BranchOut: Regularization for Online Ensemble Tracking with Convolutional Neural Networks

Bohyung Han
POSTECH, Korea
bhhan@postech.ac.kr

Jack Sim
Google Inc.
jacksim@google.com

Hartwig Adam
Google Inc.
hadam@google.com

Abstract

We propose an extremely simple but effective regularization technique of convolutional neural networks (CNNs), referred to as BranchOut, for online ensemble tracking. Our algorithm employs a CNN for target representation, which has a common convolutional layers but has multiple branches of fully connected layers. For better regularization, a subset of branches in the CNN are selected randomly for online learning whenever target appearance models need to be updated. Each branch may have a different number of layers to maintain variable abstraction levels of target appearances. BranchOut with multi-level target representation allows us to learn robust target appearance models with diversity and handle various challenges in visual tracking problem effectively. The proposed algorithm is evaluated in standard tracking benchmarks and shows the state-of-the-art performance even without additional pretraining on external tracking sequences.

1. Introduction

Visual tracking is valuable source of low-level information for high-level video understanding, so it has been applied to many computer vision tasks such as action recognition [6, 35], event detection [24], object detection from video [21], and so on. Despite tremendous amount of efforts, visual tracking is still regarded as a challenging problem since there exist a lot of variations imposed on target and surrounding background, and it is not straightforward to handle all the variations in a single framework. Most of all, it is extremely difficult to learn representative but adaptive features for robust tracking, especially in online scenarios.

We propose a novel visual tracking algorithm focusing on target appearance modeling, where the appearance is learned by a convolutional neural network (CNN) with multiple branches as shown in Figure 1. The target state is estimated by an ensemble of all branches while online model update is performed by the standard error backpropagation. In addition, we allow the individual branches to have different numbers of fully connected layers and maintain multi-level target representations.

The main challenge in this ensemble approach is how to decorrelate multiple branches and diversify learned models to maximize benefit of ensemble. Note that our problem is particularly challenging because training should be performed online with only a limited number of training examples in the presence of label noises. To deal with these challenges, we take an extremely simple strategy, *BranchOut*, which disregards a subset of the branches in the CNN chosen randomly for model update. This technique is helpful to maintain diversity of target appearance models and achieve performance improvement over deterministic methods. The proposed learning framework shares the motivation with Dropout [32] and DropConnect [34], where a subset of activations or weights in the fully connected layers are set to zeros randomly for each mini-batch during training. Our contribution is summarized as follows:

- We propose a simple but effective regularization technique, BranchOut, which is well-suited for online ensemble tracking. BranchOut alleviates the limitations of naïve ensemble learning approach—lack of model diversity and noisy labels of training data.
- Our network has a different number of fully connected layers in individual branches and maintains multi-level representations based on a CNN using the branches.
- We explore various options of online ensemble learning for visual tracking and verify the effectiveness of BranchOut and multi-level representation. Our algorithm illustrates the state-of-the-art performance even without pretraining with external tracking videos.

The rest of this paper is organized as follows. We first review existing visual tracking algorithms in Section 2. The proposed online stochastic learning and visual tracking algorithm design are described in Section 3 and 4, respectively. Section 5 presents experimental results with discussion, and Section 6 concludes our paper.

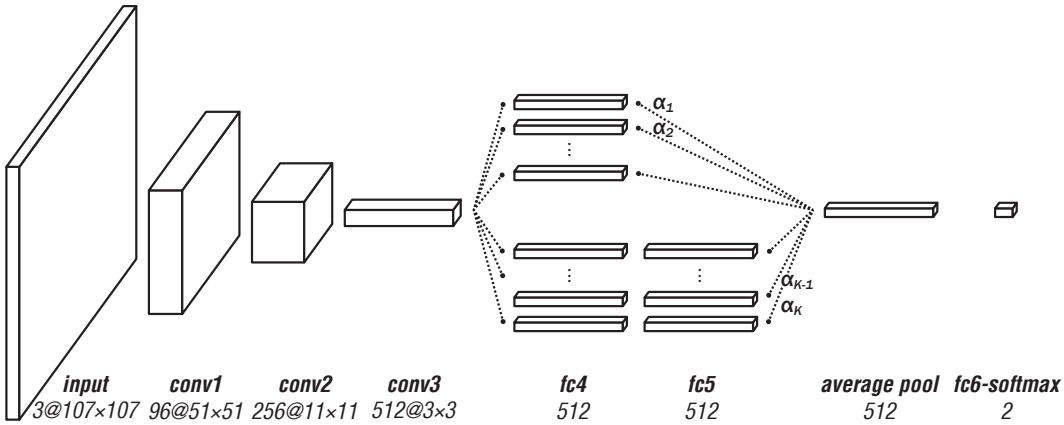


Figure 1. The proposed architecture. The network is composed of three convolutional layers and has multiple branches with fully connected layers. Each branch may have different number of layers, and one or two fully connected layers are integrated in our experiment.

2. Related Work

Visual tracking has a long history and there are tremendously many papers published in the last few decades. However, due to space limitation, we will review several active classes of methodology only in this section.

Tracking algorithms based on correlation filters are popular these days. This trend is mainly attributed to their great performance in terms of accuracy and efficiency. Bolme *et al*. [3] have introduced a minimum output sum of squared error (MOSSE) filter for visual tracking. Kernelized correlation filters (KCF) using circulant matrices [15] are employed to handle multi-channel features in Fourier domain. DSST [7] decouples the filters for translation and scaling to achieve accurate scale estimation, and MUSTer [17], motivated by a psychological memory model, utilizes short- and long-term memory stores for robust appearance modeling. Tracking algorithms relying on correlation filters often suffer from boundary effects. To alleviate this issue, [11] proposes the Alternating Direction Method of Multipliers (ADMM) technique, and Spatially Regularized Discriminative Correlation Filters (SRDCF) [9] introduces a spatial regularization term.

