Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

Hyeonseob Nam   Bohyung Han
Dept. of Computer Science and Engineering, POSTECH, Korea
{namhs09, bhhan}@postech.ac.kr

Abstract

We propose a novel visual tracking algorithm based on the representations from a discriminatively trained Convolutional Neural Network (CNN). Our algorithm pretrains a CNN using a large set of videos with tracking groundtruths to obtain a generic target representation. Our network is composed of shared layers and multiple branches of domain-specific layers, where domains correspond to individual training sequences and each branch is responsible for binary classification to identify target in each domain. We train each domain in the network iteratively to obtain generic target representations in the shared layers. When tracking a target in a new sequence, we construct a new network by combining the shared layers in the pretrained CNN with a new binary classification layer, which is updated online. Online tracking is performed by evaluating the candidate windows randomly sampled around the previous target state. The proposed algorithm illustrates outstanding performance in existing tracking benchmarks.

1. Introduction

Convolutional Neural Networks (CNNs) have recently been applied to various computer vision tasks such as image classification [4, 28, 37], semantic segmentation [19, 31, 33], object detection [12], and many others [34, 40, 41]. Such great success of CNNs is mostly attributed to their outstanding performance in representing visual data. Visual tracking, however, has been less affected by these popular trends since it is difficult to collect a large amount of training data for video processing applications and training algorithms specialized for visual tracking are not available yet, while the approaches based on low-level handcraft features still work well in practice [5, 17, 21, 47]. Several recent tracking algorithms [20, 44] have addressed the data deficiency issue by transferring pretrained CNNs on a large-scale classification dataset such as ImageNet [36]. Although these methods may be sufficient to obtain generic feature representations, its effectiveness in terms of tracking is limited due to the fundamental inconsistency between classification and tracking problems, *i.e.*, predicting object class labels versus locating targets of arbitrary classes.

To fully exploit the representation power of CNNs in visual tracking, it is desirable to train them on large-scale data specialized for visual tracking, which cover a wide range of variations in the combination of target and background. However, it is truly challenging to learn a unified representation based on the video sequences that have completely different characteristics. Note that individual sequences involve different types of targets whose class labels, moving patterns, and appearances are different, and tracking algorithms suffer from sequence-specific challenges including occlusion, deformation, lighting condition change, motion blur, etc. Training CNNs is even more difficult since the same kind of objects can be considered as a target in a sequence and as a background object in another. Due to such variations and inconsistencies across sequences, we believe that the ordinary learning methods based on the standard classification task are not appropriate, and another approach to capture sequence-independent information should be incorporated for better representation.

Motivated by this fact, we propose a novel CNN architecture, referred to as *Multi-Domain Network* (MDNet), to learn the shared representation of targets from multiple annotated video sequences for tracking, where each video is regarded as a separate domain. The proposed network has separate branches of domain-specific layers for binary classification at the end of the network, and shares the common information captured from all sequences in the preceding layers for generic representation learning. Each domain in MDNet is trained separately and iteratively while the shared layers are updated in every iteration. By employing this strategy, we separate domain-independent information from domain-specific one and learn generic feature representations for visual tracking. Another interesting aspect of our architecture is that we design the CNN with a small number of layers compared to the networks for classification tasks such as AlexNet [28] and VGG nets [4, 37].

We also propose an effective online tracking framework based on the representations learned by MDNet. When a test sequence is given, all the existing branches of binary classification layers, which were used in the training phase, are removed and a new single branch is constructed to compute target scores in the test sequence. The new classification layer and the fully connected layers within the shared layers are then fine-tuned online during tracking to adapt to the new domain. The online update is conducted to model long-term and short-term appearance variations of a target for robustness and adaptiveness, respectively, and an effective and efficient hard negative mining technique is incorporated in the learning procedure.

Our algorithm consists of multi-domain representation learning and online visual tracking. The main contributions of our work are summarized below:

- We propose a multi-domain learning framework based on CNNs, which separates domain-independent information from domain-specific one, to capture shared representations effectively.

- Our framework is successfully applied to visual tracking, where the CNN pretrained by multi-domain learning is updated online in the context of a new sequence to learn domain-specific information adaptively.

- Our extensive experiment demonstrates the outstanding performance of our tracking algorithm compared to the state-of-the-art techniques in two public benchmarks: Object Tracking Benchmark [45] and VOT2014 [26].

The rest of the paper is organized as follows. We first review related work in Section 2, and discuss our multi-domain learning approach for visual tracking in Section 3. Section 4 describes our online learning and tracking algorithm, and Section 5 demonstrates the experimental results in two tracking benchmark datasets.

2. Related Work

2.1. Visual Tracking Algorithms

Visual tracking is one of the fundamental problems in computer vision and has been actively studied for decades. Most tracking algorithms fall into either generative or discriminative approaches. Generative methods describe the target appearances using generative models and search for the target regions that fit the models best. Various generative target appearance modeling algorithms have been proposed including sparse representation [32, 48], density estimation [15, 22], and incremental subspace learning [35]. In contrast, discriminate methods aim to build a model that distinguishes the target object from the background. These tracking algorithms typically learn classifiers based on multiple instance learning [1], P-N learning [24], online boosting [13, 14, 38], structured output SVMs [16], etc.

