Fully-Convolutional Siamese Networks for Object Tracking

Article · June 2016

CITATIONS
437

READS
2,284

5 authors, including:

Luca Bertinetto
University of Oxford
22 PUBLICATIONS **3,020** CITATIONS
SEE PROFILE

Jack Valmadre
The Commonwealth Scientific and Industrial Research Organisation
24 PUBLICATIONS **2,250** CITATIONS
SEE PROFILE

Joao Henriques
University of Porto
44 PUBLICATIONS **5,941** CITATIONS
SEE PROFILE

Andrea Vedaldi
University of Oxford
176 PUBLICATIONS **24,817** CITATIONS
SEE PROFILE

Fully-Convolutional Siamese Networks for Object Tracking

Luca Bertinetto* Jack Valmadre* João F. Henriques
Andrea Vedaldi Philip H. S. Torr

Department of Engineering Science, University of Oxford
{name.surname}@eng.ox.ac.uk

Abstract. The problem of arbitrary object tracking has traditionally been tackled by learning a model of the object's appearance exclusively online, using as sole training data the video itself. Despite the success of these methods, their online-only approach inherently limits the richness of the model they can learn. Recently, several attempts have been made to exploit the expressive power of deep convolutional networks. However, when the object to track is not known beforehand, it is necessary to perform Stochastic Gradient Descent online to adapt the weights of the network, severely compromising the speed of the system. In this paper we equip a basic tracking algorithm with a novel fully-convolutional Siamese network trained end-to-end on the ILSVRC15 video object detection dataset. Our tracker operates at frame-rates beyond real-time and, despite its extreme simplicity, achieves state-of-the-art performance in the VOT2015 benchmark.

Keywords: object-tracking, Siamese-network, similarity-learning, deep-learning

1 Introduction

We consider the problem of tracking an arbitrary object in video, where the object is identified solely by a rectangle in the first frame. Since the algorithm may be requested to track any arbitrary object, it is impossible to have already gathered data and trained a detector.

For several years, the most successful paradigm for this scenario has been to learn a model of the object's appearance in an online fashion using examples extracted from the video itself [1]. This owes in large part to the demonstrated ability of MILTrack [2], Struck [3], TLD [4] and KCF [5]. However, a clear deficiency of using data exclusively from the current video is that only comparatively simple models can be learnt. While other problems in computer vision have seen an increasingly pervasive adoption of deep convolutional networks (conv-nets) trained from large supervised datasets, the scarcity of supervised data and the constraint of real-time operation prevent the naive application of deep learning within this paradigm of learning a detector per video.

* The first two authors contributed equally, and are listed in alphabetical order.

Several recent works have aimed to overcome this limitation using a pre-trained deep conv-net that was learnt for a different but related task. These approaches either apply shallow methods (e.g. correlation filters) using the network's internal representation as features [6,7] or perform SGD (stochastic gradient descent) to fine-tune multiple layers of the network [8,9]. While the use of shallow methods does not take full advantage of the benefits of end-to-end learning, methods that apply SGD during tracking to achieve state-of-the-art results have not been able to operate in real-time.

We advocate an alternative approach in which a deep conv-net is trained to address a more general *similarity learning* problem in an initial offline phase, and then this function is simply evaluated during tracking. The key contribution of this paper is to demonstrate that this approach achieves competitive performance in modern tracking benchmarks at speeds that far exceed the real-time requirement. Specifically, we train a Siamese network to locate an *exemplar* image within a larger *search* image. A further contribution is a novel Siamese architecture that is *fully-convolutional* with respect to the search image: dense and efficient sliding-window evaluation is achieved with a bilinear layer that computes the cross-correlation of two inputs.

We posit that the similarity learning approach has gone relatively neglected because the tracking community did not have access to vast labelled datasets. Until recently, the available datasets comprised only a few hundred annotated videos. However, we believe that the emergence of the ILSVRC dataset for object detection from video [10] (henceforth ImageNet Video) makes it possible to train such a model. Furthermore, the fairness of training and testing deep models for tracking using videos from the same domain is a point of controversy. We show that our model generalizes from the ImageNet Video domain to the ALOV/VOT domain, enabling the datasets of tracking benchmarks to be reserved for testing purposes.

