Deep Ensemble Tracking

Jie Guo and Tingfa Xu

***Abstract*—In this letter, we cast visual tracking as a template matching problem in a Siamese deep convolutional neural network architecture. In contrast to traditional or other deep feature-based tracking methods, the proposed model exploits multilevel convolutional features from a partial view. The model matches candidate patch and template patch from the feature dimension of convolutional features, leading to hundreds of thousands of base matchers. The base matchers from low-level convolutional features have small receptive fields which contain partial details of targets while the base matchers from high-level convolutional features have big receptive fields which capture semantic information of targets. The model achieves the final strong matcher as a weighted ensemble of all the base matchers. We design an effective weights propagation strategy to update the weights of base matchers. Moreover, we propose to use Cosine as the distance metric and a customized squared-loss function as cost function for robust. Experiments show that our tracker outperforms the state-of-the-art trackers in a wide range of tracking scenarios.**

***Index Terms*—Convolutional neural network (CNN), ensemble tracking, Siamese neural network, template matching.**

I. Introduction

BOOSTING algorithm has been used to do visual tracking for many years. The advantages of boosting are its capability to select reliable features and do online learning. Grabner *et al*. [1] use boosting for online feature selection for robust visual tracking. Avidan [2] uses AdaBoost to combine a set of weak classifiers into a strong classifier. The strong classifier is then used to label pixels in the next frame, forming a confident map for further tracking task. To prevent drifting caused by online learning, Grabner *et al*. [3] propose a semisupervised update process by combining decision of a given prior and an online classifier. Furthermore, MIL [4] tracker adopts multiple instance learning strategy instead of traditional supervised learning to select weak classifiers. Zhang and Song [5] integrate the sample importance into the MIL framework, leading to a more robust and much faster tracker. However, in practice, the weak classifiers (with *low-level features*) are too unstable to control and the number of the weak classifiers is always limited. Thus, it is hard to achieve a strong effective tracker.

On the other hand, convolutional neural networks (CNNs) have the ability to abstract a large range level of features. The low-level convolutional features contain partial detail information and the high-level convolutional features contain more semantic information. Wang *et al.* [6] combine deep CNN features with correlation filters, forming a coarse-to-fine search tracker. In their method, they use the last layers to handle large appearance changes and use earlier layers for precise localization. Tao *et al.* [7] use a Siamese deep neural network to learn a general matching function for their tracker. Bertinetto *et al*. [8] design a novel fully convolutional Siamese network trained end-to-end on the ILSVRC15 dataset for object detection in video. However, both of the two Siamese networks are only trained offline without any online update of parameters or template. Although they are trained on numerous videos, considering the complex appearance changes of any unknown target, the lack of online adapting may cause tracking failure.

Motivated by the above observations, we present a novel deep ensemble tracking (DET) which combines boosting algorithm (for online adapting) with Siamese deep CNN (for robust base matchers). The proposed model matches the candidates with the first frame ground-truth template and finds the optimal candidate in each frame. Then, the model parameters are updated subsequently and efficiently.

II. DET

A. Network Architecture and Learning Algorithm

To learn an effective matching function between template patch and the candidate patches taken from incoming image frame, we need the function be robust to all sorts of distortions. The Siamese architecture [9], [11] has been successfully applied to different kinds of matching tasks. Our method adopts the Siamese deep CNN. Fig. 1 shows the network architecture of our method. The architecture receives 32×32 RGB inputs, and has ten hidden convolutional layers. The ten hidden layers are adopted from the first ten convolutional layers of VGG16 net [12], including four blocks. The difference is that we put a Batch Normalization [13] layer after each convolutional layer and before the rectified linear units [14]. The first two blocks each contains two convolutional layers and the last two blocks each contains three convolutional layers. We denote each hidden layer as $\mathrm{Conv}ab$, where $a \in \{1,\ 2,\ 3,\ 4\}$ represents the block number and $b \in \{1,\ 2\}$ or $\{1, 2, 3\}$ denotes the sublayer number in each block. To take full use of the network, the proposed method uses the outputs of $\mathrm{Conv}12$, $\mathrm{Conv}22$, $\mathrm{Conv}33$,

Manuscript received July 15, 2017; revised August 30, 2017; accepted September 1, 2017. Date of publication September 6, 2017; date of current version September 18, 2017. This work was supported in part by the Major Science Instrument Program of the National Natural Science Foundation of China under Grant 61527802, and in part by the General Program of National Nature Science Foundation of China under Grant 61371132 and Grant 61471043. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mehdi Moradi. *(Corresponding author: Tingfa Xu.)*

The authors are with the School of Optical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: jieguo_2013@163.com; tfxu_bit@163.com).