In recent years, correlation filters have gained attention in the area of visual tracking due to their computational efficiency and competitive performance [2, 17, 5, 21]. Bolme *et al.* [2] proposed a fast correlation tracker with a minimum output sum of squared error (MOSSE) filter, which runs in hundreds of frames per second. Henriques *et al.* [17] formulated kernelized correlation filters (KCF) using circulant matrices, and efficiently incorporated multi-channel features in a Fourier domain. Several variations of KCF tracker have been subsequently investigated to improve tracking performance. For example, DSST [5] learns separate filters for translation and scaling, and MUSTer [21] employs short-term and long-term memory stores inspired by a psychological memory model. Although these approaches are satisfactory in constrained environments, they have an inherent limitation that they resort to low-level hand-crafted features, which are vulnerable in dynamic situations including illumination changes, occlusion, deformations, etc.

2.2. Convolutional Neural Networks

CNNs have demonstrated their outstanding representation power in a wide range of computer vision applications [12, 28, 31, 34, 37, 41, 40, 43]. Krizhevsky *et al.* [28] brought significant performance improvement in image classification by training a deep CNN with a large-scale dataset and an efficient GPU implementation. R-CNN [12] applies a CNN to an object detection task, where the training data are scarce, by pretraining on a large auxiliary dataset and fine-tuning on the target dataset.

Despite such huge success of CNNs, only a limited number of tracking algorithms using the representations from CNNs have been proposed so far [9, 20, 29, 44]. An early tracking algorithm based on a CNN can handle only predefined target object classes, *e.g.*, human, since the CNN is trained offline before tracking and fixed afterwards [9]. Although [29] proposes an online learning method based on a pool of CNNs, it suffers from lack of training data to train deep networks and its accuracy is not particularly good compared to the methods based on hand-craft features. A few recent approaches [44, 20] transfer CNNs pretrained on a large-scale dataset constructed for image classification, but the representation may not be very effective due to the fundamental difference between classification and tracking tasks. Contrary to the existing approaches, our algorithm takes advantage of large-scale visual tracking data for pretraining a CNN and obtain effective representations.

2.3. Multi-Domain Learning

Our approach to pretrain deep CNNs belongs to multi-domain learning, which refers to a learning method in which the training data are originated from multiple domains and the domain information is incorporated in learning procedure. Multi-domain learning is popular in natural lan-

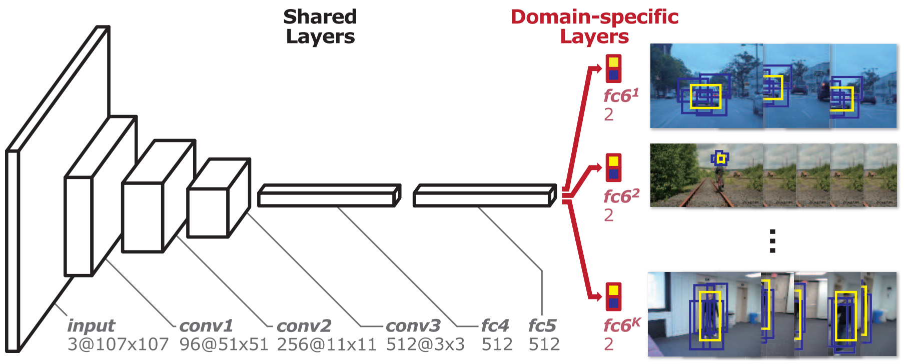


Figure 1: The architecture of our Multi-Domain Network, which consists of shared layers and K branches of domain-specific layers. Yellow and blue bounding boxes denote the positive and negative samples in each domain, respectively.

guage processing (*e.g.*, sentiment classification with multiple products and spam filtering with multiple users), and various approaches have been proposed [6, 7, 23]. In computer vision community, multi-domain learning is discussed in only a few domain adaptation approaches. For example, Duan *et al.* [8] introduced a domain-weighted combination of SVMs for video concept detection, and Hoffman *et al.* [18] presented a mixture-transform model for object classification.

3. Multi-Domain Network (MDNet)

This section describes our CNN architecture and multi-domain learning approach to obtain domain-independent representations for visual tracking.

3.1. Network Architecture

The architecture of our network is illustrated in Figure 1. It receives a 107×107 RGB input[1], and has five hidden layers including three convolutional layers (conv1-3) and two fully connected layers (fc4-5). Additionally, the network has K branches for the last fully connected layers (fc6^1-fc6^K) corresponding to K domains, in other words, training sequences. The convolutional layers are identical to the corresponding parts of VGG-M network [4] except that the feature map sizes are adjusted by our input size. The next two fully connected layers have 512 output units and are combined with ReLUs and dropouts. Each of the K branches contains a binary classification layer with softmax cross-entropy loss, which is responsible for distinguishing target and background in each domain. Note that we refer to fc6^1-fc6^K as domain-specific layers and all the preceding layers as shared layers.

Our network architecture is substantially smaller than the ones commonly used in typical recognition tasks such as AlexNet [28] and VGG-Nets [4, 37]. We believe that such a simple architecture is more appropriate for visual tracking due to the following reasons. First, visual tracking aims to distinguish only two classes, target and background, which requires much less model complexity than general visual recognition problems such as ImageNet classification with 1000 classes. Second, a deep CNN is less effective for precise target localization since the spatial information tends to be diluted as a network goes deeper [20]. Third, since targets in visual tracking are typically small, it is desirable to make input size small, which reduces the depth of the network naturally. Finally, a smaller network is obviously more efficient in visual tracking problem, where training and testing are performed online. When we tested larger networks, the algorithm is less accurate and becomes slower significantly.