2 Deep similarity learning for tracking

Learning to track arbitrary objects can be addressed using similarity learning. We propose to learn a function $f(z, x)$ that compares an exemplar image z to a candidate image x of the same size and returns a high score if the two images depict the same object and a low score otherwise. To find the position of the object in a new image, we can then exhaustively test all possible locations and choose the candidate with the maximum similarity to the past appearance of the object. In experiments, we will simply use the initial appearance of the object as the exemplar. The function f will be learnt from a dataset of videos with labelled object trajectories.

Given their widespread success in computer vision [11,12,13,14], we will use a deep conv-net as the function f. Similarity learning with deep conv-nets is typically addressed using Siamese architectures [15,16,17]. Siamese networks apply

This was recently prohibited by the VOT committee. `http://votchallenge.net/`

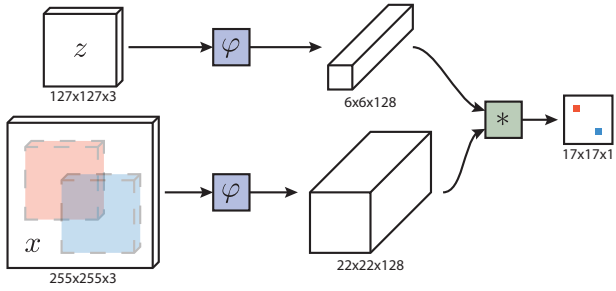

Fig. 1. Fully-convolutional Siamese architecture. Our architecture is fully-convolutional with respect to the search image x. The output is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for all translated sub-windows within the search image in one evaluation. In this example, the red and blue pixels in the score map contain the similarities for the corresponding sub-windows. Best viewed in colour.

an identical transformation φ to both inputs and then combine their representations using another function g according to $f(z, x) = g(\varphi(z), \varphi(x))$. When the function g is a simple distance or similarity metric, the function φ can be considered an embedding. Deep Siamese conv-nets have previously been applied to tasks such as face verification [16,18,12], keypoint descriptor learning [17,19] and one-shot character recognition [20].

2.1 Fully-convolutional Siamese architecture

We propose a Siamese architecture which is *fully-convolutional* with respect to the candidate image x. We say that a function is fully-convolutional if it commutes with translation. To give a more precise definition, introducing L_τ to denote the translation operator $(L_\tau x)[u] = x[u - \tau]$, a function h that maps signals to signals is fully-convolutional with integer stride k if

$$h(L_{k\tau} x) = L_\tau h(x) \tag{1}$$

for any translation τ. (When x is a finite signal, this only need hold for the valid region of the output.)

The advantage of a fully-convolutional network is that, instead of a candidate image of the same size, we can provide as input to the network a much larger *search* image and it will compute the similarity at all translated sub-windows on a dense grid in a single evaluation. To achieve this, we use a convolutional embedding function φ and combine the resulting feature maps using a cross-correlation layer

$$f(z, x) = \varphi(z) * \varphi(x) + b\,\mathbb{1} \tag{2}$$

where $b\,\mathbb{1}$ denotes a signal which takes value $b \in \mathbb{R}$ in every location. The output of this network is not a single score but rather a score map defined on a finite

grid $\mathcal{D} \subset \mathbb{Z}^2$ as illustrated in Figure 1. Note that the embedding of the exemplar $\varphi(z)$ can be a feature map with spatial structure as opposed to a plain vector.

During tracking, we use a search image centred at the previous position of the target. The position of the maximum score relative to the centre of the score map, multiplied by the stride of the network, gives the displacement of the target from frame to frame. Multiple scales are searched in a single forward-pass by assembling a mini-batch of scaled images.

Combining feature maps using cross-correlation and evaluating the network once on the larger search image is mathematically equivalent to combining feature maps using the inner product and evaluating the network on each translated sub-window independently. However, the cross-correlation layer provides an incredibly simple method to implement this operation efficiently within the framework of existing conv-net libraries. For comparison, Tao et al. [21] test a collection of bounding boxes using RoI pooling and do not report being able to achieve real-time performance. While this is clearly useful during testing, it can also be exploited during training.