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

Digital Object Identifier 10.1109/LSP.2017.2749458

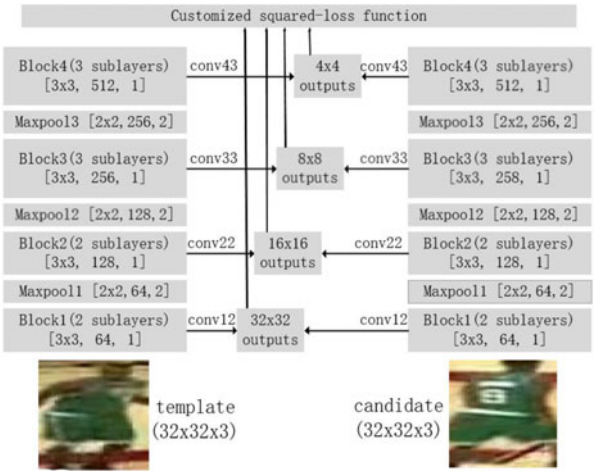

Fig. 1. Proposed Siamese deep CNN network to learn the multilevel partial information for generic matching. The network does not involve any fully connected layer. Numbers in square brackets are kernel size, number of outputs (feature dimension size), and stride. Note that Siamese network shares weights.

and $\mathrm{Conv43}$ to form the final outputs of the base matchers. In these four convolutional layers, $\mathrm{Conv12}$ and $\mathrm{Conv22}$ contain partial detail information of the target, thus can handle precise localization. $\mathrm{Conv33}$ and $\mathrm{Conv43}$ capture more semantic information which can be used to account for large appearance changes. Note that we match the template and candidates from the feature dimension, leading to hundreds of thousands of base matchers. In our architecture, $\mathrm{Conv12}$, $\mathrm{Conv22}$, $\mathrm{Conv33}$, and $\mathrm{Conv43}$ layers have $32 \times 32, 16 \times 16, 8 \times 8$, and 4×4 features abstracted from their feature dimension, respectively. Thus, we get $32 \times 32, 16 \times 16, 8 \times 8$, and 4×4 base matchers from these layers, respectively.

Outputs (distance metric function): Let F_{k_t} and F_{k_c} denote the template feature sets and the candidate feature sets, respectively, where $k \in \{12,\ 22,\ 33,\ 43\}$ corresponds to the convolutional layers. Feature $f^i_{k\_t} \in F_{k\_t}$ is the ith template feature extracted from the corresponding convolutional layer denoted by k. We propose to use Cosine as the distance metric

$$d^i_k(t,c) = \frac{\left\langle f^i_{k\_t},\ f^i_{k\_c} \right\rangle}{\left\| f^i_{k\_t} \right\|_2 \left\| f^i_{k\_c} \right\|_2} \tag{1}$$

where t indicates the template, c represents the candidate, $d^i_k(t,c)$ is the Cosine similarity between feature $f^i_{k\_t}$ and $f^i_{k\_c}$, and $v1,\ v2$ denotes inner product of the two vectors. The Cosine have two main advantages serving as the distance metric. First, it has nothing to do with the amplitude. This property can handle illumination changes. Second, its value ranges from –1 to 1. Thus, we can directly use the value as the confident score of the current candidate and further help us formulate the base matcher weights update strategy.