[1]This input size is designed to obtain 3×3 feature maps in conv3: $107 = 75$ (receptive field) $+ 2 \times 16$ (stride).

3.2. Learning Algorithm

The goal of our learning algorithm is to train a multi-domain CNN disambiguating target and background in an arbitrary domain, which is not straightforward since the training data from different domains have different notions of target and background. However, there still exist some common properties that are desirable for target representations in all domains, such as robustness to illumination changes, motion blur, scale variations, etc. To extract useful features satisfying these common properties, we separate domain-independent information from domain-specific one by incorporating a multi-domain learning framework.

Our CNN is trained by the Stochastic Gradient Descent (SGD) method, where each domain is handled exclusively in each iteration. In the k^{th} iteration, the network is updated based on a minibatch that consists of the training samples from the $(k \bmod K)^{\text{th}}$ sequence, where only a single branch fc6$^{(k \bmod K)}$ is enabled. It is repeated until the network is converged or the predefined number of iterations is reached. Through this learning procedure, domain-independent information is modeled in the shared layers from which useful generic feature representations are obtained.

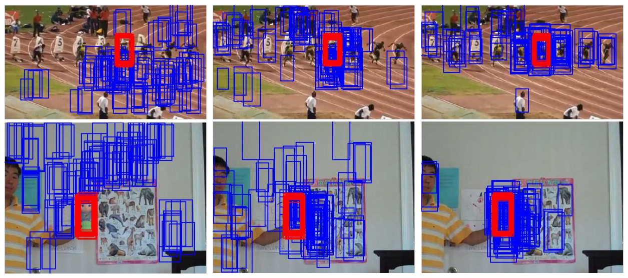

(a) 1st minibatch (b) 5th minibatch (c) 30th minibatch

Figure 2: Identified training examples through our hard negative mining in *Bolt2* (top) and *Doll* (bottom) sequences. Red and blue bounding boxes denote positive and negative samples in each minibatch, respectively. The negative samples becomes hard to classify as training proceeds.

4. Online Tracking using MDNet

Once we complete the multi-domain learning described in Section 3.2, the multiple branches of domain-specific layers (fc6^1-fc6^K) are replaced with a single branch (fc6) for a new test sequence. Then we fine-tune the new domain-specific layer and the fully connected layers in the shared network online at the same time. The detailed tracking procedure is discussed in this section.

4.1. Tracking Control and Network Update

We consider two complementary aspects in visual tracking, robustness and adaptiveness, by long-term and short-term updates. Long-term updates are performed in regular intervals using the positive samples collected for a long period while short-term updates are conducted whenever potential tracking failures are detected—when the positive score of the estimated target is less than 0.5—using the positive samples in a short-term period. In both cases we use the negative samples observed in the short-term since old negative examples are often redundant or irrelevant to the current frame. Note that we maintain a single network during tracking, where these two kinds of updates are performed depending on how fast the target appearance changes.

To estimate the target state in each frame, N target candidates $\mathbf{x}^1, \ldots, \mathbf{x}^N$ sampled around the previous target state are evaluated using the network, and we obtain their positive scores $f^+(\mathbf{x}^i)$ and negative scores $f^-(\mathbf{x}^i)$ from the network. The optimal target state $\mathbf{x}^*$ is given by finding the example with the maximum positive score as

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}^i} f^+(\mathbf{x}^i). \tag{1}$$

4.2. Hard Minibatch Mining

The majority of negative examples are typically trivial or redundant in tracking-by-detection approaches, while only a few distracting negative samples are effective to training a classifier. Hence, the ordinary SGD method, where the training samples evenly contribute to learning, easily suffers from a drift problem since the distractors are considered insufficiently. A popular solution in object detection for this issue is hard negative mining [39], where training and testing procedures are alternated to identify the hard negative examples, typically false positives, and we adopt this idea for our online learning procedure.

We integrate hard negative mining step into minibatch selection. In each iteration of our learning procedure, a minibatch consists of M^+ positives and M_h^- hard negatives. The hard negative examples are identified by testing $M^-(\gg M_h^-)$ negative samples and selecting the ones with top M_h^- scores. As the learning proceeds and the network becomes more discriminative, the classification in a minibatch becomes more challenging as illustrated in Figure 2. This approach examines a predefined number of samples and identifies critical negative examples effectively without explicitly running a detector to extract false positives as in the standard hard negative mining techniques.

4.3. Bounding Box Regression

Due to the high-level abstraction of CNN-based features and our data augmentation strategy which samples multiple positive examples around the target (which will be described in more detail in the next subsection), our network sometimes fails to find tight bounding boxes enclosing the target. We apply the bounding box regression technique, which is popular in object detection [10, 12], to improve target localization accuracy. Given the first frame of a test sequence, we train a simple linear regression model to predict the precise target location using conv3 features of the samples near the target location. In the subsequent frames, we adjust the target locations estimated from Eq. (1) using the regression model if the estimated targets are reliable (*i.e.* $f^+(x^*) > 0.5$). The bounding box regressor is trained only in the first frame since it is time consuming for online update and incremental learning of the regression model may not be very helpful considering its risk. Refer to [12] for details as we use the same formulation and parameters.