2.2 Training with large search images

We employ a discriminative approach, training the network on positive and negative pairs and adopting the logistic loss

$$\ell(y, v) = \log(1 + \exp(-yv)) \tag{3}$$

where v is the real-valued score of a single exemplar-candidate pair and $y \in \{+1, -1\}$ is its ground-truth label. We exploit the fully-convolutional nature of our network during training by using pairs that comprise an exemplar image and a larger search image. This will produce a map of scores $v : \mathcal{D} \to \mathbb{R}$, effectively generating many examples per pair. We define the loss of a score map to be the mean of the individual losses

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} \ell(y[u], v[u]) \ , \tag{4}$$

requiring a true label $y[u] \in \{+1, -1\}$ for each position $u \in \mathcal{D}$ in the score map. The parameters of the conv-net θ are obtained by applying Stochastic Gradient Descent (SGD) to the problem

$$\arg\min_{\theta} \mathop{\mathbb{E}}_{(z,x,y)} L(y, f(z, x; \theta)) \ . \tag{5}$$

Pairs are obtained from a dataset of annotated videos by extracting exemplar and search images that are centred on the target, as shown in Figure 2. The images are extracted from two frames of a video that both contain the object and are at most T frames apart. The class of the object, if present, is completely ignored during training. The scale of the object within each image is normalized without corrupting the aspect ratio of the image. The elements of the score map

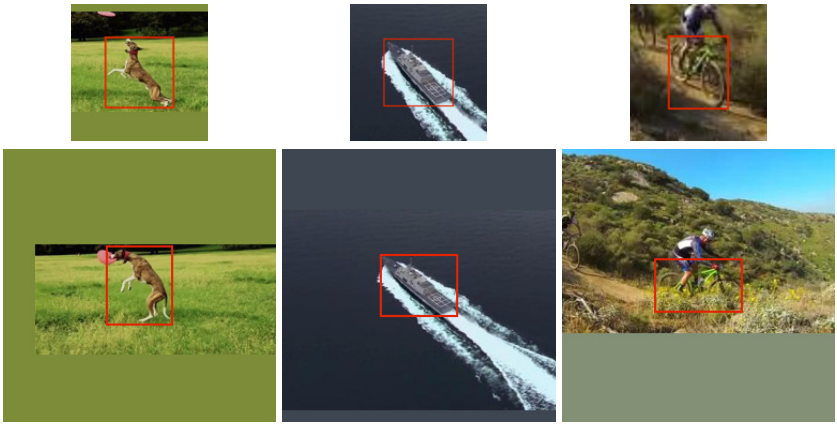

Fig. 2. Training pairs extracted from the same video: exemplar image and corresponding search image from same video. When a sub-window extends beyond the extent of the image, the missing portions are filled with the mean RGB value.

are considered to belong to a positive example if they are within radius R of the centre (accounting for the stride k of the network)

$$y[u] = \begin{cases} +1 & \text{if } k\|u - c\| \leq R \\ -1 & \text{otherwise} \end{cases} . \tag{6}$$

The losses of the positive and negative examples in the score map are weighted to eliminate class imbalance.

Since our network is fully-convolutional, there is no risk that it learns a bias for the sub-window at the centre. We believe that it is effective to consider search images centred on the target because it is likely that the most difficult sub-windows, and those which have the most influence on the performance of the tracker, are those adjacent to the target.

Note that since the network is symmetric $f(z, x) = f(x, z)$, it is in fact also fully-convolutional in the exemplar. While this allows us to use different size exemplar images for different objects in theory, we assume uniform sizes because it simplifies the mini-batch implementation. However, this assumption could be relaxed in the future.

2.3 ImageNet Video for tracking

The 2015 edition of ImageNet Large Scale Visual Recognition Challenge [10] (ILSVRC) introduced the ImageNet Video dataset as part of the new *object detection from video* challenge. Participants are required to classify and locate objects from 30 different classes of animals and vehicles. The training set alone contains almost 4000 videos, with a total of more than one million annotated frames. This number is particularly impressive if compared to the number of

labelled sequences in VOT [22], ALOV [1] and OTB [23], which together total less than 500 videos. We believe that this dataset should be of extreme interest to the tracking community not only for its vast size, but also because it depicts scenes and objects different to those found in the canonical tracking benchmarks. For this reason, it can safely be used to train a deep model for tracking without over-fitting to the domain of videos used in these benchmarks.

2.4 Practical considerations

Dataset curation In order to make ImageNet Video more suitable for the problem of object tracking, we make the following modifications.