Loss function: The convolutional features of the template and the positive candidate should be close enough to let the Cosine value be 1. The Cosine value should be small than a margin value between template features and the negative candidate features. Thus, we develop a customized squared-loss function

$$\mathcal{L}\left(f^i_{k\_t}, f^i_{k\_c},\ y\right) = y\left(1 - d^i_k\left(t,c\right)\right)^2 + (1-y)\max\left(0,\ d^i_k\left(t,c\right) - \epsilon\right)^2 \tag{2}$$

where $y = 1$ if the candidate is positive and $y = 0$ if the candidate is negative, and ϵ is the maximum margin that $d^i_k(t,c)$ should satisfy. In this letter, we set ϵ to be zero, because the background candidate is uncorrelated with the template candidate in most cases, resulting in a zero value Cosine. Note that (2) only describes the loss of the ith base matcher from the corresponding convolutional layer indicated by k and we denote it as base loss function. The final loss function is a weighted sum of different base loss functions from different output layers:

$$L(t,c,y) = \sum_k \Psi_k \sum_i \mathcal{L}(f^i_{k\_t},\ f^i_{k\_c}, y) \tag{3}$$

where Ψ_k is the normalization factor of the output layer indicated by $k \in \{12,\ 22,\ 33,\ 43\}$

$$\Psi_k = 1/\sum_i \mathcal{L}(f^i_{k\_t}, f^i_{k\_c}, y). \tag{4}$$

The normalization factors let every output layer has the same cost weight.

Training data: The training data should be diverse enough to learn a robust similarity metric function. A small number of training data will make the tuned network parameters overfit to particular object categories. Thus, we use sequences in ALOV [15], VOT2014 [16], and VOT2015 [17] datasets to train our network and evaluate our method on OTB [18] benchmark. Note that we exclude the sequences in the training datasets that are also in OTB dataset. We randomly choose two frames in a sequence. One frame is used to extract the ground-truth patch as the template while the other frame is used to extract the candidate patches. The candidate patch is considered to be positive if its intersection-over-union (IoU) overlap ratio with the corresponding ground-truth box is larger than 0.75 and considered to be negative if the IoU overlap ratio is smaller than 0.35. We collect 50 positive samples and 150 negative samples in each frame.

Training: We train the network using the Adam [19] algorithm with each mini-batch consists of 32 positive samples and 160 negative samples. We also train the network for 15 K iterations per epoch and totally ten epochs, where K is the number of training sequences. The initial learning rate is set to be 0.0001 and multiplied by 0.8 after each epoch. The weights are initialized by the first ten convolutional layer weights of VGG16 network which is pretrained on ImageNet to further avoid overfitting.

B. Boosting Strategy

Strong matcher: We have trained the general matching function which outputs a large number of base matching results. We get the final result as an ensemble of all the base matchers. However, directly averaging all the matching results is not wise and loses much useful information. The base matchers from conv12 and conv22 layers capture partial details. They can reflect the reliability of corresponding target appearance parts.

For example, when the target is partially occluded, the matching results of the corresponding base matchers tend to be small while the results corresponding to the unoccluded part will be large. Even the high-level conv33 and conv44 layers have different receptive field centers, resulting in emphasis on different partial appearances. Those base matchers corresponding with occluded or corrupted receptive fields will have low-matching scores and those with distinct receptive fields will have high-matching scores. Hence, we propose to put a weight on every base matcher and the final matching result is a weighted ensemble of all the base matchers:

$$D(t,c) = \sum_k \phi_k \sum_i w_k^i d_k^i(t,c) \tag{5}$$

where w_k^i is the weight of the *i*th base matcher from the *k*th layer and $\phi_k = 1/\sum_i w_k^i d_k^i$ is the normalization factor. The base matcher weight should reflect the reliability of each base matcher in the current circumstance. This needs the weights to satisfy two conditions. First, the weights should tell how similar the features of the corresponding receptive fields are between the current target and the template. Second, the weights should help to suppress the potential distractors.

Distractor detection: In each frame, we sort the final matching scores of all the candidates. Then, we choose $M(=4)$ estimated negative candidates which have the highest matching scores and the IoU overlap rates between each other and the optimal candidate are larger than 0.33. These M candidates form the distractor set P in the current frame.

