

# Fully-Convolutional Siamese Networks for Object Tracking

Luca Bertinetto <sup>\*</sup>   Jack Valmadre <sup>\*</sup>   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 dataset for object detection in video. Our tracker operates at frame-rates beyond real-time and, despite its extreme simplicity, achieves state-of-the-art performance in multiple benchmarks.

**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 specific 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 methods like **TLD** [2], **Struck** [3] and **KCF** [4]. However, a clear deficiency of using data derived 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.

---

<sup>\*</sup> 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** [5,6] or perform SGD (stochastic gradient descent) to fine-tune multiple layers of the network [7,8,9]. 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 online during tracking. The key contribution of this paper is to demonstrate that this approach achieves very competitive performance in modern tracking benchmarks at speeds that far **exceed** the frame-rate 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 its 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. In fact, 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 in 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, as it has been recently prohibited by the VOT committee. We show that our model generalizes from the ImageNet Video domain to the ALOV/OTB/VOT [1,11,12] domain, enabling the videos 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 [13,14,15,16], we will use a deep conv-net as the function  $f$ . Similarity learning with deep conv-nets is typically addressed using Siamese architectures [17,18,19]. Siamese networks apply an identical transformation  $\varphi$  to both inputs and then combine their represen-

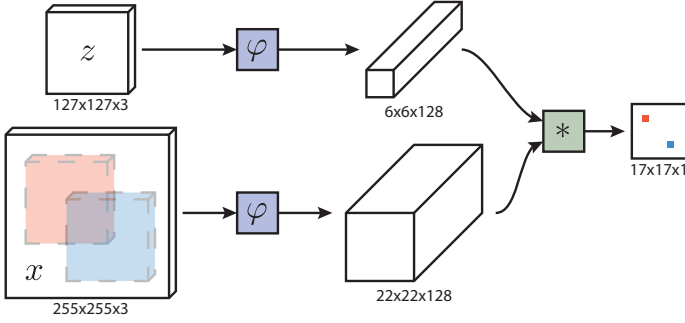


Fig.1: Fully-convolutional Siamese architecture. Our architecture is fully-convolutional with respect to the search image  $x$ . The output is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for all translated sub-windows 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.

tations 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  $\varphi$  can be considered an embedding. Deep Siamese conv-nets have previously been applied to tasks such as face verification [18, 20, 14], **keypoint descriptor learning** [19, 21] and one-shot character recognition [22].

## 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  $L_\tau$  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

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

for any translation  $\tau$ . (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**  $\varphi$  and combine the resulting feature maps using a cross-correlation layer

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

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 output of the embedding function is a feature map with spatial support as opposed to a plain vector. The same technique has been applied in contemporary work on stereo matching [23].

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. 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

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

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  $v : \mathcal{D} \rightarrow \mathbb{R}$ , effectively generating many examples per pair. We define the loss of a score map to be the mean of the individual losses

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

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  $\theta$  are obtained by applying Stochastic Gradient Descent (SGD) to the problem

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

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 is ignored during training. The scale of the object within each image is normalized without corrupting the

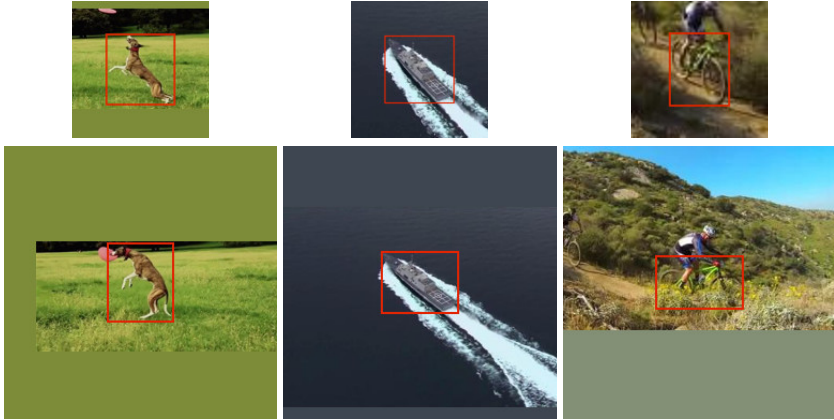


Fig. 2: Training pairs extracted from the same video: exemplar image and corresponding search image from same video. When a sub-window extends beyond the extent of the image, the missing portions are filled with the mean RGB value.

