End-to-end representation learning for Correlation Filter based tracking

Jack Valmadre* Luca Bertinetto* João F. Henriques Andrea Vedaldi Philip H. S. Torr
University of Oxford
{name.surname}@eng.ox.ac.uk

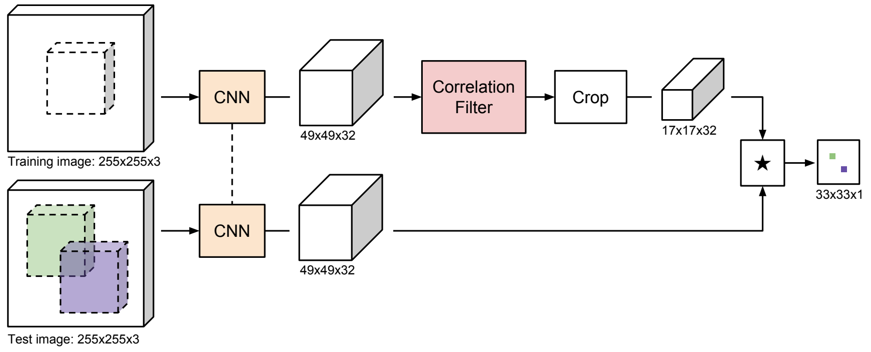

Figure 1: Overview of the proposed network architecture, **CFNet**. It is an asymmetric Siamese network: after applying the same convolutional feature transform to both input images, the "training image" is used to learn a linear template, which is then applied to search the "test image" by cross-correlation.

Abstract

The Correlation Filter is an algorithm that trains a linear template to discriminate between images and their translations. It is well suited to object tracking because its formulation in the Fourier domain provides a fast solution, enabling the detector to be re-trained once per frame. Previous works that use the Correlation Filter, however, have adopted features that were either manually designed or trained for a different task. This work is the first to overcome this limitation by interpreting the Correlation Filter learner, which has a closed-form solution, as a differentiable layer in a deep neural network. This enables learning deep features that are tightly coupled to the Correlation Filter. Experiments illustrate that our method has the important practical benefit of allowing lightweight architectures to achieve state-of-the-art performance at high framerates.

1. Introduction

Deep neural networks are a powerful tool for learning image representations in computer vision applications. However, training deep networks online, in order to capture previously unseen object classes from one or few examples, is challenging. This problem emerges naturally in applications such as visual object tracking, where the goal is to re-detect an object over a video with the sole supervision of a bounding box at the beginning of the sequence. The main challenge is the lack of a-priori knowledge of the target object, which can be of any class.

The simplest approach is to disregard the lack of a-priori knowledge and adapt a pre-trained deep convolutional neural network (CNN) to the target, for example by using stochastic gradient descent (SGD), the workhorse of deep network optimization [32, 26, 36]. The extremely limited training data and large number of parameters make this a difficult learning problem. Furthermore, SGD is quite expensive for online adaptation [32, 26].

A possible answer to these shortcomings is to have no online adaptation of the network. Recent works have focused on learning deep embeddings that can be used as universal object descriptors [3, 13, 29, 18, 5]. These methods use a Siamese CNN, trained offline to discriminate whether two image patches contain the same object or not. The idea is that a powerful embedding will allow the detection (and thus tracking) of objects via similarity, bypassing the online learning problem. However, using a fixed metric to compare appearance prevents the learning algorithm from exploiting any video-specific cues that could be helpful for discrimination.

An alternative strategy is to use instead an online learn-

*Equal first authorship.

ing method such as the *Correlation Filter* (CF). The CF is an efficient algorithm that learns to discriminate an image patch from the surrounding patches by solving a large ridge regression problem extremely efficiently [4, 14]. It has proved to be highly successful in object tracking (*e.g*. [6, 19, 23, 2]), where its efficiency enables a tracker to adapt its internal model of the object on the fly at every frame. It owes its speed to a Fourier domain formulation, which allows the ridge regression problem to be solved with only a few applications of the Fast Fourier Transform (FFT) and cheap element-wise operations. Such a solution is, by design, much more efficient than an iterative solver like SGD, and still allows the discriminator to be tailored to a specific video, contrary to the embedding methods.