Weight propagation: the weights should reflect the confident scores of the base matchers and help to suppress the potential distractors. Hence, we design the following weight update strategy:

$$w_k^i = (1-\sigma) w_{kt-1}^i + \sigma(d_k^i(t,\hat{c}) - \frac{1}{M}\sum_m d_k^i(t,p_m)) \tag{6}$$

where $\sigma(=0.5)$ is the learning rate of base matcher weights, w_{kt-1}^i is the weight at $t-1$ frame, $\hat{c}$ represents the optimal candidate in the current frame, $p_m \in P$ is the selected distractor, and $m \in \{1,2,\ldots,M\}$. Although we only use the first ground-truth target as the template, the boosting strategy helps us pay more attention to those useful base matchers in the current circumstance. This compensates for the lack of online selecting template and reduces the risk of choosing misaligned target as the template.

C. Online Tracking With the Proposed Algorithm

We draw $N(=550)$ samples from the state variable $\mathbf{x}_t = [l_x, l_y, s]^{\mathrm{T}}$, where l_x, l_y, s denote x, y translations and scale variation (SV), respectively. The state vector is modeled by the Gaussian distribution, i.e., $N(\mathbf{x}_t; \mathbf{x}_{t-1}, \varphi)$, where φ is a diagonal covariance matrix whose diagonal entries are $(0.25r^2,\ 0.25r^2,\ 0.09)$, and r is the mean of current target height and width. We sample 400 candidates around the last tracking result $\mathbf{x}_{t-1}$. To avoid temporal drifting, we also sample candidates around the previous 15 tracking results (i.e., $N(\mathbf{x}_t; \mathbf{x}_{t-2}, \varphi), \ldots, N(\mathbf{x}_t; \mathbf{x}_{t-16}, \varphi)$). This sampling strategy performs especially well when the target undergoes short-term occlusion or deformation. We sample ten candidates around each of these tracking results, thus obtaining totally 550 candidates. Through this method, we get the candidate set $C_t = \{c_t^1, c_t^2, \ldots, c_t^N\}$.

We pass all the candidates through the network and pick the candidate that matches best to the template

$$\hat{c} = \arg\max_{c_t^n} D(t, c_t^n) \tag{7}$$

where $c_t^n \epsilon\, C_t$.

Although the boosting strategy helps us update the weights of the base matchers online and this can handle most appearance changes. Still, when severe deformation or background clutter (BC) comes, the tracker may drift away. Thus, to make the tracker adapt to fast appearance changes, we collect 32 positive samples and 96 negative samples to online train the network with just one epoch. The mini-batch consists of eight positive samples and 24 negative samples. Other training parameters are the same as in the offline training phase. The whole tracking algorithm is summarized in Algorithm 1.

Algorithm 1: The Proposed Tracker.

Input: The sampled candidate set $C_t = \{c_t^1, c_t^2, \ldots, c_t^N\}$, the template t, the pre-trained Siamese network and the initial base matcher weights (all set to be 1).

1: **For** $n = 1, \ldots, N$
2: Pass candidate c_t^n and template t through the network, getting the base matching scores using (1);
3: Get final ensemble matching score $D(t,\ c_t^n)$ using (5).
4: **End**
5: Find the optimal candidate $\hat{c}$ using (7);
6: Detect the distractors and propagate the base matcher weights using (6);
7: Collect samples to update the network parameters.

Output:

8: The optimal candidate $\hat{c}$, updated base tracker weights, and updated network parameters.

III. Experiments

The proposed method in this letter is implemented using Keras toolbox [20]. We perform the experiments on a PC with Intel i7-4790 CPU (3.6 GHz) together with a single NVIDIA GeForce GTX Titan X GPU and the tracker runs at 3.4 fps. We evaluate the proposed DET on OTB dataset and compare it with nine state-of-the-art trackers, including SiamFC [8], SINT [7], staple [21], MEEM [22], DLSSVM [23], DSST [24], Struck [25], SCM [26], and DET_ed. The DET_ed tracker is a variation of the proposed DET tracker. Similarly with [7], it uses the Euclidean distance as the distance metric function instead of the Cosine. The current confident score of each base matcher is formed by rescaling all the Euclidean distances to range $[0, 1]$ linearly, i.e., the confident score of the maximum Euclidean distance will be 0 and the confident score of the minimum Euclidean distance will be 1. Note that we exclude the "*David*" sequence because of the wrongly labeled ground truth.

