why online tracking is challenging is that the object locations, except the first frame, are not always reliable as they are estimated by the visual tracker and the uncertainty is unavoidable [4]. One can treat this difficulty as the label noise problem [9, 10, 11]. Furthermore, in the literature of deep learning, the highly-nonconvex loss function of CNN is usually optimized in a stochastic fashion [12, 13]. As a result, local optimal is almost inevitable in the training procedure. For offline training tasks, this difficulty could be alleviated using a large number of training epochs based on large-size training data [13]. In visual tracking, in contrast, the time budget is highly constrained and only hundreds of training samples are available for each frame. Given different initial parameters or different training data, the stochastic gradient descent (SGD) method will easily lead to totally different CNN models. The label noise and the model uncertainty could stimulus each other and cause serious object drifting accordingly. To cope with the model uncertainty, [6] proposes to cache the CNN models over time in a CNN pool and select the best one in the test phrase. However, this requires to a extra feature matching process in the test time and the multiple CNN models also slow down the tracking speed significantly. On the other hand, [7] and [8] employ the multiple-lifespan sampling strategy to handle the noisy labels in tracking. [8] obtained higher accuracy and efficiency than [6], but performs unstably as it relies on merely a In this work, we propose to solve the above two problem in one framework, *i.e.*, a CNN Bagging.

The bagging has a few superior characteristics. For example, bagging is more sensible than the methods based on multiple lifespans [14], because it does not require additional information to combine the detected results for multiple lifespans, and does not need to cope with the dilemma between long term and short term memory. However, the bagging usually results in significant computation loads, because each individual model needs to be updated simultaneously.

Instead of multiple CNNs, we propose a single multitask CNN for learning effective feature representations of the target object. In our model, all tasks share the same set of features and each task is trained using different set of random samples. Each task generates scores for all possible hypotheses of the object locations in a given frame, and the prediction of the object is obtained by simple soft-max operation of the scores in the current frame.

Our experiments on three recent benchmarks involving over 80 videos demonstrate that our method outperforms all the compared state-of-the-art algorithms and rarely loses the track of the objects. In addition, it achieves a practical tracking speed (from 1fps to 2.5fps depending on the sequence and setting), which is comparable to state of the art visual trackers. Our main contributions include:

- We proposed to use CNN bagging for coping with noisy labels and model uncertainty simultaneously in online visual tracking.
- We designed a single multitask CNN that implements the CNN bagging effectively.
- We achieved the best reported results in the literature at the speed up to 2.5fps.

2. Related work

Image features play a crucial role in many challenging computer vision tasks such as object recognition and detection. Unfortunately, in many *online* visual trackers features are manually defined and combined [15, 16, 17, 3]. Even though these methods report satisfactory results on individual datasets, hand-crafted feature representations would limit the performance of tracking. This necessitates good representation learning mechanisms for visual tracking that are capable of capturing the appearance effectively changes over time.

As a compelling tool for feature representation learning, deep learning has gained significant attention and has been successfully adopted to different computer vision applications. Different from the traditional hand-crafted features [18, 19], a multi-layer neural network architecture can efficiently capture hierarchical features that describe the data [20]. In particular, the CNN has shown superior performance on standard object recognition tasks [13, 21, 22] with minimal domain knowledge.

In the visual tracking literature, an ensemble of CNNs [6] and a single CNN [7, 8] with multiple lifespan samples have been proposed for online tracking. However, these methods either suffer from slow speed caused by the large-size CNN pool [6] or unstable output due to the randomized learning procedure for a single CNN [7, 8]. In this work, we employs a CNN bagging model with minimal computational overhead to strike the balance between robustness and tracking speed.

Please note that in recent years, some deep learning based visual trackers [2, 23, 24] have also been proposed and illustrate the state-of-the-art performances. However, most of them requires some offline learned features based on a large dataset, which is different from the purely online setting adopted in this paper.

3. Our approach

We first introduce the basic ideas and notations for the online visual tracking using CNN, then we propose a multitask CNN framework as solution to a bag of CNN models. We further provide our sampling procedure and our loss function.

3.1. The existing online CNN trackers

We briefly describe the state of the art CNN architecture for online visual tracking ([6, 7, 8]).

In visual tracking, a bounding box of the object of interest (foreground) is given in the first frame. The CNN-based tracker then online learn a foreground-background detector, and apply the detector to the next frame(s). Once an optimal location in a new frame is detected, it is regarded as the ground truth of this frame, and a set of foreground and background image patches in this frame are added to the training samples. The detector is re-trained and this detection-training process continues for subsequent frames. One can see that the above process can easily lead to drifting, if the training samples are contaminated during the online update process.