aspect ratio of the image. The elements of the score map 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)

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

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. Training and validation sets together contain almost 4500 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 [12], ALOV [1] and OTB [11], 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** During training, we adopt exemplar images that are  $127 \times 127$  and search images that are  $255 \times 255$  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

$$s(w + 2p) \times s(h + 2p) = A . \quad (7)$$

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. In a preliminary version of this work, we adopted a few heuristics to limit the number of frames from which to extract the training data. For the experiments of this paper, instead, we have used *all* 4417 videos of ImageNet Video, which account for more than 2 million labelled bounding boxes.

**Network architecture** The architecture that we adopt for the embedding function  $\varphi$  resembles the convolutional stage of the network of Krizhevsky et al. [16]. 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 [24] 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.

**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 its simplicity, the tracking algorithm achieves surprisingly good results when equipped with our offline-learned similarity metric.

Table 1: Architecture of convolutional embedding function, which is similar to the convolutional stage of the network of Krizhevsky et al. [16]. The channel map property describes the number of output and input channels of each convolutional layer.

Layer	Support	Chan. map	Stride	Activation size		
				for exemplar	for search	chans.
				$127 \times 127$	$255 \times 255$	$\times 3$
conv1	$11 \times 11$	$96 \times 3$	2	$59 \times 59$	$123 \times 123$	$\times 96$
pool1	$3 \times 3$		2	$29 \times 29$	$61 \times 61$	$\times 96$
conv2	$5 \times 5$	$256 \times 48$	1	$25 \times 25$	$57 \times 57$	$\times 256$
pool2	$3 \times 3$		2	$12 \times 12$	$28 \times 28$	$\times 256$
conv3	$3 \times 3$	$384 \times 256$	1	$10 \times 10$	$26 \times 26$	$\times 192$
conv4	$3 \times 3$	$384 \times 192$	1	$8 \times 8$	$24 \times 24$	$\times 192$
conv5	$3 \times 3$	$256 \times 192$	1	$6 \times 6$	$22 \times 22$	$\times 128$

Online, 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 Related work

Several recent works have sought to train Recurrent Neural Networks (RNNs) for the problem of object tracking. Gan et al. [25] train an RNN to predict the absolute position of the target in each frame and Kahou et al. [26] 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. [27] 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. This must either be performed online or in an offline phase with knowledge of the object to track. 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.

While it is infeasible to train a deep conv-net from scratch for each new video, several works have investigated the feasibility of fine-tuning from pre-trained parameters at test time. SO-DLT [7] and MDNet [9] both train a convolutional network for a similar detection task in an offline phase, then at test-time use SGD to learn a detector with examples extracted from the video itself as in the conventional tracking-as-detector-learning paradigm. These methods cannot operate at frame-rate due to the computational burden of evaluating forward and backward passes on many examples. An alternative way to leverage conv-nets for tracking is to apply traditional shallow methods using the internal representation of a pre-trained convolutional network as features. While trackers in this style such as DeepSRDCF [6], Ma et al. [5] and FCNT [8] have achieved strong results, they have been unable to achieve frame-rate operation due to the relatively high dimension of the conv-net representation.

Concurrently with our own work, some other authors have also proposed using conv-nets for object tracking by learning a function of pairs of images. Held et al. [28] introduce GOTURN, in which a conv-net is trained to regress directly from two 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 their 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. Chen et al. [29] 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 final layers are fully-connected. Similarly to Held et al., this is inefficient because the training set must represent all translations of all objects. Their method is named YCNN for the Y shape of the network. Unlike our approach, they cannot adjust the size of the search region dynamically after training. Tao et al. [30] propose to train a Siamese network to identify candidate image locations that match the initial object appearance, dubbing their method SINT (Siamese INstance search Tracker). 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 employ Region of Interest (RoI) pooling to efficiently examine many overlapping sub-windows. Despite this optimization, at 2 frames per second, the overall system is still far from being real-time.