TABLE I
AVERAGE AUC OF THE TEN TRACKERS IN TERMS OF DIFFERENT ATTRIBUTES: TOP THREE RESULTS ARE SHOWN IN RED, BLUE, AND GREEN FONTS

Attribute	Struck	SCM	DSST	MEEM	DLSSVM	staple	SiamFC	SINT	DET_ed	**DET**
OCC	0.419	0.479	0.524	0.552	0.591	0.586	0.591	0.606	**0.623**	**0.655**
DEF	0.401	0.434	0.494	0.563	**0.637**	0.609	0.529	**0.657**	0.606	0.615
FM	0.462	0.296	0.435	0.552	0.553	0.508	0.561	**0.617**	0.535	**0.587**
IV	0.435	0.463	0.553	0.533	0.540	0.559	0.535	**0.624**	0.549	**0.610**
SV	0.432	0.510	0.532	0.500	0.493	0.543	0.600	**0.603**	0.566	**0.639**
MB	0.450	0.261	0.433	0.542	**0.582**	0.519	0.529	**0.610**	0.486	0.564
BC	0.458	0.450	0.517	0.570	**0.592**	0.576	0.554	**0.619**	0.543	0.562
LR	0.372	0.279	0.409	0.362	0.430	0.438	**0.566**	**0.596**	0.472	0.512
IPR	0.450	0.449	0.552	0.534	0.557	0.573	0.578	**0.595**	0.538	**0.609**
OPR	0.436	0.464	0.528	0.560	0.583	0.569	0.585	**0.608**	0.587	**0.648**
OV	0.459	0.361	0.459	0.592	0.581	0.547	0.635	**0.671**	0.600	**0.677**
Overall score	0.478	0.495	0.549	0.566	0.590	0.596	0.610	**0.632**	0.603	**0.644**

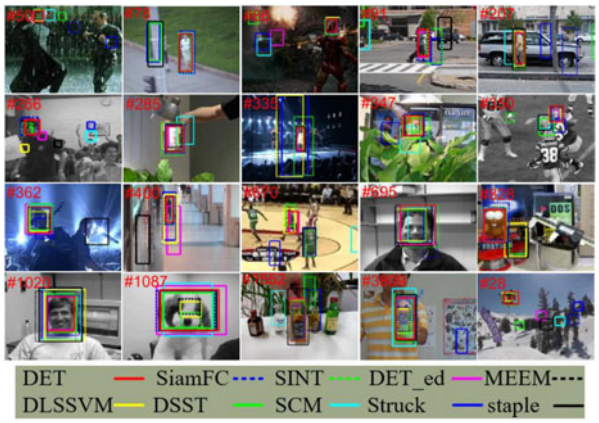

Fig. 2. Tracking results on some most challenging sequences (*matrix*, *jogging-2*, *ironman*, *couple*, *david3*, *freeman4*, *coke*, *singer1*, *tiger1*, *football*, *shaking*, *walking2*, *basketball*, *fleetface*, *lemming*, *dudek*, *dog1*, *liquor*, *doll*, *skiing*, from left to right and top to bottom).

Qualitative evaluation: The OTB sequences pose many challenging problems, including 29 sequences with target occlusion (OCC), 19 sequences with target deformation (DEF), 17 sequences with target fast motion (FM), 25 sequences with target illumination variation (IV), 28 sequences with target SV), 12 sequences with target motion blur (MB), 21 sequences with target BCs, four sequences with target low resolution (LR), 31 sequences with target in-plane rotation (IPR), 39 sequences with target out-of-plane rotation (OPR), and six sequences with target out-of-view (OV).