- We discard the classes snake, train, whale and lizard because these objects often occupy a small fraction of their rectangle or extend beyond the edge of the image.
- We exclude objects whose area is greater than 0.75^2 or less than 0.1^2 of the total image area.
- We exclude frames in which the distance of the object from the edge of the image is less than 5% of its average dimension. This is to eliminate examples where the object is truncated by the image boundary.

The curated dataset comprises 843,371 objects from 2820 videos.

Scale normalization During training, we adopt exemplar images that are 127×127 and search images that are 255×255 pixels. Images are scaled such that the bounding box, plus an added margin for context, has a fixed area. More precisely, if the tight bounding box has size (w, h) and the context margin is p, then the scale factor s is chosen such that the area of the scaled rectangle is equal to a constant

$$s(w + 2p) \times s(h + 2p) = A \ . \tag{7}$$

We use the area of the exemplar images $A = 127^2$ and set the amount of context to be half of the mean dimension $p = (w+h)/4$. Exemplar and search images for every frame are extracted offline to avoid image resizing during training. This pre-processed dataset and the code to generate it will be made available on our website.

Network architecture The architecture that we adopt for the embedding function φ resembles the convolutional stage of the network of Krizhevsky et al. [14]. The dimensions of the parameters and activations are given in Table 1. Max-pooling is employed after the first two convolutional layers. ReLU non-linearities follow every convolutional layer except for conv5, the final layer. During training, batch-normalization is inserted immediately after every linear layer. The stride of the final representation is eight. An important aspect of the design is that no padding is introduced within the network. Although this is common practice in image classification, it violates the fully-convolutional property of eq. 1.

Table 1. Architecture of convolutional embedding function, which is similar to the convolutional stage of the network of Krizhevsky et al. [14]. The channel map property describes the number of input and output channels of each convolutional layer.

Layer	Support	Chan. map	Stride	Activation size for exemplar	Activation size for search	chans.
				127×127	255×255	$\times 3$
conv1	11×11	48×3	2	59×59	123×123	$\times 48$
pool1	3×3		2	29×29	61×61	$\times 48$
conv2	5×5	128×24	1	25×25	57×57	$\times 48$
pool2	3×3		2	12×12	28×28	$\times 128$
conv3	3×3	192×128	1	10×10	26×26	$\times 192$
conv4	3×3	192×192	1	8×8	24×24	$\times 192$
conv5	3×3	128×192	1	6×6	22×22	$\times 128$

Tracking algorithm Since our purpose is to prove the efficacy of our fully-convolutional Siamese network and its generalization capability when trained on ImageNet Video, we use an extremely simplistic algorithm to perform tracking. Unlike more sophisticated trackers, we do not update a model or maintain a memory of past appearances, we do not incorporate additional cues such as optical flow or colour histograms, and we do not refine our prediction with bounding box regression. Yet despite the simplicity of the tracking algorithm, it achieves surprisingly good results when equipped with our learnt similarity metric. We do incorporate some elementary temporal constraints: we only search for the object within a region of approximately four times its previous size, and a cosine window is added to the score map to penalize large displacements. Tracking through scale space is achieved by processing several scaled versions of the search image. Any change in scale is penalized and updates of the current scale are damped.

3 Experiments

3.1 The VOT 2015 benchmark

For all our experiments, we use the latest stable version of the Visual Object Tracking (VOT) benchmark, which evaluates trackers on 60 sequences chosen from a pool of 356. The sequences have been selected so that seven challenging scenarios are well represented. Many of the sequences were originally presented in other datasets (e.g. ALOV [1] and OTB [23]).

Within the benchmark, trackers are automatically re-initialized five frames after failure, which is deemed to have occurred when the intersection-over-union (IoU) between the estimated bounding box and the ground truth becomes zero. Trackers are evaluated according to two measures of performance: *accuracy* and *robustness*. The former is calculated as the average IoU, while the latter is expressed in terms of the total number of failures. These give insight into the behaviour of a tracker. However, in order to obtain a single number by which to compare trackers, the *expected average overlap measure* is used, which computes the average IoU with no re-initialization following a failure.

`https://github.com/votchallenge/vot-toolkit` at tag `vot2015-final`

Table 2. Effects of using increasing portions of the dataset on tracker's performance.