All of the competitive methods above that train on video sequences (MDNet [9], SINT [30], GOTURN [28]), use training data belonging to the same



ALOV/OTB/VOT domain used by the benchmarks. This practice has been forbidden in the VOT challenge due to concerns about over-fitting to the scenes and objects in the benchmark. Thus an important contribution of our work is to demonstrate that a conv-net can be trained for effective object tracking without using videos from the same distribution as the testing set.

## 4 Experiments

### 4.1 Implementation details

**Training.** The parameters of the embedding function are found by minimizing eq. 5 with straightforward SGD using MatConvNet [31]. The initial values of the parameters follow a Gaussian distribution, scaled according to the improved Xavier method [32]. Training is performed over 50 epochs, each consisting of 50,000 sampled pairs (according to sec. 2.2). 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  $10^{-5}$ .

**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 feature representation of) the exemplar online through simple strategies, such as linear interpolation, does not gain much performance and thus we keep it fixed. We found that upsampling the score map using bicubic interpolation, from  $17 \times 17$  to  $272 \times 272$ , results in more accurate localization since the original map is relatively coarse. To handle scale variations, we also search for the object over five scales  $1.025^{\{-2, -1, 0, 1, 2\}}$ , and update the scale by linear interpolation with a factor of 0.35 to provide damping.

In order to make our experimental results reproducible, we share training and tracking code, together with the scripts to generate the curated dataset at <http://www.robots.ox.ac.uk/~luca/siamese-fc.html>. On a machine equipped with a single NVIDIA GeForce GTX Titan X and an Intel Core i7-4790K at 4.0GHz, our full online tracking pipeline operates at 86 and 58 frames-per-second, when searching respectively over 3 and 5 scales.

### 4.2 Evaluation

We evaluate two variants of our simplistic tracker: SiamFC (Siamese Fully-Convolutional) and SiamFC-3s, which searches over 3 scales instead of 5.

### 4.3 The OTB-13 benchmark

The OTB-13 [11] benchmark considers the average per-frame *success rate* at different thresholds: a tracker is successful in a given frame if the intersection-over-union (IoU) between its estimate and the ground-truth is above a certain

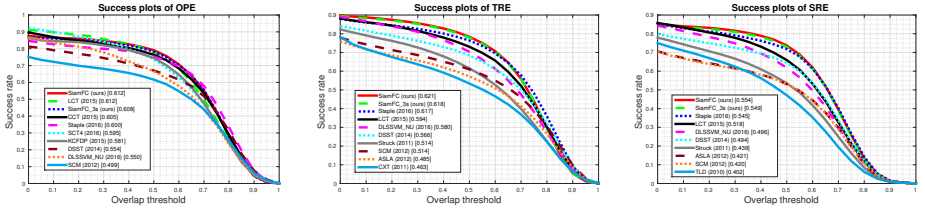


Fig. 3: Success plots for OPE (one pass evaluation), TRE (temporal robustness evaluation) and SRE (spatial robustness evaluation) of the OTB-13 [11] benchmark. The results of CCT, SCT4 and KCFDP were only available for OPE at the time of writing.

threshold. Trackers are then compared in terms of area under the curve of success rates for different values of this threshold. In addition to the trackers reported by [11], in Figure 3 we also compare against seven more recent state-of-the-art trackers presented in the major computer vision conferences and that can run at frame-rate speed: Staple [33], LCT [34], CCT [35], SCT4 [36], DLSSVM\_NU [37], DSST [38] and KCFDP [39]. Given the nature of the sequences, for this benchmark only we convert 25% of the pairs to grayscale during training. All the other hyper-parameters (for training and tracking) are fixed.