Let $\mathbf{x}_n$ and $\mathbf{l}_n \in \{[0,1]^{\mathrm{T}}, [1,0]^{\mathrm{T}}\}$ denote the the input patch and its ground truth label (background or foreground) respectively, and $f(\mathbf{x}_n; \Omega)$ be the predicted score of $\mathbf{x}_n$ with network weights Ω. When a new frame $\Gamma_{(t)}$ comes, we predict the object motion state $\mathbf{y}^*_{(t)}$ as

$$\mathbf{y}^*_{(t)} = \arg\max_{\mathbf{y}_n \in \mathcal{Y}} \left(f(\phi\langle \Gamma_{(t)}, \mathbf{y}_n \rangle; \Omega) \right), \tag{1}$$

where $\mathcal{Y}$ contains all the test patches in the current frame, and the operation $\phi\langle \Gamma, \mathbf{y}_n \rangle$ suffices to crop the features from Γ using the motion $\mathbf{y}_n$.

In [6, 7, 8], the CNN is learned based on multiple cues, which allows the CNN select the most informative ones in a data driven fashion. As to the model structure, the online CNN tracker used in [8] takes 32×32 image patches as input. The first convolution layer contains 12 kernels, each of size 13×13, followed by a max pooling operation that reduces the obtained feature map to a lower dimension. The second layer contains 18 kernels with size 7×7. This leads to a 72-dimensional feature vector as the output, after the pooling operation. The two fully connected layers firstly map the 72-D vector into the 8-D vector and then generate a 2-D confidence vector $\mathbf{s} = [s_1, s_2]^{\mathrm{T}} \in \mathcal{R}^2$, with s_1 and s_2 corresponding to the positive score and negative score, respectively. The CNN score is then given by

$$f(\mathbf{x}_n; \Omega) = s_n = s_1 \cdot \exp(s_1 - s_2). \tag{2}$$

In this work, our CNN bagging algorithm follows the settings of [8] for both image cue generation and single CNN structure. Readers can refer to [7, 8] for more details.

3.2. Bagging CNN architecture

This section describes the architecture for our CNN bagging approach. The loss function, learning procedure and the speedup are presented in the next section.

3.2.1. From a bag of CNNs to a multitask CNN

Bagging (bootstrap aggregation) is a learning algorithm designed to improve the model robustness, when small changes in the training set causes significant changes in a single model. As one of the ensemble learning method, Bagging uses multiple subsets of the training samples and each is used to train a different model. The results of the models are combined to create a single output.

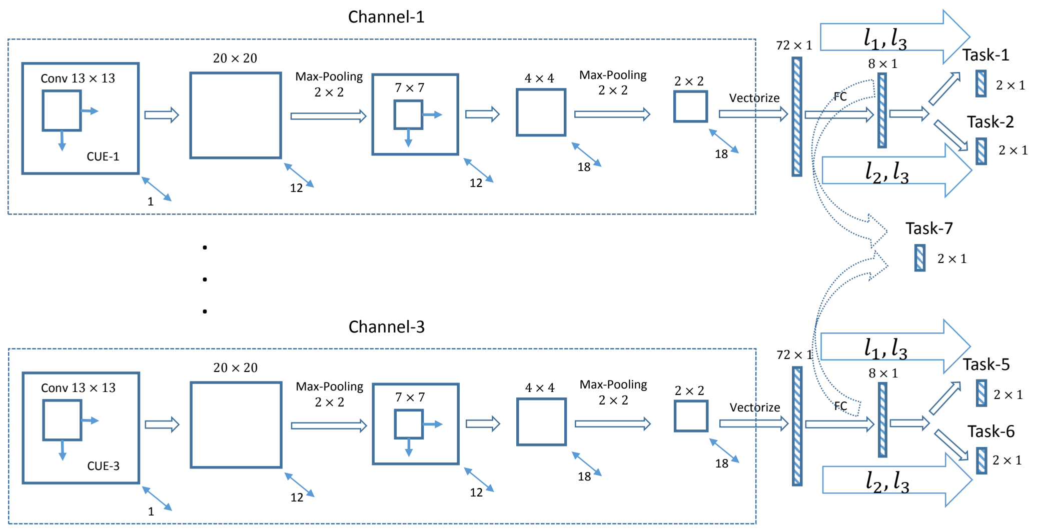


Figure 1: The architecture of our CNN tracker with multiple image cues. The dashed blocks on the left are the CNN channels for different image cues; the 7 tasks are essentially 7 linear mappings from higher dimensional spaces to 2-D spaces, with the training samples generated from different lifespans.

Training multiple copies usually implies increasing training time. Therefore, while the concept of bagging is very attractive, applying this learning algorithm naively to the online CNN methods such as [6] is impractical in visual tracking. We also want to note that the idea of using samples from different lifespan [14] may also suffer from this computational issue, because each individual model needs to be handled and updated simultaneously.