The challenge, then, is to combine the online learning efficiency of the CF with the discriminative power of CNN features trained offline. This has been done in several works (*e.g*. [22, 7, 9, 32]), which have shown that CNNs and CFs are complementary and their combination results in improved performance.

However, in the aforementioned works, the CF is simply applied on top of pre-trained CNN features, without any deep integration of the two methods. End-to-end training of deep architectures is generally preferable to training individual components separately. The reason is that in this manner the free parameters in all components can co-adapt and cooperate to achieve a single objective. Thus it is natural to ask whether a CNN-CF combination can also be trained end-to-end with similar benefits.

The key step in achieving such integration is to interpret the CF as a differentiable CNN layer, so that errors can be propagated through the CF back to the CNN features. This is challenging, because the CF itself is the solution of a learning problem. Hence, this requires to differentiate the solution of a large linear system of equations. This paper provides a closed-form expression for the derivative of the Correlation Filter. Moreover, we demonstrate the practical utility of our approach in training CNN architectures end-to-end.

We present an extensive investigation into the effect of incorporating the CF into the fully-convolutional Siamese framework of Bertinetto *et al*. [3]. We find that the CF does not improve results for networks that are sufficiently deep. However, our method enables ultra-lightweight networks of a few thousand parameters to achieve state-of-the-art performance on multiple benchmarks while running at high framerates.

Code and results are available online [1].

[1] www.robots.ox.ac.uk/~luca/cfnet.html

2. Related work

Since the seminal work of Bolme *et al*. [4], the Correlation Filter has enjoyed great popularity within the tracking community. Notable efforts have been devoted to its improvement, for example by mitigating the effect of periodic boundaries [10, 16, 8], incorporating multi-resolution feature maps [22, 9] and augmenting the objective with a more robust loss [27]. For the sake of simplicity, in this work we adopt the basic formulation of the Correlation Filter.

Recently, several methods based on Siamese networks have been introduced [29, 13, 3], raising interest in the tracking community for their simplicity and competitive performance. For our method, we prefer to build upon the fully-convolutional Siamese architecture [3], as it enforces the prior that the appearance similarity function should commute with translation.

At its core, the Correlation Filter layer that we introduce amounts to computing the solution to a regularized deconvolution problem, not to be confused with upsampling convolution layers that are sometimes referred to as "deconvolution layers" [21]. Before it became apparent that algorithms such as SGD are sufficient for training deep networks, Zeiler *et al*. [35] introduced a deep architecture in which each layer solves a convolutional sparse coding problem. In contrast, our problem has a closed-form solution since the Correlation Filter employs quadratic regularization rather than 1-norm regularization.

The idea of back-propagating gradients through the solution to an optimization problem during training has been previously investigated. Ionescu *et al*. [15] and Murray [25] have presented back-propagation forms for the SVD and Cholesky decomposition respectively, enabling gradient descent to be applied to a network that computes the solution to either a system of linear equations or an eigenvalue problem. Our work can be understood as an efficient back-propagation procedure through the solution to a system of linear equations, where the matrix has circulant structure.

When the solution to the optimization problem is obtained iteratively, an alternative is to treat the iterations as a Recurrent Neural Network, and to explicitly unroll a fixed number of iterations [37]. Maclaurin *et al*. [24] go further and back-propagate gradients through an entire SGD learning procedure, although this is computationally demanding and requires judicious bookkeeping. Gould *et al*. [11] have recently considered differentiating the solution to general $\arg\min$ problems without restricting themselves to iterative procedures. However, these methods are unnecessary in the case of the Correlation Filter, as it has a closed-form solution.

Back-propagating through a learning algorithm invites a comparison to meta-learning. Recent works [31, 1] have proposed feed-forward architectures that can be interpreted as learning algorithms, enabling optimization by gradient descent. Rather than adopt an abstract definition of learning, this paper propagates gradients through a conventional learning problem that is already widely used.