As machine learning techniques for object detection makes great progress in recent years, tracking-by-detection becomes one of the standard approaches for visual tracking. In this framework, tracking is performed using a classifier that distinguishes target object from background. The critical challenge in this approach is how to avoid drift problem during online learning, where only a small number of training examples are available and the labels are potentially noisy. Various learning frameworks have been investigated, and they include structured SVMs [14], multiple instance learning [1], P-N learning [20], online boosting [13], etc.

Although the above approaches work fairly well in constrained environments, they have a common inherent limitation that they rely on low-level hand-crafted features, which are not sufficiently robust to various challenges imposed on target objects. CNNs have achieved great performance improvement in many computer vision tasks, and visual tracking is not an exception. Recent approaches often transfer CNNs pretrained on a large-scale dataset such as ImageNet [31]. CNN-SVM [16] combines a pretrained CNN and online SVMs to obtain target-specific saliency maps for tracking and segmentation. Wang *et al*. [36] employ a fully convolutional framework and propose a feature map selection method to generate foreground heat maps, while Ma *et al*. [28] adaptively train correlation filters using feature hierarchy in a pretrained CNN. DeepSRDCF [8] integrates CNN-based features into [9] for performance improvement. To reduce the drawback from single resolution feature maps, [10] proposes to integrate multi-resolution deep feature maps through implicit interpolation. Since the CNNs trained for image classification task may not be appropriate for visual tracking, MDNet [30] attempts to train a CNN using external tracking sequences in a multi-domain learning framework. This approach is very successful and shows outstanding performance compared to all the prior methods; its performance is competitive even without the multi-domain pretraining stage.

Ensemble learning based on CNNs has been studied actively for visual tracking. TCNN [29] maintains multiple CNNs in a tree structure to learn ensemble models and estimate target states while allowing all CNNs to share convolutional layers. This approach achieves competitive performance even to MDNet without pretraining on the sequences for tracking. STCT [37] has a similar motivation to ours in the sense that an ensemble CNN-based classifier is trained to reduce correlation across models. Our algorithm is closely related to [26], which realizes stochastic learning using bagging [4]—standard method to diversify training examples for ensemble learning.

3. Stochastic Ensemble Learning

This section describes our stochastic ensemble learning technique, referred to as BranchOut, for visual tracking, and discusses why the proposed framework is effective to maintain model diversity and improve tracking performance potentially.

3.1. Stochastic Learning for Regularization

Our main goal is to develop an ensemble tracking algorithm based on a CNN with multiple branches by a proper regularization. This objective is hard to achieve particularly because there are only a limited number of training examples while labels are potentially noisy because they need to be estimated by imperfect tracking algorithms. To deal with such challenging situations, we randomly select a subset of branches for model updates and hope each model to evolve independently over time. This idea is somewhat related to bagging technique in random forests [4], but we need more reliable methods well-suited for online visual tracking since the size of training data is small and there are many redundant examples due to temporal coherency.

In deep neural networks, there are a few techniques proposed by the same motivation. Dropout [32] sets a subset of activations in fully connected layers to zeros randomly to regularize CNNs. This idea is generalized in [34], where, instead of turning off activations, a subset of weights are disregarded randomly. For the regularization of convolutional layers, [33] introduces SpatialDropout technique, which makes all the values in the randomly selected channels to zeros. This technique is successfully applied to joint point estimation in human body. Wang *et al*. [37] point out the potential drawback of SpatialDropout [33] and apply binary masks to the output of convolutional feature maps for model regularization in visual tracking application.

CNNs with stochastic depth [19] is a novel and interesting framework for regularization, where a subset of layers are randomly dropped and bypassed with identity functions. Lee *et al*. [25] propose an efficient stochastic gradient descent approach for stochastic multiple choice learning, which minimizes the loss with respect to an oracle. Although this algorithm is impractical due to the absence of model selection technique, it conceptually demonstrates that stochastic learning may be helpful for performance improvement of ensemble classifier.

3.2. BranchOut

Let $\mathbb{D} = \{(\mathbf{x}^i, \mathbf{y}^i) \mid i = 1, \dots, M\}$ be a training dataset for target appearance model update, where $\mathbf{x}^i$ is an image patch and $\mathbf{y}^i = (y^i_+, y^i_-)^\top$ is the binary label of $\mathbf{x}^i$, *i.e*., $(1, 0)^\top$ for positive and $(0, 1)^\top$ for negative. When we train a CNN with multiple branches, *e.g*., CNN in Figure 1, a subset of branches are selected randomly by a Bernoulli distribution. Specifically, if we assume that there exist K branches, a binary random variable α_k $(k = 1, \dots, K)$ is obtained by

$$\alpha_k \sim \text{Bernoulli}(p_k), \tag{1}$$

where p_k is the parameter of Bernoulli distribution corresponding to the k-th branch. The binary variable α_k indicates whether the k-th branch is to be selected for update. Note that our regularization is not performed per mini-batch but per batch; this is because training set is small and mostly redundant due to online learning restriction and temporal coherency of videos. After going through the forward pass of the multi-branch CNN parametrized by θ_k $(k = 1, \dots, K)$, the aggregated loss corresponding to all branches is given by

$$\mathcal{L} = -\sum_{i=1}^{M_b}\sum_{k=1}^{K} \alpha_k \left[\sum_{*\in\{+,-\}} y^i_* \, \mathcal{F}_*(\mathbf{x}^i; \theta_k) \right], \tag{2}$$

where M_b is mini-batch size, and $\mathcal{F}_+(\cdot; \theta_k)$ and $\mathcal{F}_-(\cdot; \theta_k)$ denote the outputs of the nodes in SOFTMAX layer corresponding to positive and negative labels, respectively.