Neural networks can be set up for multi-task learning conveniently. For instance, it is natural to design a network that has some shared layers, and multiple separate higher-level layers for different tasks [25]. Motivated by the consensus that the features of the first few convolutional layers in CNN are generic, we proposed to use a single multitask CNN as bagging for visual tracking. In this architecture, each task shares the same set of convolutional features while its fully connected layers are trained with different training samples. Fig. 1 illustrates this idea. In specific, a single task is referred to as updating a single CNN model corresponding to a certain image cue obtained from a certain lifespan. In this work, we use 3 image cues and 2 different lifespans (l_1 and l_2 as shown in the figure) for generating positive samples [1]. This leads to 6 tasks to solve at each frame. In addition, to jointly update the fully-connected layers for all the image cues (see [7, 8]) is the 7-th task in our multi-task setting. The 7 tasks are solved alternatively in a random fashion as described in Sec. 3.3.2.

3.3. Loss function and randomized task selection for CNN training

3.3.1. Structural truncated loss function

For a single task (single CNN model), we using the same loss function as defined in [8]. The structural loss function is as follows

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left[\Delta(\mathbf{y}_n, \mathbf{y}^*) \cdot \| f(\phi\langle \Gamma, \mathbf{y}_n \rangle; \Omega) - \mathbf{l}_n \|_{\mathrm{T}} \right], \tag{3}$$

where $\mathbf{y}^*$ is the (estimated) motion state of the target object in the current frame. $\Delta(\mathbf{y}_n, \mathbf{y}^*)$ is based on the IoU criterion [26]. The underlying assumption of using this structural loss is that patches that are very close to object center and reasonably far from it may play more significant roles in training the CNN, while the patches in between are less important.

The function $\| \cdot \|_{\mathbb{T}}$ denotes the truncated l_2 norm which is visualized in Fig. 2. Mathematically, the truncated l_2 loss writes:

$$\|e\|_{\mathbb{T}} = \|e\|_2 \cdot \left(1 - \mathbb{1}\left[\|e\|_2 \leq \frac{\beta}{(1 + u \cdot l_n)}\right]\right), \tag{4}$$

where $u > 0$ and $l_n = \mathbf{l}_n(1)$, *i.e.*, the scalar label of the n^{th} sample. It is easy to see that with the truncated norm $\| \cdot \|_{\mathbb{T}}$ the backpropagation process only depends on the training samples with large errors, *i.e.*, $\| f(\phi\langle \Gamma, \mathbf{y}_n \rangle; \Omega) - \mathbf{l}_n \|_{\mathbb{T}} > 0$. As a result, by ignoring the samples with small errors the backpropagation procedure is significantly accelerated. Furthermore, Eq. 4 also reflects the intuition that tracking is more sensitive to the prediction errors on positive samples rather than those on negative samples. In the experiment we set $u = 4$ and $\beta = 0.0025$.

3.3.2. Randomized task selection

In the multi-task framework, the loss function can be formulated as a weighted sum of each task's loss. Let α_i and L_i $(1 \leq i \leq N)$ denote the weight and loss for

[1]the negative samples are generated using a fixed lifespan l_3

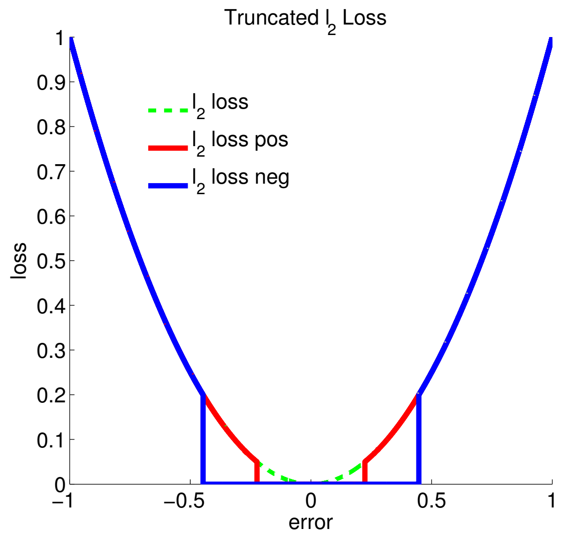


Figure 2: The truncated l_2 losses. The dashed green curve indicates the original l_2 loss, the red and blue curves are the truncated losses for positive and negative samples.

task i respectively,and L is the overall loss, the multi-task learning is formulated as:

$$\Omega^* = \min_{\Omega} L = \min_{\Omega} \frac{1}{N} \sum_{i=1}^{N} \alpha_i L_i. \tag{5}$$

If $\alpha_i \neq 0$ and $\alpha_j = 0$ $(\forall i \neq j)$, the learning is identical to task i. Since we employ SGD and back-propagation to approximately minimize the loss, the network parameters are updated using the weighted sum of the gradients from all tasks. Let w_q denote the q^{th} layer network parameter in Ω, D_i^q be the gradient for w_q from task i, and p be the learning rate. The following updating rule applies to each iteration:

$$w^q \leftarrow w^q - p \sum_{i=1}^{N} \alpha_i D_i^q. \tag{6}$$

Unfortunately, updating the CNN using Eq. 6 does not reduce computational cost compared to the naive idea that uses a set of identical CNN models. Note that putting a set of models into an unified multitask framework allows us to exploit the structure and eventually speed up the training significantly, we propose a randomized procedure for efficient training in this section.