Dataset (%)	# videos	# objects	accuracy	# failures	overlap
5	141	32k	0.459	217	0.144
10	282	95k	0.492	163	0.188
50	1410	430k	0.497	127	0.210
100	2820	843k	**0.502**	**93**	**0.244**

3.2 Implementation details

Training The parameters of the embedding are found by minimizing eq. 5 with a standard SGD solver using MatConvNet [24]. The initial values of the parameters follow a Gaussian distribution, scaled according to the improved Xavier method [25]. Training is performed over 50 epochs, each consisting of 50,000 sampled pairs (according to sec. 2.2). Negative pairs were chosen with probability 0.25. A negative pair comprises exemplar and search images from different videos, and all elements of the response map are considered negative examples. The gradients for each iteration are estimated using mini-batches of size 8, and the learning rate is annealed geometrically at each epoch from 10^{-2} to 10^{-5}. Finally, given the depth of the networks we applied batch normalization after each linear layer to aid convergence.

Tracking As mentioned earlier, the online phase is deliberately minimalistic. The embedding $\varphi(z)$ of the initial object appearance is computed once, and is compared convolutionally to sub-windows of the subsequent frames. We found that updating the embedding online through simple strategies, such as linear interpolation, does not gain much performance and thus we consider it to be fixed. We found that upsampling the score map using bicubic interpolation, from 17×17 to 272×272, results in more accurate localization since the original map is relatively coarse. To handle scale variations, we also search for the object over scales $1.03^{\{-1,0,1\}}$, and adapt the scale by linear interpolation with a factor of 0.65.

3.3 Dataset size

Table 2 illustrates how the size of the dataset used to train the Siamese network greatly influences the performance. The expected overlap steadily improves from 0.144 to 0.244 when increasing the size of the dataset from 5% to 100%. This finding suggests that using a larger video dataset could increase the performance even further.

3.4 The VOT15 benchmark results

We compare the method described in Section 2 (*SiameseFC* for Siamese Fully-Convolutional) against the best of the 62 trackers that participated in the 2015 edition of the VOT challenge [22]. Figure 3 illustrates the final ranking in terms

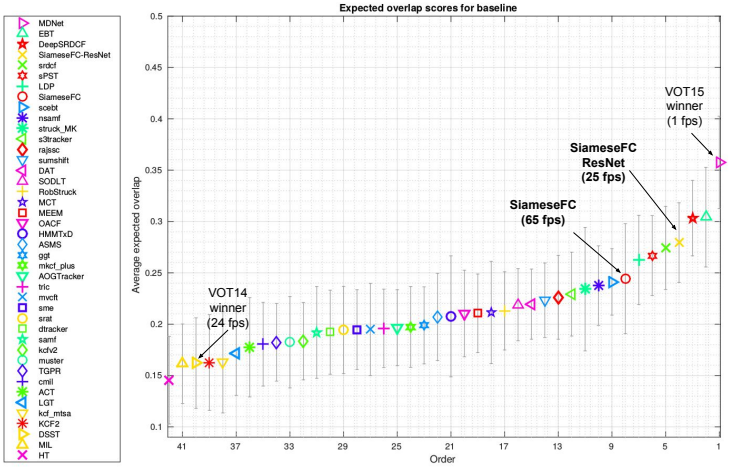

Fig. 3. VOT2015 final ranking in terms of expected average overlap. Only the best 40 results have been reported.

of expected average overlap. Despite its simplicity, our method outperforms most of the competing trackers, ranking among the top. This result is particularly interesting if we consider that the proposed method estimates the new position of the target object by merely cross-correlating the embeddings of two patches over three scales. The result demonstrates that the expressiveness of the similarity metric learnt by our fully-convolutional Siamese network trained on a large dataset *alone* is enough to achieve state-of-the-art results. We believe that even higher performance could be obtained by augmenting the online tracking pipeline with the methods often adopted by the tracking community (e.g. model update, bounding-box regression, fine-tuning, memory).

Table 3 reports raw scores and speed of the best 15 trackers of the challenge. The low complexity of our approach allows a fast execution time: the network alone can be evaluated at 130 Hz and the overall tracker runs at 65 fps, making it by far the fastest and the only one that achieves real-time performance. The performance of our system is particularly interesting especially if compared with other trackers that make use of conv-net features like MDNet and SO-DLT. In fact, their need to perform SGD online to adapt the network to the current video severely limits their applicability.