4.4. Implementation Details

The overall procedure of our tracking algorithm is presented in Algorithm 1. The filter weights in the j^{th} layer of CNN are denoted by $\mathbf{w}_j$, where $\mathbf{w}_{1:5}$ are pretrained by mutli-domain learning and $\mathbf{w}_6$ is initialized randomly for a new sequence. Only the weights in the fully connected layers $\mathbf{w}_{4:6}$ are updated online whereas the ones in the convolutional layers $\mathbf{w}_{1:3}$ are fixed throughout tracking; this strategy is beneficial to not only computational efficiency but also avoiding overfitting by preserving domain-independent information. $\mathcal{T}_s$ and $\mathcal{T}_l$ are frame index sets in short-term ($\tau_s = 20$) and long-term ($\tau_l = 100$) periods, respectively.

Algorithm 1 Online tracking algorithm

```
Input  : Pretrained CNN filters $\{\mathbf{w}_1, \ldots, \mathbf{w}_5\}$
         Initial target state $\mathbf{x}_1$
Output: Estimated target states $\mathbf{x}_t^*$
 1: Randomly initialize the last layer $\mathbf{w}_6$.
 2: Train a bounding box regression model.
 3: Draw positive samples $S_1^+$ and negative samples $S_1^-$.
 4: Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S_1^+$ and $S_1^-$;
 5: $\mathcal{T}_s \leftarrow \{1\}$ and $\mathcal{T}_l \leftarrow \{1\}$.
 6: repeat
 7:     Draw target candidate samples $\mathbf{x}_t^i$.
 8:     Find the optimal target state $\mathbf{x}_t^*$ by Eq. (1).
 9:     if $f^+(\mathbf{x}_t^*) > 0.5$ then
10:         Draw training samples $S_t^+$ and $S_t^-$.
11:         $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{t\}$, $\mathcal{T}_l \leftarrow \mathcal{T}_l \cup \{t\}$.
12:         if $|\mathcal{T}_s| > \tau_s$ then $\mathcal{T}_s \leftarrow \mathcal{T}_s \setminus \{\min_{v \in \mathcal{T}_s} v\}$.
13:         if $|\mathcal{T}_l| > \tau_l$ then $\mathcal{T}_l \leftarrow \mathcal{T}_l \setminus \{\min_{v \in \mathcal{T}_l} v\}$.
14:         Adjust $\mathbf{x}_t^*$ using bounding box regression.
15:     if $f^+(\mathbf{x}_t^*) < 0.5$ then
16:         Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S_{v \in \mathcal{T}_s}^+$ and $S_{v \in \mathcal{T}_s}^-$.
17:     else if $t \bmod 10 = 0$ then
18:         Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S_{v \in \mathcal{T}_l}^+$ and $S_{v \in \mathcal{T}_s}^-$.
19: until end of sequence
```

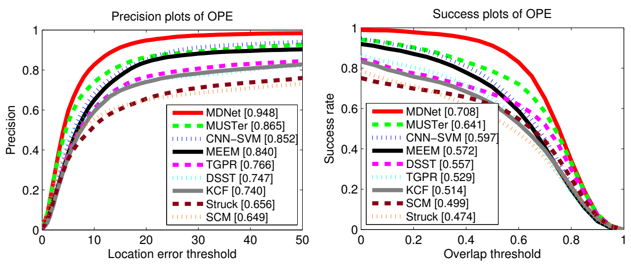
The further implementation details are described below.

Target candidate generation To generate target candidates in each frame, we draw $N(= 256)$ samples in translation and scale dimension, $\mathbf{x}_t^i = (x_t^i, y_t^i, s_t^i)$, $i = 1, \ldots, N$, from a Gaussian distribution whose mean is the previous target state $\mathbf{x}_{t-1}^*$ and covariance is a diagonal matrix $\text{diag}(0.09r^2, 0.09r^2, 0.25)$, where r is the mean of the width and height of the target in the previous frame. The scale of each candidate bounding box is computed by multiplying 1.05^{s_i} to the initial target scale.

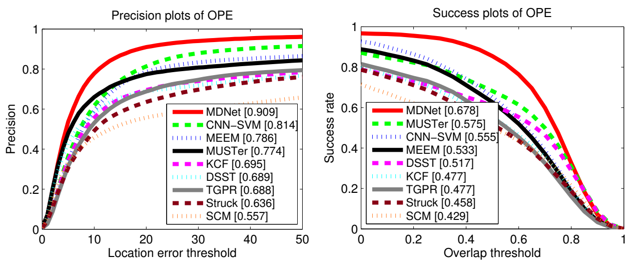
Training data For offline multi-domain learning, we collect 50 positive and 200 negative samples from every frame, where positive and negative examples have ≥ 0.7 and ≤ 0.5 IoU overlap ratios with ground-truth bounding boxes, respectively. Similarly, for online learning, we collect $S_t^+(= 50)$ positive and $S_t^-(= 200)$ negative samples with ≥ 0.7 and $\leq$ 0.3 IoU overlap ratios with the estimated target bounding boxes, respectively, except that $S_1^+ = 500$ and $S_1^- = 5000$. For bounding-box regression, we use 1000 training examples with the same parameters as [12].