The gradients for each mini-batch are computed by computing the partial derivatives of all relevant branches, which is formally given by

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\sum_{i=1}^{M_b}\sum_{k=1}^{K} \alpha_k \frac{\partial}{\partial \theta_k} \left[\sum_{*\in\{+,-\}} y^i_* \, \mathcal{F}_*(\mathbf{x}^i; \theta_k) \right]. \tag{3}$$

We only update the fully connected layers, and adopt only one or two of them for training efficacy since it is difficult to learn more than two fully connected layers online based on a limited number of training data.

3.3. Discussion

We claim that BranchOut provides diverse models and effective regularization. Suppose that the model at time t_1, denoted by $\mathcal{F}_{t_1}(\mathbf{x}^i; \theta_k)$, evolves to $\mathcal{F}_{t_2}(\mathbf{x}^i; \theta_k)$ at time t_2. In our online learning scenario, training datasets $\mathbb{D}_t$ changes dynamically overtime but there are substantial overlap between temporally close datasets, such as $\mathbb{D}_t$ and $\mathbb{D}_{t+1}$. However, for simplicity at the moment, let us assume that all the training datasets between t_1 and t_2 are identical, *i.e*., $\mathbb{D}_{t_1+1} = \mathbb{D}_{t_1+2} = \cdots = \mathbb{D}_{t_2}$, and compare the two models learned by deterministic and stochastic approaches. Note that the deterministic learning means that all the K models are updated whenever model updates are triggered.

After $|t_2 - t_1|$ deterministic model updates with the same training datasets, all the branches with the same architecture are likely to converge to the almost same model since they are updated with the same data for a substantial amount of iterations. Contrary to the model, stochastic learning with BranchOut is supposed to have at least several different

models—underfitted models, near optimal models and overfitted models—depending on how many times each model is involved in the updates.

If we consider more general cases that have gradually changing training datasets, the diversity of stochastic learning approach is even more prominent compared to the deterministic one. One reason is that, even with substantial overlap between training datasets at the time steps temporally close to each other, it can implicitly generate a variety of combinations of training datasets after a certain number of time steps. On the other hand, the negative impact of noisy labels obtained from failed targets may be reduced by sharing the risk across multiple models. In terms of computational complexity, BranchOut approach is obviously cheaper than naïve updates of all branches.

4. Tracking Algorithm

This section describes our tracking algorithm based on the CNN with multiple branches using BranchOut technique for stochastic ensemble.

4.1. CNN Initialization

The CNN integrated in our tracking algorithm has three convolutional layers (CONV1-3), each of which is followed by a rectified linear unit (RELU) layer and a max pooling (MAXPOOL) layer as illustrated in Figure 1. The three convolutional layers are initialized using VGG-M [5] pretrained on ImageNet [31]. Suppose that there are K separate branches connected to the last MAXPOOL layer and each branch based on fully connected (FC) layers is parametrized by θ_k ($k = 1, \dots, K$). In our implementation, the number of branches is 10, and each of the branches is composed of one or two FC layers. When two FC are employed, RELU and DROPOUT layers are located between the two FC layers. All weights in all the FC layers are initialized randomly using zero-mean Gaussian distributions.

At the first frame, we extract positive and negative training sets, denoted by $\mathbb{S}_1^+$ and $\mathbb{S}_1^-$ respectively, based on ground-truth bounding box information, and train the FC layers in the network using the standard stochastic gradient descent method. All branches are trained with same training examples, but they are shuffled independently so that each branch has some degree of diversity at least.

4.2. Main Loop of Tracking

Once the initial model is constructed, we start to track the target defined at the first frame. Given the input frame at time t, we draw dense samples $\mathbf{x}_t^i$ ($i = 1, \dots, N$) from a Gaussian distribution centered at the previous target state in translation and scale dimension and compute the scores from all branches. The target state is estimated by

$$\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t^i} \sum_{k=1}^{K} \mathcal{F}_+(\mathbf{x}_t^i; \theta_k), \tag{4}$$

where $\mathcal{F}_+(\mathbf{x}_t^i; \theta_k)$ denotes the positive score of $\mathbf{x}_t^i$ from the final SOFTMAX layer of the k-th branch.

To improve localization accuracy, we adopt bounding box regression [12] as suggested in [30]. We train the bounding box regressor using 1000 training examples at the first frame only and apply the model to all the subsequent frames because learning a bounding box regressor is time consuming and learned regression model using the examples from other frames may not be reliable due to absence of ground-truths. The detailed implementation of bounding box regressor is described in [12].

One of the critical factors in online learning is how to construct training examples. If the score of the estimated target is positive at frame t, we collect training examples for future model updates. Since ground-truths are not available, we generally rely on the estimated target locations. The positive examples extracted from frame t, denoted by $\mathbb{S}_t^+$, are composed of the bounding boxes with more than 0.7 IoU while examples in $\mathbb{S}_t^-$ have less than 0.3 IoU.