In addition to the architecture described in Table 1, we also measure the performance of the same simple tracker equipped with the ResNet architecture [33]. For this experiment, we start from a pre-trained model and fine-tune the parameters on ImageNet Video. With seven "bottleneck" blocks and an initial convolution, this network has 22 linear layers on the longest path from input to cross-correlation. The deeper model improves the performance: the expected average overlap increases 14% relative to our baseline *SiameseFC*. Nonetheless, the overall system can still track objects in real-time (25 fps). A benefit of our

Table 3. Raw scores, overlap and reported speed for our proposed method and the best 15 performing trackers of the VOT15 challenge. When available, we compare with the speed reported by the authors, otherwise (*) we report the values from the VOT15 results [22] in EFO units, which roughly correspond to fps (e.g. the speed of the NCC tracker is 140 fps and 160 EFO).

Tracker	accuracy	# failures	overlap	speed (fps)
MDNet [9]	0.5620	46	0.3575	1
EBT [26]	0.4481	49	0.3042	5
DeepSRDCF [7]	0.5350	60	0.3033	< 1 *
SiameseFC-ResNet	0.5527	81	0.2794	25
SRDCF [27]	0.5260	71	0.2743	5
sPST [28]	0.5230	85	0.2668	2
LDP [22]	0.4688	78	0.2625	4 *
SiameseFC-AlexNet	0.5016	93	0.2443	**65**
SC-EBT [29]	0.5171	103	0.2412	–
NSAMF [30]	0.5027	87	0.2376	5 *
StruckMK [3]	0.4442	90	0.2341	2
S3Tracker [31]	0.5031	100	0.2292	14 *
RAJSSC [22]	0.5301	105	0.2262	2 *
SumShift [31]	0.4888	97	0.2233	17 *
DAT [32]	0.4705	113	0.2195	15
SO-DLT [8]	0.5233	108	0.2190	5

approach is that any advances in deep learning and conv-nets can be almost effortlessly incorporated into the system.

4 Related work

Several recent works have sought to train Recurrent Neural Networks (RNNs) for the problem of object tracking. Gan et al. [34] train an RNN to predict the absolute position of the target in each frame and Kahou et al. [35] similarly train an RNN for tracking using a differentiable attention mechanism. These methods have not yet demonstrated competitive results on modern benchmarks, however it is certainly a promising avenue for future research. We remark that an interesting parallel can be drawn between this approach and ours, by interpreting a Siamese network as an unrolled RNN that is trained and evaluated on sequences of length two. Siamese networks could therefore serve as strong initialization for a recurrent model.

Denil et al. [36] track objects with a particle filter that uses a learnt distance metric to compare the current appearance to that of the first frame. However, their distance metric is vastly different to ours. Instead of comparing images of the entire object, they compute distances between fixations (foveated glimpses of small regions within the object's bounding box). To learn a distance metric, they train a Restricted Boltzmann Machine (RBM) and then use the Euclidean distance between hidden activations for two fixations. Although RBMs are unsupervised, they suggest training the RBM on random fixations within centred images of the object to detect. While tracking an object, they learn a stochastic policy for choosing fixations which is specific to that object, using uncertainty as a reward signal. Besides synthetic sequences of MNIST digits, this method has only been demonstrated qualitatively on problems of face and person tracking.

Held et al. [37] regress directly from a pair of images to the location in the second image of the object shown in the first image. Predicting a rectangle instead of a position has the advantage that changes in scale and aspect ratio can be handled without resorting to exhaustive evaluation. However, a disadvantage of this approach is that it does not possess intrinsic invariance to translation of the second image. This means that the network must be shown examples in all positions, which is achieved through considerable dataset augmentation. The distribution of translations must be chosen during training.

Chen et al. [38] train a network that maps an exemplar and a larger search region to a response map. However, their method also lacks invariance to translation of the second image since the last layers are fully-connected. Similarly to Held et al. [37], this is inefficient because the training set must represent all translations of all objects. Unlike our approach, they cannot adjust the size of the search region dynamically after training.