Network learning For multi-domain learning with K training sequences, we train the network for $100K$ iterations with learning rates 0.0001 for convolutional layers[2] and 0.001 for fully connected layers. At the initial frame of a test sequence, we train the fully connected layers for 30 iterations with learning rate 0.0001 for fc4-5 and 0.001 for fc6. For online update, we train the fully connected layers for 10 iterations with the learning rate three times larger than that in the initial frame for fast adaptation. The momentum and weight decay are always set to 0.9 and 0.0005, respectively. Each mini-batch consists of $M^+(= 32)$ positives and $M_h^-(= 96)$ hard negatives selected out of $M^-(= 1024)$ negative examples.

[2]The convolutional layers are initialized by VGG-M network, which is pretrained on ImageNet.



(a) OTB50 result



(b) OTB100 result

Figure 3: Precision and success plots on OTB50 [46] and OTB100 [45]. The numbers in the legend indicate the representative precision at 20 pixels for precision plots, and the average area-under-curve scores for success plots.

5. Experiment

We evaluated MDNet on two datasets, Object Tracking Benchmark (OTB) [45] and VOT2014 [26]. Our algorithm is implemented in MATLAB using MatConvNet toolbox [42], and runs at around 1 fps with eight cores of 2.20GHz Intel Xeon E5-2660 and an NVIDIA Tesla K20m GPU. The source code of MDNet is available at `http://cvlab.postech.ac.kr/research/mdnet/`.

5.1. Evaluation on OTB

OTB [45] is a popular tracking benchmark that contains 100 fully annotated videos with substantial variations. The evaluation is based on two metrics: center location error and bounding box overlap ratio. The one-pass evaluation (OPE) is employed to compare our algorithm with the six state-of-the-art trackers including MUSTer [21], CNN-SVM [20], MEEM [47], TGPR [11], DSST [5] and KCF [17], as well as the top 2 trackers included in the benchmark—SCM [49]

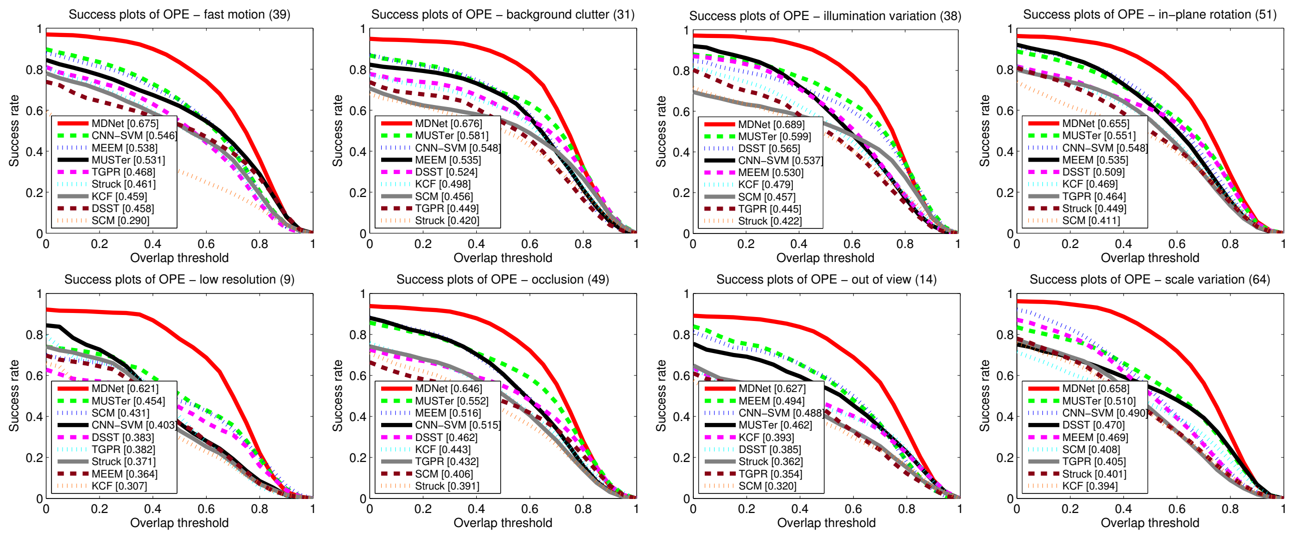


Figure 4: The success plots for eight challenge attributes: fast motion, background clutter, illumination variation, in-plain rotation, low resolution, occlusion, out of view, and scale variation.

and Struck [16]. Note that CNN-SVM is another tracking algorithm based on the representations from CNN, which provides a baseline for tracking algorithms that adopt deep learning. In addition to the results on the entire 100 sequences in [45] (OTB100), we also present the results on its earlier version containing 50 sequences [46] (OTB50). For offline training of MDNet, we use 58 training sequences collected from VOT2013 [27], VOT2014 [26] and VOT2015 [25], excluding the videos included in OTB100.

Figure 3 illustrates the precision and success plots based on center location error and bounding box overlap ratio, respectively. It clearly illustrates that our algorithm, denoted by MDNet, outperforms the state-of-the-art trackers significantly in both measures. The exceptional scores at mild thresholds means our tracker hardly misses targets while the competitive scores at strict thresholds implies that our algorithm also finds tight bounding boxes to targets. For detailed performance analysis, we also report the results on various challenge attributes in OTB100, such as occlusion, rotation, motion blur, etc. Figure 4 demonstrates that our tracker effectively handles all kinds of challenging situations that often require high-level semantic understanding. In particular, our tracker successfully track targets in low resolution while all the trackers based on low-level features are not successful in the challenge.