We first define two categories of tasks: 1) a major task that updates both its fully connected layers and convolutional layers, and 2) minor tasks that only update their fully connected layers. In each model update, we select a task as the major task, and learn both convolutional and fully connected features for the major task. The rest of the tasks are regarded as minor tasks, and only their fully connected layers are updated. In this way, we can employ multiple tasks with only a marginal complexity increase. This also bypasses the difficulty in selecting α_i for sub-tasks.

In practice, we first update the normal multi-cue CNN (blue dashed block) using the same method as [8], with different lifespans for positive (lifespan l_1 as shown in Fig. 1) and negative (lifespan l_3) samples. Note that this learning process already solved 4 tasks we defined before. Then, the remaining 3 task is solved by updating the fully-connected layers corresponding to the 3 image cues with positive lifespan l_2. It is easy to see that compared with the single task solved in [7, 8], the multi-task proposed in this work only increase the computational burden marginally. That is because the learning stage of the convolutional layers dominates the whole procedure and the fully-connected layer for each image cue can be updated very efficiently.

3.4. Weighted aggregation with fast parameter selection

In the test phrase, the input image patch passes through the CNN model shown in Figure 1. This leads to 7 CNN scores $s_n{}^j = f(\mathbf{x}_n; \Omega^j), j = 1, 2, \cdots, 7$ for each test patch. Theoretically, one can learn a linear model $\mathbf{v} \in \mathcal{R}^7, \mathbf{v} \succ 0$ to combine them with different importances. However, to save the test time and also curb overfitting, we pre-fix the candidates pool $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_K]^\mathbf{T} \in \mathcal{R}^{K \times 7}$ of the weight vector $\mathbf{v}$, $\mathbf{1}^\mathbf{T}\mathbf{v} = 1$. Note the with a relatively large numerical resolution [2], K is not large and usually in the order of $1e5$.

Given that the training samples over all the past frames are $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and the corresponding prediction score vectors are $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N\}$, with each $\mathbf{s}_n = [s_n{}^1, \cdots, s_n{}^7]^\mathbf{T}$. We want to select a "optimal" $\mathbf{v}$ such that it yields the smallest training loss for all the observed training samples. This can be done easily via exhaustive search given K and N are both small.

3.5. Data augmentation and processing

Our other important implementation details include

[2]In this work we select $v_i \in \{\frac{1}{6}, \frac{2}{6}, \cdots, 1\}$

- All the training samples are flipped as augmented data to better curb the overfitting.
- The pixel values of each the image cue are normalized to the range $[0, 10]$, which aims at balancing the importance between different image cues.

4. Experiments

4.1. Benchmarks and experiment setting

We evaluated our method on three recently proposed tracking benchmarks: the CVPR2013 Visual Tracker Benchmark [27], the VOT2013 Challenge Benchmark [28], and the TB-50 benchmark [29]. Most parameters of the CNN tracker are given in Sec. 3.2. In addition, there are some motion parameters for sampling the image patches. In this work, we only consider the displacement Δ_x, Δ_y and the relative scale $s = h/32$ of the object, where h is object's height.

Given a new frame, we sample 1500 random patches in a Gaussian distribution which centers on the previous predicted state. The standard deviation for the three dimensions are $\min(10, 0.5 \cdot h)$, $\min(10, 0.5 \cdot h)$ and $0.01 \cdot h$, respectively. Note that, all parameters are fixed for all videos in both two benchmarks; no parameter tuning is performed for any specific video sequence.

We run our algorithm in Matlab with an unoptimized code mixed with CUDA-PTX kernels. The hardware environment includes one quad-core CPU, 16GB Memory, and one NVIDIA GTX980 GPU.

4.2. Results on the CVPR2013 benchmark

The CVPR2013 Visual Tracker Benchmark [27] contains 50 fully annotated sequences. These sequences include many popular sequences used in the online tracking literature over the past several years. The first row in Fig. 3 shows the first frames of some of the videos. For better evaluation and analysis of the strength and weakness of tracking approaches, these sequences are annotated with the 11 attributes including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. Here, we compare our method with other 11 tracking methods. Among the competitors, TPGR [5] and KCF [30] are the most recently state-of-the-art visual trackers; TLD [31], VTD [32], CXT [33], ASLA [34], Struck [3], SCM [35] are the top-6 methods as reported in the benchmark; CPF [15], IVT [36] and MIL [4] are classical tracking methods which are used as baselines.