#### 4.4 The VOT benchmarks

For our experiments, we use the latest stable version of the Visual Object Tracking (VOT) toolkit (tag `vot2015-final`), which evaluates trackers on sequences chosen from a pool of 356, selected so that seven different challenging situations are well represented. Many of the sequences were originally presented in other datasets (e.g. ALOV [1] and OTB [11]). Within the benchmark, trackers are automatically re-initialized five frames after failure, which is deemed to have occurred when the IoU between the estimated bounding box and the ground truth becomes zero.

**VOT-14 results.** We compare our method SiamFC (and the variant SiamFC-3s) against the best 10 trackers that participated in the 2014 edition of the VOT challenge [40]. We also include Staple [33] and GOTURN [28], two recent real-time trackers presented respectively at CVPR 2016 and ECCV 2016. 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. Figure 4 shows the Accuracy-Robustness plot, where the best trackers are closer to the top-right corner.

**VOT-15 results.** We also compare our method against the 40 best participants in the 2015 edition [12]. In this case, the raw scores of accuracy and number

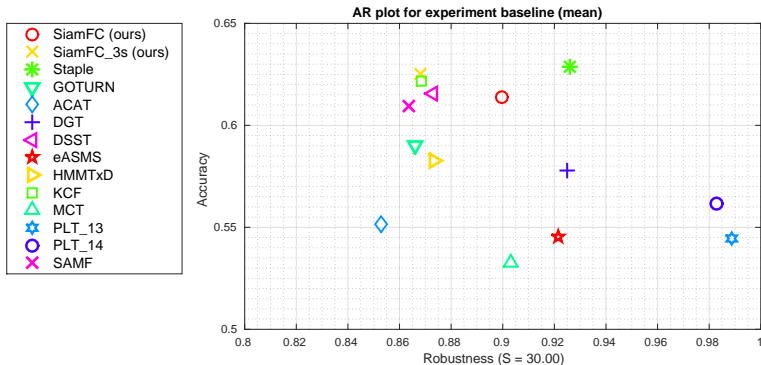


Fig. 4: VOT-14 Accuracy-robustness plot. Best trackers are closer to the top-right corner.

of failures are used to compute the *expected average overlap measure*, which represents the average IoU with no re-initialization following a failure. Figure 5 illustrates the final ranking in terms of expected average overlap, while Table 2 reports scores and speed of the 15 highest ranked trackers of the challenge.

**VOT-16 results.** At the time of writing, the results of the 2016 edition were not available. However, to facilitate an early comparison with our method, we report our scores. For SiamFC and SiamFC-3s we obtain, respectively, an overall expected overlap (average between the *baseline* and *unsupervised* experiments) of 0.3876 and 0.4051. Please note that these results are different from the VOT-16 report, as our entry in the challenge was a preliminary version of this work.

Despite its simplicity, our method improves over recent state-of-the-art real-time trackers (Figures 3 and 4). Moreover, it outperforms most of the best methods in the challenging VOT-15 benchmark, while being the only one that achieves frame-rate speed (Figure 5 and Table 2). These results demonstrate that the expressiveness of the similarity metric learnt by our fully-convolutional Siamese network on ImageNet Video *alone* is enough to achieve very strong results, comparable or superior to recent state-of-the-art methods, which often are several orders of magnitude slower. We believe that considerably higher performance could be obtained by augmenting the minimalist online tracking pipeline with the methods often adopted by the tracking community (e.g. model update, bounding-box regression, fine-tuning, memory).