4.3. Model Update Strategy

The CNN maintaining target appearance models needs to be adaptive to new training examples. We employ BranchOut for online learning described in Section 3. Specifically, given K branches with FC layers, our algorithm selects a subset of branches randomly whenever model update is required. We do not update any of convolutional layers but fine-tune fully connected layers only. The model is optimized by stochastic gradient descent method.

There are two different situations to trigger model update module. One is periodic update, which simply revises our CNN-based model in a regular term, *e.g.*, every 10 frame. The other is when the positive classification score of the estimated target $\mathbf{x}_t^*$ in Eq. (4) is below 0.5. In both cases, we train the CNN using the BranchOut technique with the training examples obtained from the recent τ successful frames at which the estimated target scores are positive.

4.4. Implementation Details

We utilize the implementation of MDNet [30] as baseline. To train the CNN, we extract 50 positive examples and 200 negative examples based on IoU measure as described in Section 4.2. However, at the first frame, since we have to initialize FC layers from scratch, we extract much more examples. In our implementation, $|\mathbb{S}_1^+| = 500$ and $|\mathbb{S}_1^-| = 5000$. We store the training examples from the last $\tau = 20$ successful frames.

Algorithm 1 Stochastic ensemble tracking by BranchOut

Require: CNN with K branches of FC layers parametrized by $\Theta = \{\theta_1, \dots, \theta_K\}$, and initial target state $\mathbf{x}_1$
Ensure: Estimated target states $\mathbf{x}_t^*$

Randomly initialize $\Theta = \{\theta_1, \dots, \theta_K\}$.
Train a bounding box regression model.
Draw positive samples $\mathbb{S}_1^+$ and negative samples $\mathbb{S}_1^-$.
Update Θ using $\mathbb{S}_1^+$ and $\mathbb{S}_1^-$.
$\mathbb{T} \leftarrow \{1\}$.
repeat
  Draw target candidate samples $\mathbf{x}_t^i$ $(i = 1, \dots, N)$.
  Find the optimal target state $\mathbf{x}_t^*$ by Eq. (4).
  if $\mathcal{F}_{t,+}(\mathbf{x}_t^*) > 0.5$ **then**
    Draw training samples $\mathbb{S}_t^+$ and $\mathbb{S}_t^-$.
    $\mathbb{T} \leftarrow \mathbb{T} \cup \{t\}$.
    if $|\mathbb{T}| > \tau$ **then**
      $\mathbb{T} \leftarrow \mathbb{T} \setminus \{\min_{v \in \mathbb{T}} v\}$.
    end if
    Adjust $\mathbf{x}_t^*$ using bounding box regression.
  end if
  if $\mathcal{F}_{t,+}(\mathbf{x}_t^*) < 0.5$ **or** $t \bmod 10 = 0$ **then**
    Select $\Theta' \subseteq \Theta$ for model update using Eq. (1).
    Update Θ' using $\mathbb{S}_{v \in \mathbb{T}}^+$ and $\mathbb{S}_{v \in \mathbb{T}}^-$ by Eq. (3).
  end if
until end of sequence

When searching for target in each frame, we draw $N = 256$ samples for observation. We enlarge search space wildly if classification scores from CNN are below the predefined threshold for more than 10 frames in a row. If target appearance model update is required, a subset of branches are selected based on α_k from a Bernoulli distribution with $p_k = 0.5$. The size of a mini-batch is 128, which includes 36 positive examples and 92 negative examples. For online learning, 30 iterations is performed with learning rate 0.0001 and the momentum and weight decay are set to 0.9 and 0.0005, respectively. Overall procedure of our algorithm is presented in Algorithm 1.

5. Experiments

We show the performance of the BranchOut technique with ensemble tracking application on two standard public benchmarks—Object Tracking Benchmark (OTB100) [39] and VOT2015 [22], and compare our algorithm with the state-of-the-art trackers.

5.1. Evaluation on OTB

OTB100 [39] is a popular benchmark dataset, which contains 100 fully annotated videos with substantial variations and challenges. Two evaluation metrics are employed in our experiment: bounding box overlap ratio and center location error in the one-pass evaluation (OPE) protocol.

External comparison Our algorithm, denoted by BranchOut, is compared with another nine competitive tracking methods including C-COT [10], TCNN [29], DeepSRDCF [8], HCF [28], CNN-SVM [16], MUSTer [17], FCNT [36], DSST [7] and SRDCF [9]. All methods except MUSTer, DSST and SRDCF are based on the features from convolutional neural networks.

Figure 2(a) illustrates the overall success and precision plots based on bounding box overlap ratio and center location error, respectively. It illustrates that BranchOut outperforms the state-of-the-art trackers in both measures The performance of BranchOut is as competitive as (or even better than) MDNet [30], which requires pretraining process with external tracking sequences to achieve the best performance. Note that MDNet shows 0.678 and 0.909 in success and precision plot, respectively, as shown in Table 2.