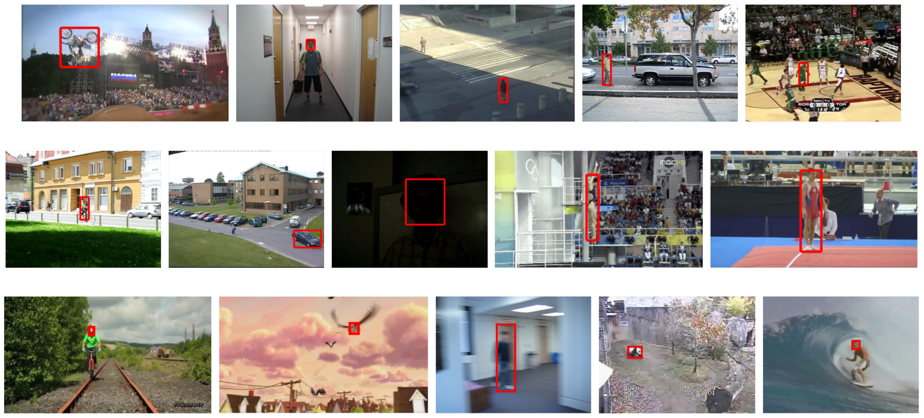


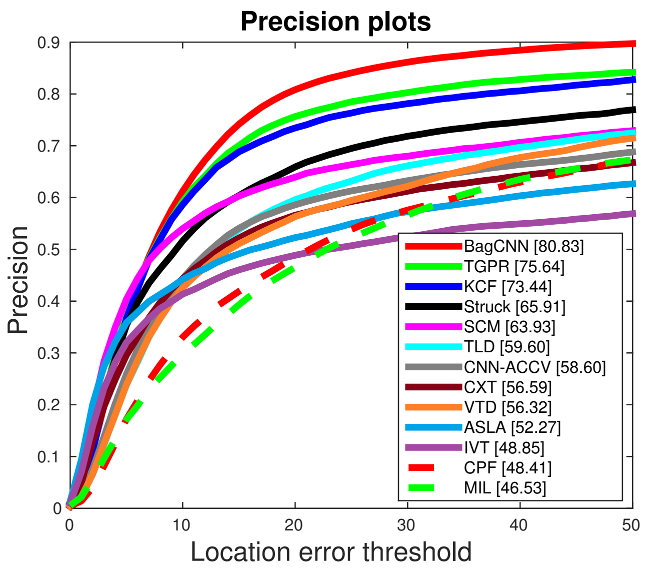
Figure 3: The first frames of the video sequences from the CVPR2013 (first row), the VOT2013 (second row), and the TB-50 benchmarks (third row). From top left to bottom right: MotorRolling, Boy, Crossing, David3, Basketball, Bicycle, Car, David, Diving, Gymnastics, and the sequences in the TB-50 benchmark. The red blocks are the object locations given in the first frame.

The tracking results are evaluated via the following two measurements: 1) Tracking Precision (TP), the percentage of the frames whose estimated location is within the given distance-threshold (τ_d) to the ground truth, and 2) Tracking Success Rate (TSR), the percentage of the frames in which the overlapping score between the estimated location and the ground truth is larger than a given overlapping-threshold (τ_o). Following the setting in the recently published work [5, 30], we conduct the experiment using the OPE (one-pass evaluation) evaluation strategy for a better comparison to the latest methods.

First, we evaluated our method using different criteria (different τ_d, τ_o). Specifically, we evaluate the trackers with the thresholds $\tau_d = 1, 2, \cdots, 50$ for TP. For TSR, we use the thresholds $\tau_o = 0$ to 1 at the step of 0.05. Accordingly we generated the precision curves and the success-rate curves for each tracking method (Fig. 4).

From the plots we can see that overall the CNN tracker ranks the first (red curves) in both TP and TSR evaluations. The proposed method outperformed all the other trackers when $\tau_o < 0.68$ and $\tau_d > 10$. When the overlap thresholds are tight (*e.g.* $\tau_o > 0.75$ or $\tau_d < 5$), our tracker has similar behavior to rest of the trackers we tested.

In many applications, it is important not to loose the target object. Fig. 4 shows that when the evaluation threshold is reasonably loose, (*i.e.*, $\tau_o < 0.45$ and $\tau_d > 20$) our algorithm is very robust with both the accuracies higher than 80%.