#### 4.5 Dataset size

Table 3 illustrates how the size of the dataset used to train the Siamese network greatly influences the performance. The expected average overlap (measured on

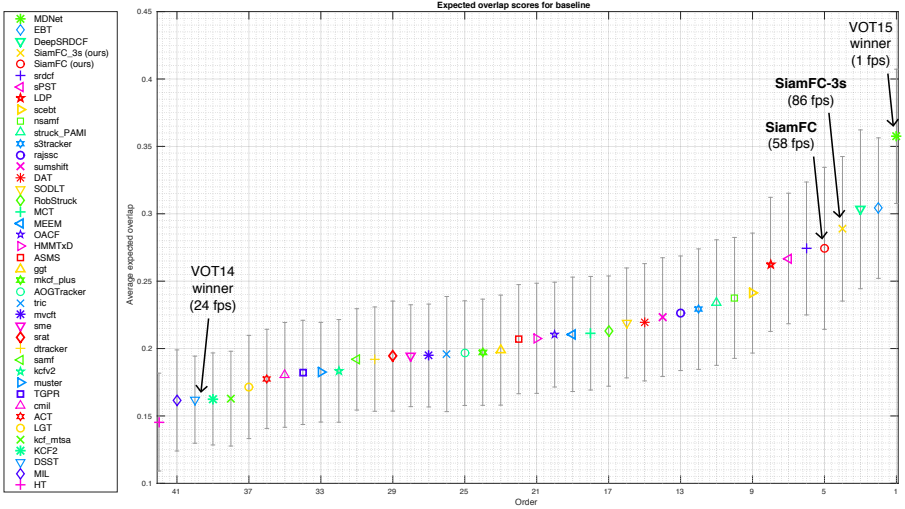


Fig. 5: VOT-15 ranking in terms of expected average overlap. Only the best 40 results have been reported.

VOT-15) steadily improves from 0.168 to 0.274 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. In fact, even if 2 million supervised bounding boxes might seem a huge number, it should not be forgotten that they still belong to a relatively moderate number of videos, at least compared to the amount of data normally used to train conv-nets.

## 5 Conclusion

In this work, we depart from the traditional online learning methodology employed in tracking, and show an alternative approach that focuses on learning strong embeddings in an offline phase. Differently from their use in classification settings, we demonstrate that for tracking applications Siamese 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 enable 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.

Table 2: Raw scores, overlap and reported speed for our proposed method and the best 15 performing trackers of the VOT-15 challenge. Where available, we compare with the speed reported by the authors, otherwise (\*) we report the values from the VOT-15 results [12] 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 [41]	0.4481	49	0.3042	5
DeepSRDCF [6]	0.5350	60	0.3033	< 1 *
<b>SiamFC-3s</b> (ours)	0.5335	84	0.2889	<b>86</b>
<b>SiamFC</b> (ours)	0.5240	87	0.2743	58
SRDCF [42]	0.5260	71	0.2743	5
sPST [43]	0.5230	85	0.2668	2
LDP [12]	0.4688	78	0.2625	4 *
SC-EBT [44]	0.5171	103	0.2412	—
NSAMF [45]	0.5027	87	0.2376	5 *
StruckMK [3]	0.4442	90	0.2341	2
S3Tracker [46]	0.5031	100	0.2292	14 *
RAJSSC [12]	0.5301	105	0.2262	2 *
SumShift [46]	0.4888	97	0.2233	17 *
DAT [47]	0.4705	113	0.2195	15
SO-DLT [7]	0.5233	108	0.2190	5

Table 3: Effects of using increasing portions of the ImageNet Video dataset on tracker’s performance.

Dataset (%)	# videos	# objects	accuracy	# failures	expected avg. overlap
2	88	60k	0.484	183	0.168
4	177	110k	0.501	160	0.192
8	353	190k	0.484	142	0.193
16	707	330k	0.522	132	0.219
32	1413	650k	0.521	117	0.234
100	4417	2m	<b>0.524</b>	<b>87</b>	<b>0.274</b>