To verify the contribution of each component in our algorithm, we implement and evaluate several variations of our approach. The effectiveness of our multi-domain pretraining technique is tested by comparison with the single-domain learning method (SDNet), where the network is trained with a single branch using the data from multiple sequences. We also investigate two additional versions of our tracking algorithm—MDNet without bounding box regression (MDNet–BB) and MDNet without bounding box regression and hard negative mining (MDNet–BB–HM). The performances of all the variations are not as good as our full algorithm (MDNet) and each component in our tracking algorithm is helpful to improve performance. The detailed results are illustrated in Figure 5.

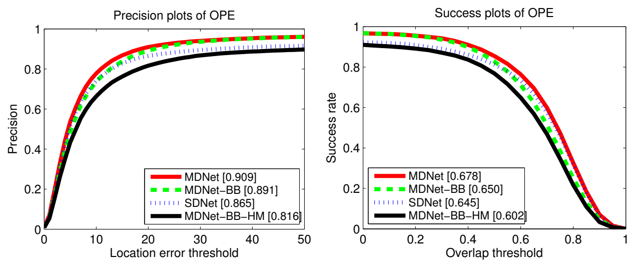


Figure 5: Precision and success plots on OTB100 for the internal comparisons.

Figure 6 presents the superiority of our algorithm qualitatively compared to the state-of-the-art trackers. Figure 7 shows a few failure cases of our algorithm; it has drift problem due to slow target appearance changes in *Coupon* sequence, and a sudden appearance change makes our tracker miss the target completely in *Jump* sequence.

5.2. Evaluation on VOT2014 Dataset

For completeness, we also present the evaluation results on VOT2014 dataset [26], which contains 25 sequences with substantial variations. In VOT challenge protocol, target is re-initialized whenever tracking fails and the evaluation module reports both accuracy and robustness, which correspond to the bounding box overlap ratio and the number of failures, respectively. There are two types of exper-