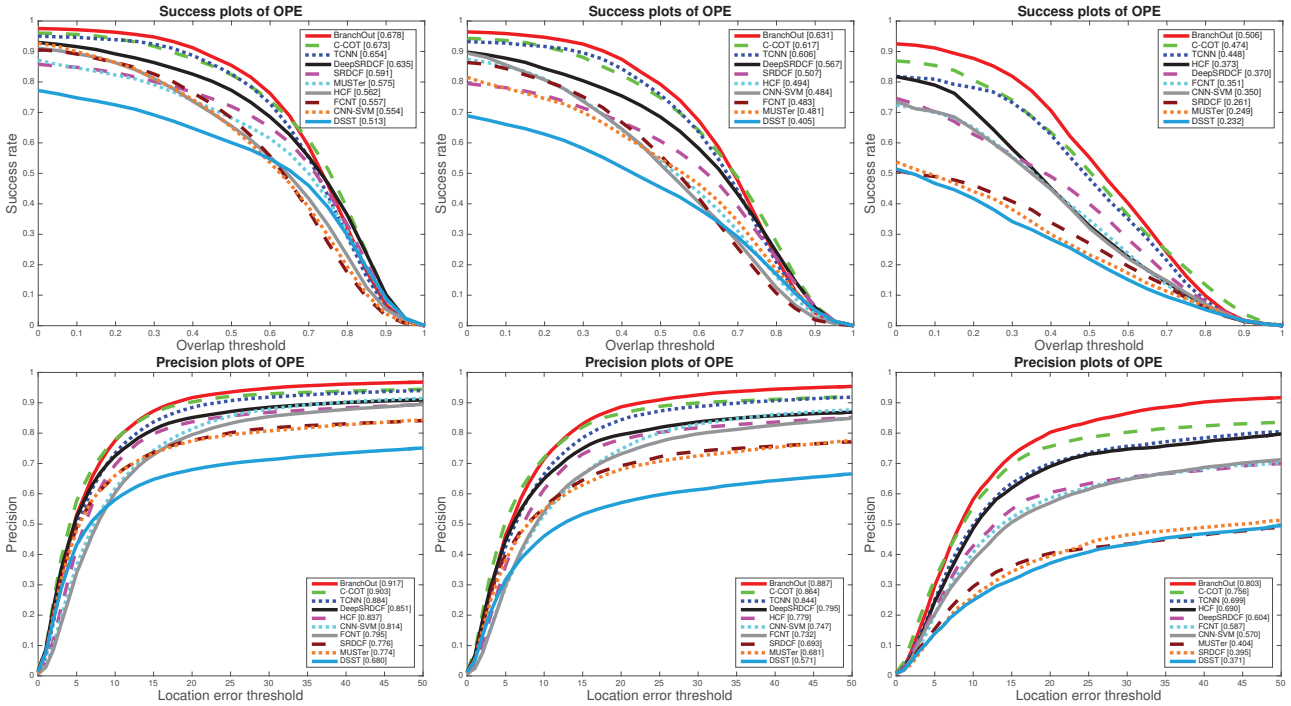
In addition to the standard visualization of tracker performance, we also illustrate how each tracker performs on more challenging subsets of video sequences. This information would be useful to analyze tracker performance because the recent state-of-the-art trackers are almost equally accurate in most of easy sequences and their results based on all sequences in the dataset often fail to show performance in more realistic situations. Hence, we construct two subsets of OTB100 based on average accuracy of the 10 compared algorithms; the two subsets are composed of the sequences that have lower average bounding box overlap ratios than two predefined thresholds, 0.7 and 0.5. These two subsets include 69 and 21 sequences, and can be regarded as hard[1] and very hard examples[2]. As illustrated in Figure 2(b) and 2(c), the gaps between our algorithm and the others are more noticeable. Figure 3 presents success plots for individual challenge attributes. As illustrated in the figure, BranchOut is robust to all challenges consistently.

Ablation experiment To verify the contribution of each component in our algorithm, we implement and evaluate several variations of our approach. The effectiveness of our stochastic ensemble strategy is tested by comparing with two options—a naïve deterministic ensemble and a greedy BranchOut ensemble.

In the naïve ensemble approach, all branches are trained using the same training examples whenever model update

[1]Hard sequences: *basketball, biker, bird1, blurBody, blurOwl, board, bolt2, bolt, box, car1, car24, carScale, clifBar, coke, couple, coupon, crowds, diving, dog, dragonBaby, fleetface, football1, football, freeman1, freeman3, freeman4, girl2, girl, gym, human2, human3, human4, human5, human6, human7, human8, human9, ironman, jogging-1, jogging-2, jump, jumping, kiteSurf, lemming, matrix, motorRolling, panda, redTeam, rubik, shaking, singer1, singer2, skater2, skater, skating1, skating2-1, skating2-2, skiing, soccer, subway, surfer, tiger1, tiger2, toy, trans, twinnings, vase, walking2, walking*

[2]Very hard sequences: *biker, bird1, bolt2, clifBar, diving, dog, girl2, gym, human3, human9, ironman, jump, matrix, motorRolling, panda, skating1, skating2-1, skating2-2, skiing, soccer, vase*



(a) All sequences (b) Hard sequences (c) Very hard sequences

Figure 2. Tracking results in OTB100 dataset. (a) Comparisons with the state-of-the-art algorithms based on all 100 videos. (b) Comparisons with other competitive algorithms based on the sequences below average overlap ratio 0.7. (c) Comparisons with other competitive algorithms based on the sequences below average overlap ratio 0.5. Note that the gaps between BranchOut and other methods get larger as easier sequences are disregarded.