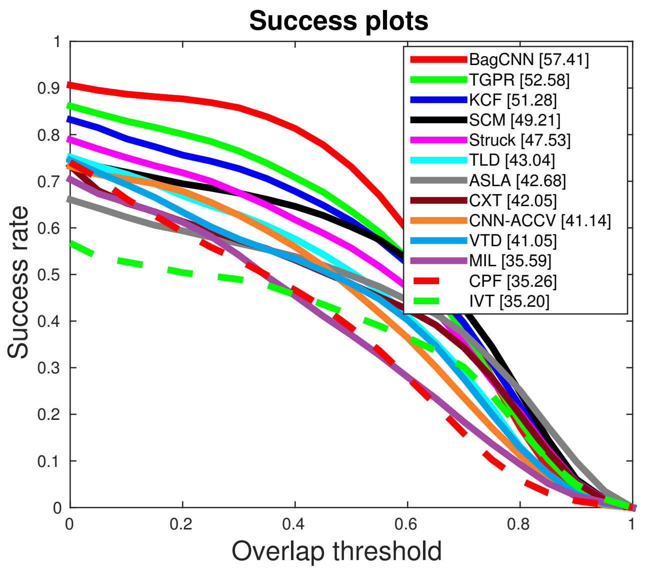


Figure 4: The Precision Plot (left) and the Success Plot (right) of the tracking results on the CVPR2013 benchmark. Note that the color of one curve is determined by the rank of the corresponding trackers, not their names.

Notably, it achieves the accuracies around 90% when $\tau_o < 0.3$ and $\tau_d > 30$, which has a significant performance gain compared to the state of the art and suggests that our tracker rarely looses the object.

Notably, our tracker performs significantly better in fast motion, illumination change, and occlusion, which are some of the most difficult scenarios in tracking. Please refer to the Supplemental Materials for the detailed comparison. This demonstrates that our bagging mechanism is very effective in tackling noisy labels and avoids overfitting.

It is also worth noting that from the previous CNN-based tracker (CNN-ACCV in the plots), significant performance gain has been made thanks to the proposed Bagging framework and other modifications.

4.3. Results on the VOT2013 benchmark

The VOT2013 Challenge [28] provides an evaluation kit and the dataset with 16 fully annotated sequences for evaluating tracking algorithms in realistic scenes subject to various common conditions (the second row in Fig. 3).

	Our	PLT	LGTpp	FoT	EDFT	LGT
Acc. Rank	**9.22**	9.98	12.57	9.44	11.10	14.01
Rob. Rank	**6.40**	6.79	7.79	12.87	11.72	8.86
Rank	**7.81**	8.39	10.18	11.15	11.41	11.44

Table 1: The performance comparison between CNN tracker and the top-5 trackers on the VOT2013 benchmark. The best score in each row is shown in red while the second best is shown in blue.

The tracking performance in the VOT2013 Challenge Benchmark is primarily evaluated with two evaluation criteria: accuracy and robustness. The accuracy measure is the average of the overlap ratios over the valid frames of each sequence while the tracking robustness is the average number of failures over 15 runs. A tracking failure happens once the overlap ratio measure drops to zero and an re-initialization of the tracker in the failure frame is conducted so it can continue. According to the evaluation protocol, three types of experiments are conducted. In Experiment-1, the tracker is run on each sequence in the dataset 15 times by initializing it on the ground truth bounding box. The setting of Experiment-2 is the same to Experiment-1, except that the initial bounding box is randomly perturbed in the order of ten percent of the object size. In Experiment-3, the colorful frames are converted into grayscale images.

We follow the evaluation protocol to compare our method against 27 tracking algorithms provided in the benchmark. On average, the proposed method ranks the first for both accuracy and robustness comparison (Table. 1). For each row, the best score is highlighted in bold and red and the second best score is denoted by blue.

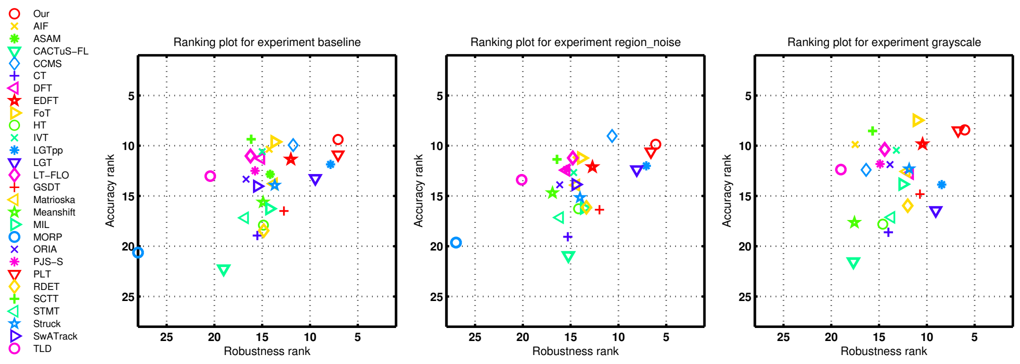


Figure 5: The rank plots of accuracy and robustness for the VOT2013 benchmark (Experiments 1-3, respectively. See text for details). Note that the for a good tracker, the symbol is usually shown in the top-right area.