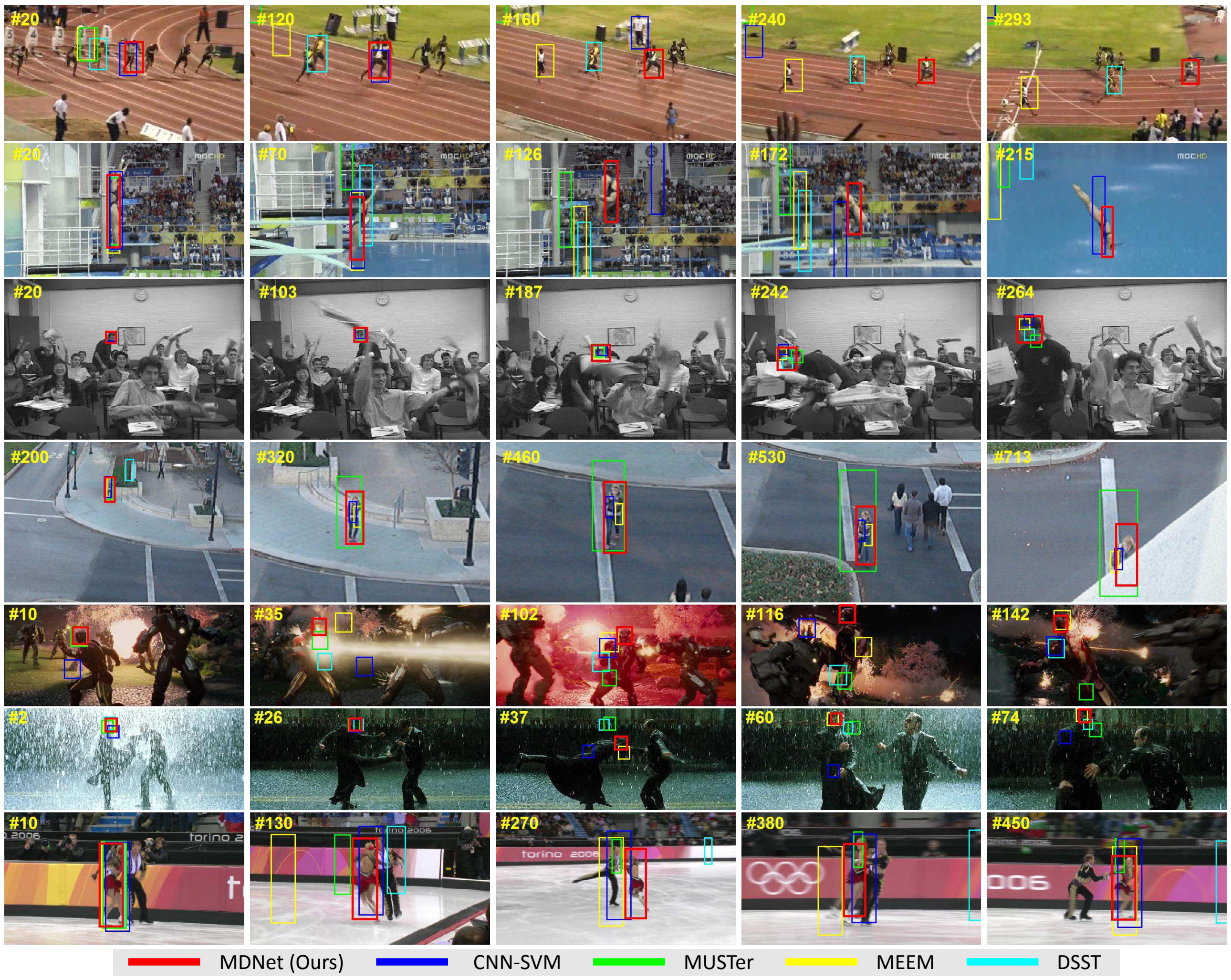


Figure 6: Qualitative results of the proposed method on some challenging sequences (*Bolt2*, *Diving*, *Freeman4*, *Human5*, *Ironman*, *Matrix* and *Skating2-1*).

Tracker	Accuracy		Robustness		Combined
	Score	Rank	Score	Rank	Rank
MUSTer	0.58	4.50	0.99	5.67	5.09
MEEM	0.48	7.17	0.71	5.50	6.34
DSST	0.60	4.03	0.68	5.17	4.60
SAMF	0.60	3.97	0.77	5.58	4.78
KCF	0.61	3.82	0.79	5.67	4.75
DGT	0.53	4.49	0.55	3.58	4.04
PLT_14	0.53	5.58	0.14	2.75	4.17
MDNet	0.63	2.50	0.16	2.08	2.29

(a) Baseline result

Tracker	Accuracy		Robustness		Combined
	Score	Rank	Score	Rank	Rank
MUSTer	0.55	4.67	0.94	5.53	5.10
MEEM	0.48	7.25	0.74	5.76	6.51
DSST	0.58	4.00	0.76	5.10	4.55
SAMF	0.57	3.72	0.81	4.94	4.33
KCF	0.58	3.92	0.87	4.99	4.46
DGT	0.54	3.58	0.67	4.17	3.88
PLT_14	0.51	5.43	0.16	2.08	3.76
MDNet	0.60	3.31	0.30	3.58	3.45

(b) Region_noise result

Table 1: The average scores and ranks of accuracy and robustness on the two experiments in VOT2014 [26]. The first and second best scores are highlighted in red and blue colors, respectively.

iment settings; trackers are initialized with either ground-truth bounding boxes (baseline) or randomly perturbed ones (region_noise). The VOT evaluation also provides a ranking analysis based on both statistical and practical significance of the performance gap between trackers. Please refer to [26] for more details. We compare our algorithm

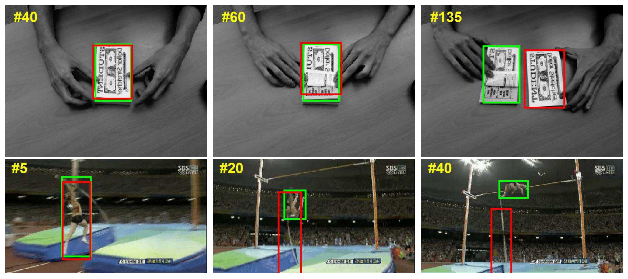
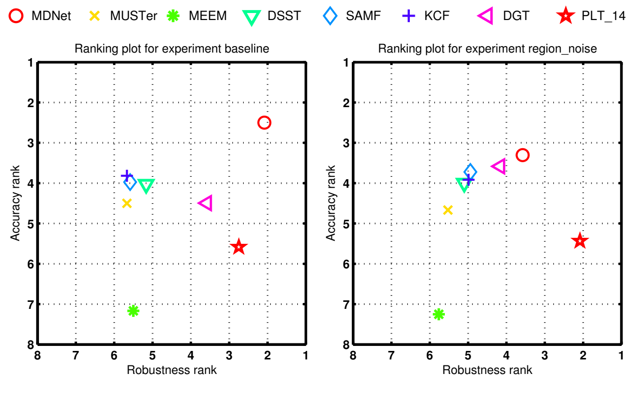


Figure 7: Failure cases of our method (*Coupon* and *Jump*). Green and red bounding boxes denote the ground-truths and our tracking results, respectively.



(a) Baseline (b) Region_noise

Figure 8: The robustness-accuracy ranking plots of tested algorithms in VOT2014 dataset. The better trackers are located at the upper-right corner.

with the top 5 trackers in VOT2014 challenge—DSST [5], SAMF [30], KCF [17], DGT [3] and PLT_14 [26]—and additional two state-of-the-art trackers MUSTer [21] and MEEM [47]. Our network is pretrained using 89 sequences from OTB100, which do not include the common sequences with the VOT2014 dataset.

As illustrated in Table 1 and Figure 8, MDNet is ranked top overall—the first place in accuracy and the first or second place in robustness; it demonstrates much better accuracy than all other methods, even with fewer re-initializations. Furthermore, MDNet works well with imprecise re-initializations as shown in the region_noise experiment results, which implies that it can be effectively combined with a re-detection module and achieve long-term tracking. We also report the results with respect to several visual attributes in Figure 9, which shows that our tracker is stable in various challenging situations.