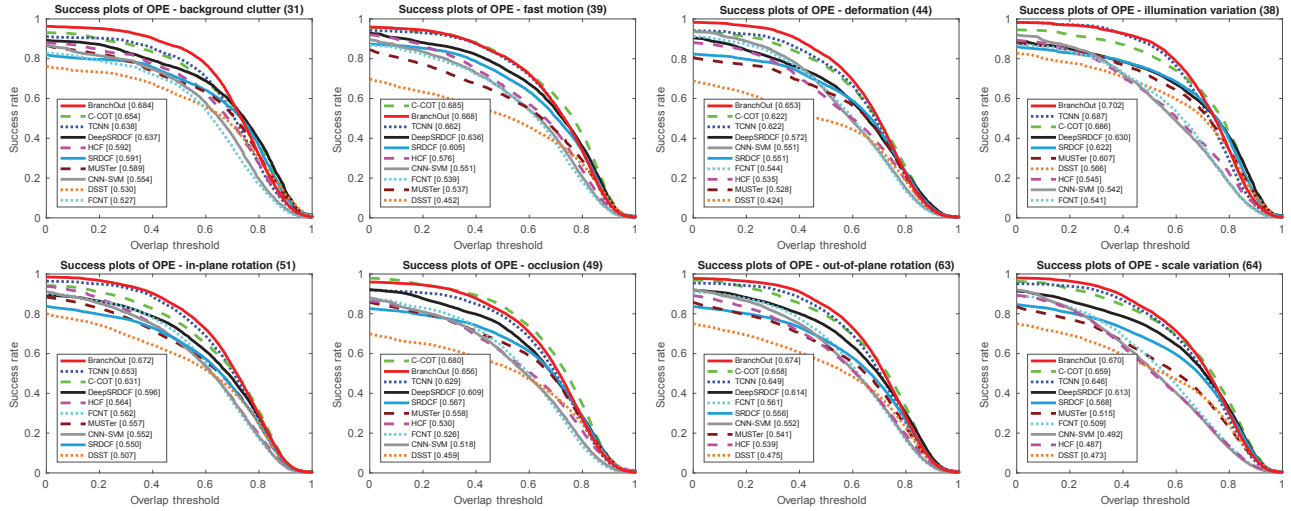


Figure 3. Tracking results in OTB100 dataset for 8 challenge attributes: background clutter, fast motion, deformation, illumination variation, in-plane rotation, occlusion, out-of-plane rotation, and scale variation. BranchOut outperforms other algorithms in most cases.

module is triggered. This approach generates homogeneous CNN models in all branches and may not be effective to handle the variations imposed on target and background. On the other hand, the greedy BranchOut specializes individual branches by giving more chance to the ones with higher target scores. This strategy is motivated by stochastic multiple choice learning [25], but it turns out that achieving better performance is not straightforward and the strategy is probably too sensitive to parameter setting. The detailed results are presented in Table 1. Compared to the other two ensemble methods, our stochastic ensemble based on BranchOut improves tracker performance.

Table 1. Internal comparison results in OTB100 dataset. Among three ensemble learning options, our stochastic BranchOut technique outperforms *Naïve* ensemble and *Greedy* BranchOut methods. On the other hand, multi-level representations for BranchOut, denoted by Multi-5-5, is helpful to improve performance compared to single-level representations such as Single-0-10 and Single-10-0. Fonts in bold faces denote the best performance within the group of options.

		All			@0.7			@0.5		
		Success (AUC)	Success (mOR)	Precision @20px	Success (AUC)	Success (mOR)	Precision @20px	Success (AUC)	Success (mOR)	Precesion @20px
BranchOut (with Multi5-5)		**0.678**	**0.688**	**0.917**	**0.631**	**0.639**	**0.887**	**0.506**	**0.508**	**0.803**
Learning options	Naive	0.661	0.670	0.890	0.604	0.611	0.846	0.465	0.467	0.760
	Greedy	0.658	0.668	0.887	0.602	0.610	0.844	0.456	0.458	0.745
Multi-level representation	Single-10-0	0.667	0.677	0.901	0.615	0.622	0.864	0.498	0.500	0.780
	Single-0-10	0.651	0.660	0.880	0.590	0.600	0.832	0.454	0.455	0.729

Table 2. The accuracy of MDNet ensemble with BranchOut. Our ensemble tracking algorithm outperforms the original MDNet and its naïve ensemble results.

Tracker	Success (AUC)	Precision (@20px)
MDNet–BranchOut	**0.683**	**0.919**
MDNet [30]	0.678	0.909
MDNet–Naïve	0.674	0.905

We evaluated the benefit of multi-level representations of target, and constructed CNNs with 10 branches to test performance of BranchOut technique with two kinds of single-level representations. One is with one FC layer per branch followed by the common CONV1-3 layers, and the other is with two FC layers per branch. These options are denoted by Single-10-0 and Single-0-10, respectively, where the numbers indicate the number of branches with two different depths in the FC layers. Note that our BranchOut is based on Multi-5-5, which has 5 branches with one FC layers and another 5 branches with two FC layers. According to our observation, more than two FC layers are not helpful in terms of accuracy and incur too much computational cost because additional layers increase the number of parameters and need more iterations for convergence. The use of multi-level representation does not improve performance significantly, but Table 1 presents clearly visible benefit.

Each component in our tracking algorithm—stochastic ensemble learning and mult-level representation—is helpful to improve performance. This advantage is observed more clearly in challenging sequences of the dataset as shown in the columns corresponding to predefined average overlap ratio threshold 0.7 and 0.5.

We employ BranchOut for ensemble of MDNet [30] to present its generality. We create 5 branches with two FC layers, and pretrain the network using multi-domain learning with BranchOut technique. For online tracking, we re-initialize all the 5 branches randomly, and perform tracking by our BranchOut and the naïve ensemble for comparison. Table 2 illustrates the results; BranchOut turns out to be helpful to improve MDNet-based ensemble tracking algorithm (MDNet–BranchOut) while deterministic MDNet–Naïve suffers from lack of representation diversity.