Concurrently with our own work, other authors have also suggested using Siamese networks in the context of tracking. Tao et al. [21] propose SINT (Siamese INstance search for Tracking), training a Siamese network to identify candidate image locations that match the initial object appearance. In contrast to our approach, they do not adopt an architecture which is fully-convolutional with respect to the search image. Instead, at test time, they sample bounding boxes uniformly on circles of varying radius as in Struck [3]. Moreover, they incorporate optical flow and bounding box regression to improve the results. In order to improve the computational speed of their system, they must employ Region of Interest (RoI) pooling to efficiently examine many overlapping sub-windows. Despite this optimization, the overall system speed is still far from being real-time.

Many of the above approaches start with a network that was trained for a different task and then fine-tune the parameters. We believe to be one of the first to demonstrate strong tracking results with a network that was trained from scratch using only a video dataset.

5 Conclusion

In this work, we depart from the traditional online learning methodology employed in tracking, and show a complementary approach that focuses on learning strong embeddings in an offline phase. Differently from their use in classification settings, we demonstrate that for tracking applications fully-convolutional deep networks have the ability to use the available data more efficiently. This is reflected both at test-time, by performing efficient spatial searches, but also at training-time, where every sub-window effectively represents a useful sample with little extra cost. The experiments show that deep embeddings provide a naturally rich source of features for online trackers, and allow for very simplistic test-time strategies to perform well. We believe that this approach is complementary to more sophisticated online tracking methodologies, and expect future work to explore this relationship more thoroughly.

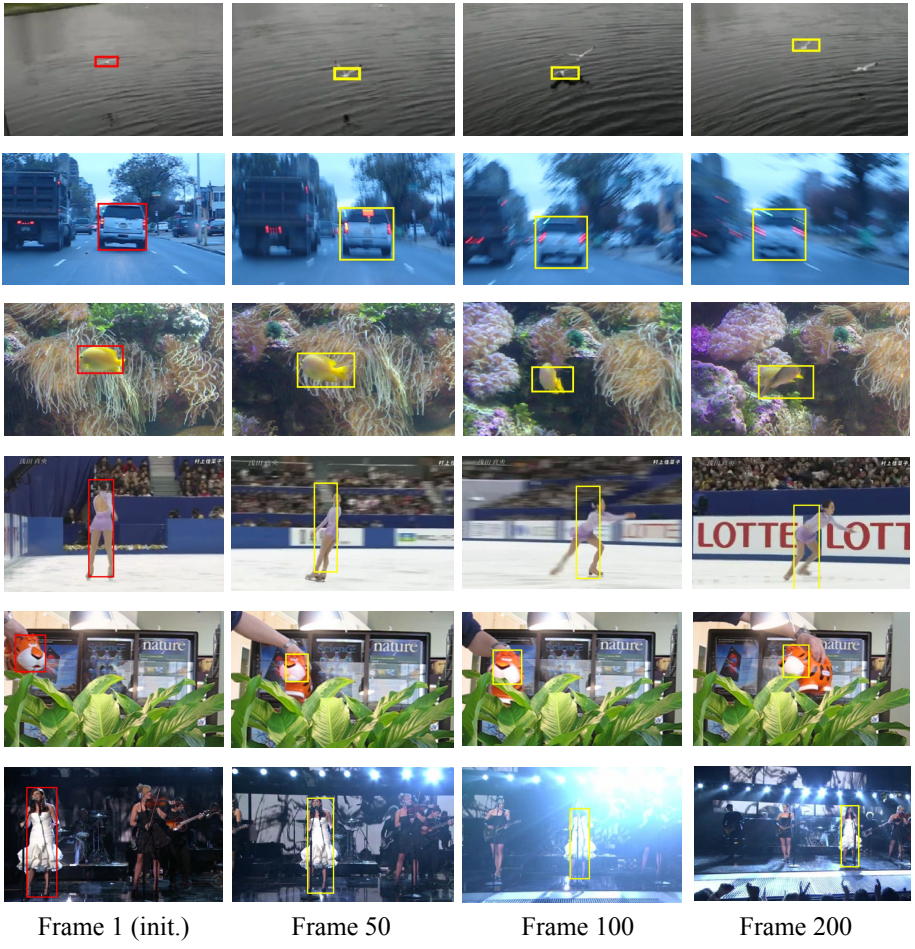

Fig. 4. Snapshots of the simple tracker described in Section 2.4 equipped with our proposed fully-convolutional Siamese network trained from scratch on ImageNet Video. Our method does not perform any model update, so it uses only the first frame to compute $\varphi(z)$. Nonetheless, it is surprisingly robust to a number of challenging situations like motion and camera blur (rows 1 and 2), change of appearance (rows 3, 4 and 5), poor illumination (row 6) and scale change (row 6). Sequences taken from the VOT15 benchmark: *birds1*, *car1*, *fish3*, *iceskater1*, *tiger*, *singer1*. The snapshots have been taken at fixed frames (1, 50, 100 and 200) and the tracker is not re-initialized.