6. Conclusion

We proposed a novel tracking algorithm based on CNN trained in a multi-domain learning framework, which is re-

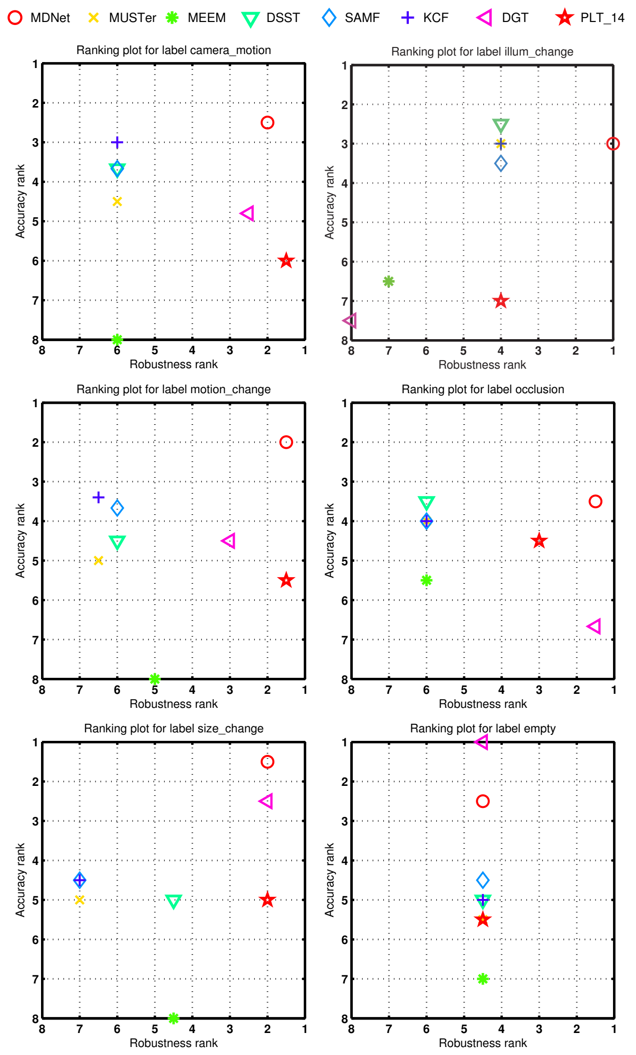


Figure 9: The robustness-accuracy ranking plots for five visual attributes (camera motion, illumination change, motion change, occlusion and size change) and an empty attribute.

ferred to as MDNet. Our tracking algorithm learns domain-independent representations from pretraining, and captures domain-specific information through online learning during tracking. The proposed network has a simple architecture compared to the one designed for image classification tasks. The entire network is pretrained offline, and the fully connected layers including a single domain-specific layer are fine-tuned online. We achieved outstanding performance in two large public tracking benchmarks, OTB and VOT2014, compared to the state-of-the-art tracking algorithms.

Acknowledgements

This work was partly supported by IITP grant (B0101-16-0307; Machine Learning Center, B0101-16-0552; DeepView) funded by the Korean government (MSIP).

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, 2011. 2

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

[3] Z. Cai, L. Wen, J. Yang, Z. Lei, and S. Z. Li. Structured visual tracking with dynamic graph. In *ACCV*, 2012. 8

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 1, 3

[5] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 1, 2, 5, 8

[6] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007. 3

[7] M. Dredze, A. Kulesza, and K. Crammer. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2):123–149, 2010. 3

[8] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, 2009. 3

[9] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *IEEE Trans. Neural Networks*, 21(10):1610–1623, 2010. 2

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010. 4

[11] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014. 5

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 4, 5

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

[14] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008. 2

[15] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1186–1197, 2008. 2

[16] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 2, 6

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):583–596, 2015. 1, 2, 5, 8

[18] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. Discovering latent domains for multisource domain adaptation. In *ECCV*, 2012. 3

[19] S. Hong, H. Noh, and B. Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *NIPS*, 2015. 1

[20] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015. 1, 2, 3, 5

[21] 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. 1, 2, 5, 8

[22] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1296–1311, 2003. 2

[23] M. Joshi, W. W. Cohen, M. Dredze, and C. P. Rosé. Multi-domain learning: when do domains matter? In *EMNLP-CoNLL*, 2012. 3

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

[25] 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*, 2015. 6

[26] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernandez, et al. The visual object tracking VOT2014 challenge results. In *ECCVW*, 2014. 2, 5, 6, 7, 8

[27] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernandez, T. Vojir, et al. The visual object tracking VOT2013 challenge results. In *ICCVW*, 2013. 6

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3

[29] H. Li, Y. Li, and F. Porikli. DeepTrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *BMVC*, 2014. 2

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

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[32] X. Mei and H. Ling. Robust visual tracking using ℓ_1 minimization. In *ICCV*, 2009. 2

[33] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1

[34] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*, 2016. 1, 2

[35] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008. 2

[36] 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. *IJCV*, pages 1–42, 2015. 1

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 3

[38] J. Son, I. Jung, K. Park, and B. Han. Tracking-by-segmentation with online gradient boosting decision tree. In *ICCV*, 2015. 2

[39] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):39–51, 1998. 4

[40] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 1, 2

[41] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1, 2

[42] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *ACM MM*, 2015. 5

[43] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 2

[44] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015. 1, 2

[45] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, Sept 2015. 2, 5, 6

[46] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 5, 6

[47] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. 1, 5, 8

[48] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012. 2

[49] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012. 5