Qualitative results We illustrate the qualitative evaluation results in several challenging sequences in Figure 4, which show how robust our tracking algorithm is to a variety of realistic challenges. However, the proposed algorithm sometimes fails. Figure 5 demonstrates that it suffers from full occlusion in *Bird1* sequence and significant appearance changes with large motion in *Matrix* sequence.

5.2. Evaluation on VOT2015

BranchOut is also evaluated on VOT2015 dataset [22], which contains 60 sequences with substantial variations and challenges. We follow the VOT challenge protocol to compare tracking algorithms, where trackers are re-initialized whenever a tracking failure is observed. Based on bounding box overlap ratios wth ground-truths and the number of re-initializations, accuracy and robustness of each tracking algorithm is computed for comparison, respectively. In addition, VOT2015 reports the expected average overlap, which is a combined measure of accuracy and robustness, and ranks the trackers based on the measure.

Table 3 presents the results of VOT2015 dataset for 12 trackers including BranchOut (ours), DeepSRDCF [9], EBT [40], SRDCF [9], LDP [22], C-COT [10], sPST [18], SC-EBT [38], NSAMF [27], Struck [14], sPST [18] and RAJSSC [22]. We obtain the results of the algorithm from the official VOT Challenge website[3] or the authors of the corresponding papers. TCNN and BranchOut demonstrate outstanding scores and ranks while the performance C-COT in VOT2015 dataset is surprisingly low; it is probably because the algorithm is overfitted to other datasets.

6. Conclusion

We presented a novel visual tracking algorithm based on stochastic ensemble learning based on a CNN with multiple branches. Our ensemble tracking algorithm selects a random subset of branches for model update to diversify learned target appearance models. This technique, referred to as BranchOut, is effective to regularize ensemble classifiers and improve tracking accuracy consequently. We also

[3] http://www.votchallenge.net/

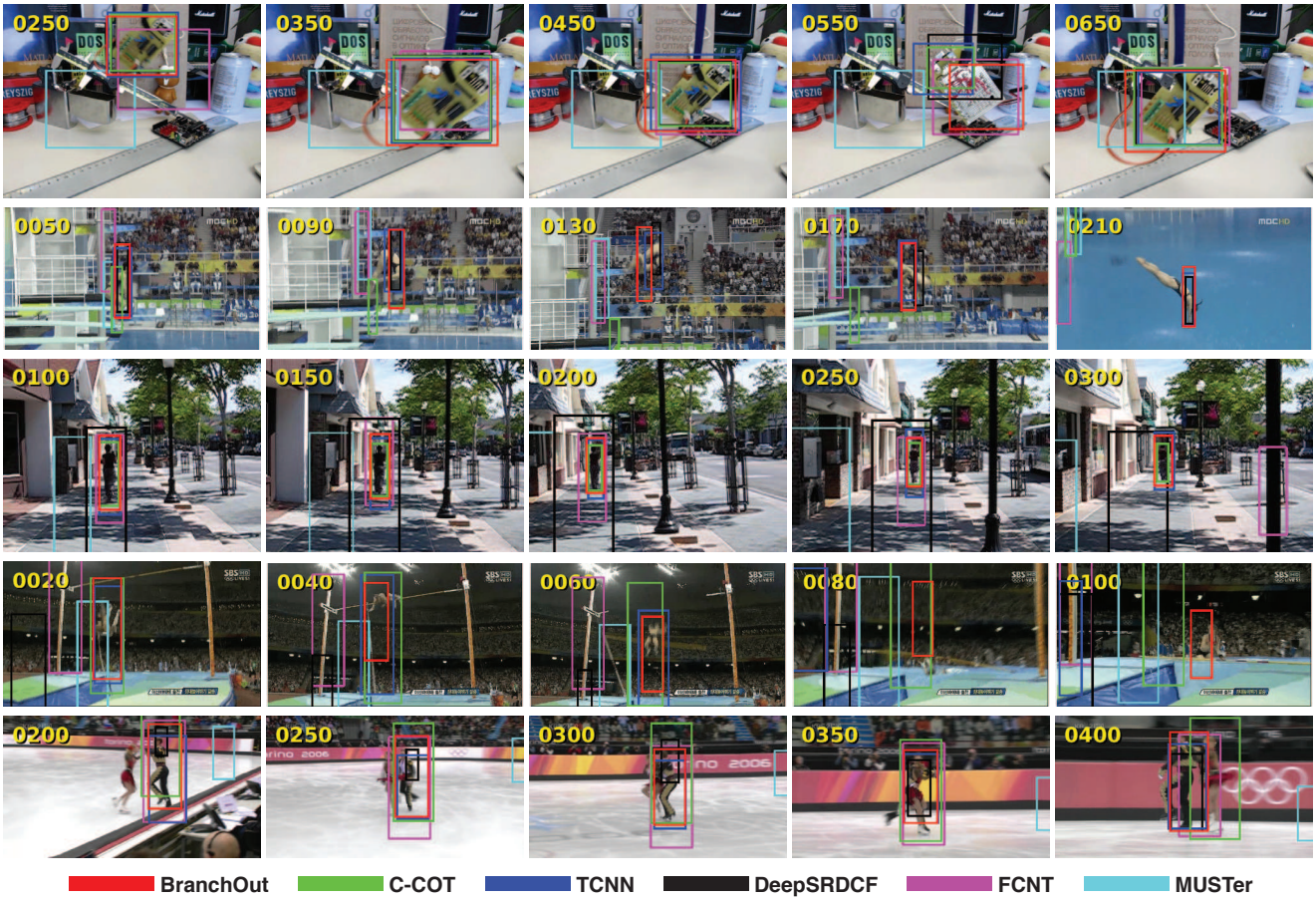


Figure 4. Qualitative comparisons of the proposed algorithm with several algorithms on some challenging sequences in OTB100: *Board*, *Diving*, *Human9*, *Jump*, and *Skating2-2*.