References

1. Smeulders, A.W.M., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: An experimental survey. PAMI **36**(7) (2014) 1442–1468
2. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR 2009, IEEE (2009) 983–990
3. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: ICCV 2011, IEEE (2011) 263–270
4. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. PAMI **34**(7) (2012) 1409–1422
5. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. PAMI **37**(3) (2015) 583–596
6. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV 2015. (2015) 3074–3082
7. Danelljan, M., Hager, G., Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: ICCV 2015 Workshop. (2015) 58–66
8. Wang, N., Li, S., Gupta, A., Yeung, D.Y.: Transferring rich feature hierarchies for robust visual tracking. arXiv preprint arXiv:1501.04587 (2015)
9. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. arXiv preprint arXiv:1510.07945 (2015)
10. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
11. Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: An astounding baseline for recognition. In: CVPR 2014 Workshop. (2014) 806–813
12. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. BMVC 2015 **1**(3) (2015) 6
13. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: ICCV 2015. (2015) 2758–2766
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS 2012. (2012) 1097–1105
15. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using a "Siamese" time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence (1993)
16. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: Closing the gap to human-level performance in face verification. In: CVPR 2014. (2014) 1701–1708
17. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: CVPR 2015. (2015) 4353–4361
18. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: CVPR 2015. (2015) 815–823
19. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: ICCV 2015. (2015) 118–126
20. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML 2015 Deep Learning Workshop. (2015)
21. Tao, R., Gavves, E., Smeulders, A.W.M.: Siamese instance search for tracking. arXiv preprint arXiv:1605.05863 (2016)
22. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernandez, G., Vojir, T., Hager, G., Nebehay, G., Pflugfelder, R.: The Visual Object Tracking VOT2015 Challenge results. In: ICCV 2015 Workshop. (2015) 1–23

23. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR 2013. (2013)
24. Vedaldi, A., Lenc, K.: MatConvNet – Convolutional Neural Networks for MATLAB. (2015)
25. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: ICCV 2015. (2015)
26. Zhu, G., Porikli, F., Li, H.: Tracking randomly moving objects on edge box proposals. arXiv preprint arXiv:1507.08085 (2015)
27. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV 2015. (2015) 4310–4318
28. Hua, Y., Alahari, K., Schmid, C.: Online object tracking with proposal selection. In: ICCV 2015. (2015) 3092–3100
29. Wang, N., Yeung, D.Y.: Ensemble-based tracking: Aggregating crowdsourced structured time series data. In: ICML 2014. (2014) 1107–1115
30. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: ECCV 2014 Workshops, Springer (2014) 254–265
31. Li, A., Lin, M., Wu, Y., Yang, M.H., Yan, S.: NUS-PRO: A new visual tracking challenge. PAMI **38**(2) (2016) 335–349
32. Possegger, H., Mauthner, T., Bischof, H.: In defense of color-based model-free tracking. In: CVPR 2015. (2015) 2113–2120
33. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
34. Gan, Q., Guo, Q., Zhang, Z., Cho, K.: First step toward model-free, anonymous object tracking with recurrent neural networks. arXiv preprint arXiv:1511.06425 (2015)
35. Kahou, S.E., Michalski, V., Memisevic, R.: RATM: Recurrent Attentive Tracking Model. arXiv preprint arXiv:1510.08660 (2015)
36. Denil, M., Bazzani, L., Larochelle, H., de Freitas, N.: Learning where to attend with deep architectures for image tracking. Neural Computation **24**(8) (2012) 2151–2184
37. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. arXiv preprint arXiv:1604.01802 (2016)
38. Chen, K., Tao, W.: Once for all: A two-flow convolutional neural network for visual tracking. arXiv preprint arXiv:1604.07507 (2016)