Fig. 2 shows the tracking results on some most challenge sequences in the benchmark. In sequences *singer1*, *dudek*, *dog1*, and *doll*, the targets undergo severe scale changes. The proposed tracker works well in all these sequences while other trackers fail in some frames. In *skiing*, the target goes through deformation, scale change, MB, LR, and rotation simultaneously. The DET and the SINT are the only trackers which estimate both position and scale well among all these trackers. In *david3*, the target first flips horizontally, then shortly occluded by a tree. The SiamFC fails to track the target, because they only use the first ground-truth patch as their template and never do any update procedure. Our tracker tracks well since we update the base matcher weights to adapt the tracker to current circumstance.

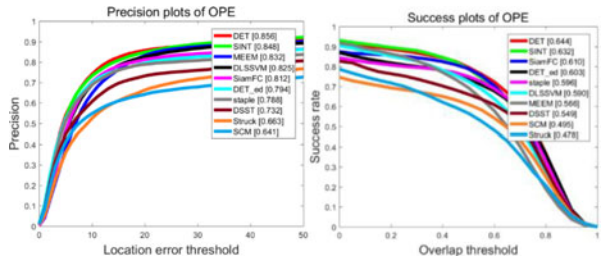

Fig. 3. Precision and success plots. The legend of the precision plot reports the distance precision score for the threshold = 20 pixels and the legend of the success plot reports AUC scores.

Quantitative evaluation: Fig. 3 contains the precision and success plots. The legend of the precision plot reports the precision scores threshold of 20 pixels for each method and the legend of the success plot reports area under curve (AUC) scores. Both plots show that our tracker has the best overall performance among all the trackers.

Table I reports the AUC scores of all the compared trackers under different attributes. In 5 out of the total 11 attributes, our tracker achieves the highest AUC scores. In attributes DEF, FM, IV, MB, and LR, our trackers are among the best three. However, in attribute BC, when the target undergoes BC, our tracker performs not as well as SINT, DLSSVM, staple and MEEM. This is because we just use the first frame's ground-truth patch as the template. When BC comes together with severe target appearance change, the tracker may drift away. However, AUC score of 0.562 is still much better than the other four state-of-the-art trackers. The overall AUC scores presented in Table I show that our tracker significantly outperforms the others.

IV. CONCLUSION

This letter presents a novel deep ensemble-tracking method which combines the boosting algorithm with Siamese deep CNN. The proposed model matches the candidates with the first frame ground-truth template and finds the optimal candidate in each frame. The final robust strong matcher is a weighted ensemble of both low-level feature and high-level feature base matchers. The proposed boosting strategy helps the model adapt to various target appearance changes. Experiments show effectiveness and robustness of the proposed method.

REFERENCES

[1] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. Brit. Mach. Vis. Conf.*, 2006, pp. 6.1–6.10.

[2] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[3] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 234–247.

[4] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 983–990.

[5] K. Zhang and H. Song, "Real-time visual tracking via online weighted multiple instance learning," *Pattern Recognit.*, vol. 46, no. 1, pp. 397–411, 2013.

[6] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 3119–3127.

[7] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 1420–1429.

[8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 850–865.

[9] S. Chopra, R. Hadsell, and Y. Lecun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. Comput. Vis. Pattern Recognit.*, vol. 1, 2005, pp. 539–546.

[10] X. Han, T. Leung, Y. Jia, and R. Sukthankar, "MatchNet: Unifying feature and metric learning for patch-based matching," in *Proc. Comput. Vis. Pattern Recognit.*, 2015, pp. 3279–3286.

[11] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. Comput. Vis. Pattern Recognit.*, 2015, pp. 4353–4361.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–13.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[15] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[16] M. Kristan *et al.*, "The visual object tracking VOT2014 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2014, pp. 98–111.

[17] M. Kristan *et al.*, "The visual object tracking vot2015 challenge results," in *Proc. Int. Conf. Comput. Vis. Workshop*, 2015, pp. 1–23.

[18] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. Comput. Vis. Pattern Recognit.*, 2013, pp. 2411–2418.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[20] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/ fchollet/keras

[21] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 1401–1409.

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

[23] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang, "Object tracking via dual linear structured SVM and explicit feature map," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 4266–4274.

[24] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 65.1–65.11

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

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