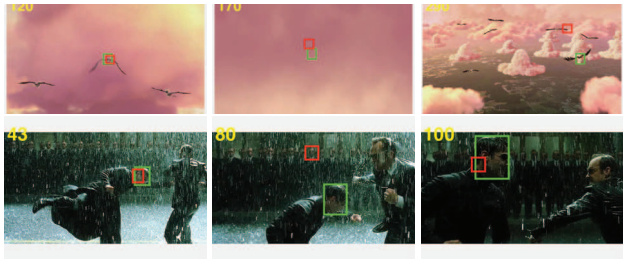


Figure 5. Failure cases of our method in *Bird1* and *Matrix* sequences. Green and red bounding boxes denote ground-truths and our tracking results, respectively. BranchOut sometimes loses targets due to occlusion and significant appearance changes.

employed multi-level representations for effective target appearance modeling. The proposed algorithm showed outstanding performance in the standard tracking benchmarks.

Acknowledgement

This work was performed while the first author was a visiting researcher at Google, Venice, CA. This work was partly supported by the ICT R&D program of MSIP/IITP [2014-0-00147, Machine Learning Center; 2014-0-00059, DeepView; 2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion].

Table 3. The experimental results in VOT2015 dataset. The first and second best algorithms are highlighted in red and blue colors, respectively. The algorithms are sorted in a decreasing order based on expected overlap ratio.

Tracker	Accuracy		Robustness		Expected
	Rank	Score	Rank	Score	Overlap
BranchOut	1.73	0.59	2.73	0.71	0.3384
TCNN [29]	1.57	0.59	4.05	0.74	0.3404
DeepSRDCF [8]	2.28	0.56	3.02	1.05	0.3181
EBT [40]	5.92	0.47	2.93	1.02	0.3130
C-COT [10]	2.92	0.54	3.02	0.82	0.3034
SiamFC-3s [2]	2.82	0.55	4.98	1.58	0.2915
SRDCF [9]	2.67	0.56	3.55	1.24	0.2877
LDP [22]	4.33	0.49	4.73	1.33	0.2785
sPST [18]	2.82	0.55	4.88	1.48	0.2767
NSAMF [27]	3.17	0.53	4.17	1.29	0.2536
Struck [14]	5.10	0.47	5.18	1.61	0.2458
RAJSSC [22]	2.10	0.57	5.22	1.63	0.2420

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 2

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *arXiv:1606.09549*, 2016. 8

[3] D. S. Bolme, J. R. Beveridge, B. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 2, 3

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 4

[6] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. In *ECCV*, 2012. 1

[7] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 2, 5

[8] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCVW*, 2015. 2, 5, 8

[9] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 2, 5, 7, 8

[10] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 2, 5, 7, 8

[11] H. K. Galoogahi, T. Sim, and S. Lucey. Correlation filters with limited boundaries. In *CVPR*, 2015. 2

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 4

[13] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006. 2

[14] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 2, 7, 8

[15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 2

[16] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015. 2, 5

[17] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. MUlti-Store Tracker (MUSTer): a cognitive psychology inspired approach to object tracking. In *CVPR*, 2015. 2, 5

[18] Y. Hua, K. Alahari, and C. Schmid. Online object tracking with proposal selection. In *ICCV*, 2015. 7, 8

[19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 3

[20] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. 2

[21] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 1

[22] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernandez, T. Vojir, G. Häger, G. Nebehay, R. Pflugfelder, et al. The visual object tracking VOT2015 challenge results. In *ICCVW*, pages 564–586, 2015. 5, 7, 8

[23] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 38, pages 2137–2155, 2016.

[24] S. Kwak, B. Han, and J. H. Han. Multi-agent event detection: Localization and role assignment. In *CVPR*, 2013. 1

[25] S. Lee, S. Purushwalkam, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *NIPS*, 2016. 3, 6

[26] H. Li, Y. Li, and F. Porikli. Convolutional neural net bagging for online visual tracking. *Computer Vision and Image Understanding*, 153:120–129, 2016. 2

[27] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCVW*, 2014. 7, 8

[28] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 2, 5

[29] H. Nam, M. Baek, and B. Han. Modeling and propagating cnns in a tree structure for visual tracking. In *arXiv:1608.07242*, 2016. 2, 5, 8

[30] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 2, 4, 5, 7

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 4

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 1, 3

[33] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. 3

[34] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural network using dropconnect. In *ICML*, 2013. 1, 3

[35] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 1

[36] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 2, 5

[37] L. Wang, W. Ouyang, X. Wang, and H. Lu. STCT: sequentially training convolutional networks for visual tracking. In *CVPR*, 2016. 2, 3

[38] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *ICML*, 2014. 7

[39] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 5

[40] G. Zhu, F. Porikli, and H. Li. Tracking randomly moving objects on edge box proposals. arXiv:1507.08085, 2015. 7, 8