Fig. 6: 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 blur (row 2), drastic change of appearance (rows 1, 3 and 4), poor illumination (row 6) and scale change (row 6). On the other hand, our method is sensitive to scenes with confusion (row 5), arguably because the model is never updated and thus the cross-correlation gives a high scores for all the windows that are similar to the first appearance of the target. All sequences come from the VOT-15 benchmark: *gymnastics1*, *car1*, *fish3*, *iceskater1*, *marching*, *singer1*. The snapshots have been taken at fixed frames (1, 50, 100 and 200) and the tracker is never 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. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. *PAMI* **34**(7) (2012) 1409–1422
3. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: *ICCV 2011*, IEEE
4. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. *PAMI* **37**(3) (2015) 583–596
5. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: *ICCV 2015*
6. Danelljan, M., Hager, G., Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: *ICCV 2015 Workshop*. (2015) 58–66
7. Wang, N., Li, S., Gupta, A., Yeung, D.Y.: Transferring rich feature hierarchies for robust visual tracking. *arXiv CoRR* (2015)
8. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: *ICCV 2015*
9. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. *arXiv CoRR* (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. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: *CVPR 2013*. (2013)
12. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernandez, G., Vojir, T., Hager, G., Nebel, G., Pflugfelder, R.: The Visual Object Tracking VOT2015 Challenge results. In: *ICCV 2015 Workshop*. (2015) 1–23
13. Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: An astounding baseline for recognition. In: *CVPR 2014 Workshop*
14. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. *BMVC 2015* (2015)
15. 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*
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *NIPS 2012*
17. 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)
18. 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
19. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: *CVPR 2015*. (2015)
20. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: *CVPR 2015*. (2015) 815–823
21. 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

22. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML 2015 Deep Learning Workshop. (2015)
23. Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. In: CVPR 2016. (2016) 5695–5703
24. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML 2015. (2015) 448–456
25. Gan, Q., Guo, Q., Zhang, Z., Cho, K.: First step toward model-free, anonymous object tracking with recurrent neural networks. arXiv CoRR (2015)
26. Kahou, S.E., Michalski, V., Memisevic, R.: RATM: Recurrent Attentive Tracking Model. arXiv CoRR (2015)
27. Denil, M., Bazzani, L., Larochelle, H., de Freitas, N.: Learning where to attend with deep architectures for image tracking. *Neural Computation* (2012)
28. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. arXiv CoRR (2016)
29. Chen, K., Tao, W.: Once for all: A two-flow convolutional neural network for visual tracking. arXiv CoRR (2016)
30. Tao, R., Gavves, E., Smeulders, A.W.M.: Siamese instance search for tracking. arXiv CoRR (2016)
31. Vedaldi, A., Lenc, K.: MatConvNet – Convolutional Neural Networks for MATLAB. (2015)
32. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: ICCV 2015. (2015)
33. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. CVPR 2016 (2016)
34. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. In: CVPR 2015
35. Zhu, G., Wang, J., Wu, Y., Lu, H.: Collaborative correlation tracking. In: BMVC 2015
36. Choi, J., Jin Chang, H., Jeong, J., Demiris, Y., Young Choi, J.: Visual tracking using attention-modulated disintegration and integration. In: CVPR 2016
37. Ning, J., Yang, J., Jiang, S., Zhang, L., Yang, M.H.: Object tracking via dual linear structured svm and explicit feature map. In: CVPR 2016
38. Danelljan, M., Häger, G., Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC 2014
39. Huang12, D., Luo, L., Wen12, M., Chen12, Z., Zhang12, C.: Enable scale and aspect ratio adaptability in visual tracking with detection proposals
40. LIRIS, F.: The visual object tracking vot2014 challenge results
41. Zhu, G., Porikli, F., Li, H.: Tracking randomly moving objects on edge box proposals. arXiv CoRR (2015)
42. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV 2015. (2015) 4310–4318
43. Hua, Y., Alahari, K., Schmid, C.: Online object tracking with proposal selection. In: ICCV 2015. (2015) 3092–3100
44. Wang, N., Yeung, D.Y.: Ensemble-based tracking: Aggregating crowdsourced structured time series data. In: ICML 2014. (2014) 1107–1115
45. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: ECCV 2014 Workshops. (2014)
46. Li, A., Lin, M., Wu, Y., Yang, M.H., Yan, S.: NUS-PRO: A new visual tracking challenge. PAMI **38**(2) (2016) 335–349
47. Possegger, H., Mauthner, T., Bischof, H.: In defense of color-based model-free tracking. In: CVPR 2015. (2015)