We further show the scores for Experiments 1-3, respectively (Fig. 5). Our method (denoted by red circle) is located in the top-right compared to other methods, which means higher accuracy and better robustness. Compared to other competitive tracking methods such as PLT [28], FoT [37], EDFT [38] and LGT++ [39], our method has a noticeable performance gain in accuracy and/or robustness. Specifically, our method achieves the best robustness scores for all the scenarios while ranks the second in accuracy for all the experimental settings.

Note that the scores listed in the plots in Fig. 5 are rank-based, which is different from the measuring criterion used in the CVPR2013 benchmark. This offers a different point of view for assessing the tracking method. Therefore, the performance on the VOT2013 benchmark justifies the superiority of the proposed tracker.

4.4. Results on TB-50 benchmark

TB-50 is the latest benchmark dataset (the third row in Fig. 3). It contains half of sequences in the full benchmark (TB-100), which has 100 sequences from recent literature. There are 22 new sequences in TB-50 that are not included in CVPR2013 dataset. Following the CVPR2013 benchmark, the test sequences are tagged with 11 attributes, and 29 publicly available visual trackers have been tested.

The evaluation criteria of TB-50 are similar to those in the CVPR2013 benchmark, with the addition of SRER (Spatial Robustness Evaluation with Restart). As pointed out in [29], the computational cost is very high. While our tracker is fast in the category of CNN-based method, in our experiment we choose not to compare to this criterion due to this reason, and report the scores using the OPE score.

Since the OPE data for other methods on this dataset are not publicly available, we only report our results compared with KCF [30] which is one of the state-of-the-art method in Fig. 6. From the plots we can observe the big performance gap between these two tracking methods. Compared to the results reported in the project website of [29] (`https://sites.google.com/site/benchmarkpami/`), our method outperforms state of the art methods on this latest benchmark. A further comparison with Fig. 4 suggests that the TB-50 dataset is a more challenging dataset than the CVPR2013, because both Precision and Success scores decrease. More results for individual categories are shown in the Supplemental Materials.

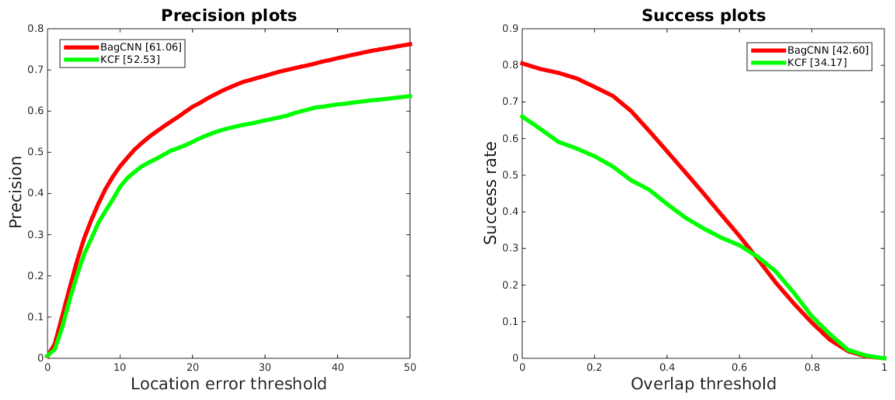


Figure 6: The Precision Plot (left) and the Success Plot (right) of the tracking results of BagCNN and KCF on the TB-50 benchmark.

4.5. Tracking speed analysis

We report the average speed (in fps) of the proposed method in Table 2. Note that there are two kinds of average speed scores: the average fps over all the sequences and the average fps over all the frames. The latter one reduces the influence of short sequences where the initialization process usually dominates the computational burden.

Sequence Average	Frame Average
1.32fps	1.68fps

Table 2: The tracking speed of the proposed method.

We can see that our method tracks the object at an average speed around 1.7fps. Considering that the speed of TPGR is around 3fps [5] and those of the Sparse Representation based methods are usually lower than 2.5fps according to [14], this suggests that our method achieves comparable speed to the state of the art methods.

5. Conclusion

This paper proposed a single multitask CNN as bagging for coping with noisy labels in visual tracking. We developed an efficient training method for the multitask CNN, resulting minimal overhead in training. Together with a new loss function and an improved sampling process, our CNN tracker outperformed state of the art methods on three recently proposed benchmarks (over 80 video sequences), which demonstrates the superiority of our purely online bagging framework.

6. Reference

[1] J. Fan, W. Xu, Y. Wu, Y. Gong, Human tracking using convolutional neural networks, Trans. Neur. Netw. 21 (10) (2010) 1610–1623. 2

[2] N. Wang, D.-Y. Yeung, Learning a deep compact image representation for visual tracking, in: NIPS 2013. 2, 4

[3] S. Hare, A. Saffari, P. H. Torr, Struck: Structured output tracking with kernels, in: ICCV 2011, pp. 263–270. 2, 3, 10

[4] B. Babenko, M.-H. Yang, S. Belongie, Visual tracking with online multiple instance learning, in: CVPR, 2009, pp. 983–990. 2, 10

[5] J. Gao, H. Ling, W. Hu, J. Xing, Transfer learning based visual tracking with gaussian processes regression, in: ECCV, Springer, 2014, pp. 188–203. 2, 10, 11, 16

[6] H. Li, Y. Li, F. Porikli, Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking, BMVC. 2, 3, 4, 5, 6

[7] H. Li, Y. Li, F. Porikli, Robust online visual tracking with an single convolutional neural network, ACCV. 2, 4, 5, 7, 9

[8] H. Li, Y. Li, F. Porikli, Deeptrack: Learning discriminative feature representations online for robust visual tracking, arXiv preprint arXiv:1503.00072. 2, 3, 4, 5, 7, 9

[9] N. D. Lawrence, B. Schölkopf, Estimating a kernel fisher discriminant in the presence of label noise, in: ICML, 2001, pp. 306–313. 2

[10] P. M. Long, R. A. Servedio, Random classification noise defeats all convex potential boosters, Machine Learning 78 (3) (2010) 287–304. 2

[11] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, A. Tewari, Learning with noisy labels, in: NIPS, 2013, pp. 1196–1204. 2

[12] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML, 2010, pp. 807–814. 2

[13] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS 2012. 2, 4

[14] J. Xing, J. Gao, B. Li, W. Hu, S. Yan, Robust object tracking with online multi-lifespan dictionary learning, in: ICCV, 2013, pp. 665–672. 3, 6, 16

[15] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: ECCV 2002. 3, 10

[16] R. T. Collins, Y. Liu, M. Leordeanu, Online selection of discriminative tracking features, Trans. PAMI 27 (10) (2005) 1631–1643. 3

[17] A. Adam, E. Rivlin, I. Shimshoni, Robust fragments-based tracking using the integral histogram, in: CVPR 2006, Vol. 1. 3

[18] D. G. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110. 4

[19] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, Vol. 1, 2005, pp. 886–893. 4

[20] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, Trans. PAMI 35 (8) (2013) 1798–1828. 4

[21] D. C. Ciresan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: CVPR 2012. 4

[22] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: CVPR, 2014, pp. 580–587. 4

[23] N. Wang, S. Li, A. Gupta, D.-Y. Yeung, Transferring rich feature hierarchies for robust visual tracking, arXiv preprint arXiv:1501.04587. 4

[24] K. Zhang, Q. Liu, Y. Wu, M.-H. Yang, Robust tracking via convolutional networks without learning, arXiv preprint arXiv:1501.04505. 4

[25] R. Caruana, Multitask learning, Machine learning 28 (1) (1997) 41–75. 6

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, IJCV 88 (2) (2010) 303–338. 7

[27] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, CVPR 2013. 10

[28] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al., The visual object tracking vot2013 challenge results, in: ICCV Workshops (ICCVW), 2013, pp. 98–111. 10, 13, 14

[29] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, PAMI, 2015. 10, 14, 15

[30] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, Trans. PAMIdoi:10.1109/TPAMI.2014.2345390. 10, 11, 15

[31] Z. Kalal, J. Matas, K. Mikolajczyk, Pn learning: Bootstrapping binary classifiers by structural constraints, in: CVPR 2010, pp. 49–56. 10

[32] J. Kwon, K. M. Lee, Visual tracking decomposition, in: in CVPR, 2010. 10

[33] T. B. Dinh, N. Vo, G. Medioni, Context tracker: Exploring supporters and distracters in unconstrained environments, in: CVPR 2011, pp. 1177–1184. 10

[34] X. Jia, H. Lu, M.-H. Yang, Visual tracking via adaptive structural local sparse appearance model, in: CVPR 2012, pp. 1822–1829. 10

[35] W. Zhong, H. Lu, M.-H. Yang, Robust object tracking via sparsity-based collaborative model, in: CVPR 2012, pp. 1838–1845. 10

[36] D. A. Ross, J. Lim, R.-S. Lin, M.-H. Yang, Incremental learning for robust visual tracking, IJCV 77 (1-3) (2008) 125–141. 10

[37] A. Wendel, S. Sternig, M. Godec, Robustifying the flock of trackers, in: 16th Computer Vision Winter Workshop, 2011, p. 91. 14

[38] M. Felsberg, Enhanced distribution field tracking using channel representations, in: ICCV Workshops (ICCVW), 2013, pp. 121–128. 14

[39] J. Xiao, R. Stolkin, A. Leonardis, An enhanced adaptive coupled-layer lgtracker++, in: ICCV Workshops (ICCVW), 2013, pp. 137–144. 14