3. Method

We briefly introduce a framework for learning embeddings with Siamese networks (Section 3.1) and the use of such an embedding for object tracking (Section 3.2) before presenting the CFNet architecture (Section 3.3). We subsequently derive the expressions for evaluation and back-propagation of the main new ingredient in our networks, the Correlation Filter layer, which performs online learning in the forward pass (Section 3.4).

3.1. Fully-convolutional Siamese networks

Our starting point is a network similar to that of [3], which we later modify in order to allow the model to be interpreted as a Correlation Filter tracker. The fully-convolutional Siamese framework considers pairs (x', z') comprising a training image x' and a test image z'[2]. The image x' represents the object of interest (*e.g.* an image patch centered on the target object in the first video frame), while z' is typically larger and represents the search area (*e.g.* the next video frame).

Both inputs are processed by a CNN f_ρ with learnable parameters ρ. This yields two feature maps, which are then cross-correlated:

$$g_\rho(x', z') = f_\rho(x') \star f_\rho(z') \ . \tag{1}$$

Eq. 1 amounts to performing an exhaustive search of the pattern x' over the test image z'. The goal is for the maximum value of the response map (left-hand side of eq. 1) to correspond to the target location.

To achieve this goal, the network is trained offline with millions of random pairs (x'_i, z'_i) taken from a collection of videos. Each example has a spatial map of labels c_i with values in $\{-1, 1\}$, with the true object location belonging to the positive class and all others to the negative class. Training proceeds by minimizing an element-wise logistic loss ℓ over the training set:

$$\arg\min_\rho \quad \sum_i \ell\left(g_\rho(x'_i, z'_i),\, c_i\right) \ . \tag{2}$$

3.2. Tracking algorithm

The network itself only provides a function to measure the similarity of two image patches. To apply this network to object tracking, it is necessary to combine this with a procedure that describes the logic of the tracker. Similar to [3], we employ a simplistic tracking algorithm to assess the utility of the similarity function.

Online tracking is performed by simply evaluating the network in forward-mode. The feature representation of the target object is compared to that of the search region, which is obtained in each new frame by extracting a window centred at the previously estimated position, with an area that is four times the size of the object. The new position of the object is taken to be the location with the highest score.

The original fully-convolutional Siamese network simply compared every frame to the initial appearance of the object. In contrast, we compute a new template in each frame and then combine this with the previous template in a moving average.

3.3. Correlation Filter networks

We propose to modify the baseline Siamese network of eq. 1 with a Correlation Filter block between x and the cross-correlation operator. The resulting architecture is illustrated in Figure 1. This change can be formalized as:

$$h_{\rho,s,b}(x', z') = s\,\omega\left(f_\rho(x')\right) \star f_\rho(z') + b \tag{3}$$

The CF block $w = \omega(x)$ computes a standard CF template w from the training feature map $x = f_\rho(x')$ by solving a ridge regression problem in the Fourier domain [14]. Its effect can be understood as crafting a discriminative template that is robust against translations. It is necessary to introduce scalar parameters s and b (scale and bias) to make the score range suitable for logistic regression. Offline training is then performed in the same way as for a Siamese network (Section 3.1), replacing g with h in eq. 2.

We found that it was important to provide the Correlation Filter with a large region of context in the training image, which is consistent with the findings of Danelljan et al. [8] and Kiani et al. [16]. To reduce the effect of circular boundaries, the feature map x is pre-multiplied by a cosine window [4] and the final template is cropped [30].

Notice that the forward pass of the architecture in Figure 1 corresponds exactly to the operation of a standard CF tracker [14, 6, 23, 3] with CNN features, as proposed in previous work [22, 7]. However, these earlier networks were not trained end-to-end. The novelty is to compute the derivative of the CF template with respect to its input so that a network incorporating a CF can be trained end-to-end.

3.4. Correlation Filter

We now show how to back-propagate gradients through the Correlation Filter solution efficiently and in closed form via the Fourier domain.

Formulation. Given a scalar-valued image $x \in \mathbb{R}^{m\times m}$, the Correlation Filter is the template $w \in \mathbb{R}^{m\times m}$ whose inner product with each circular shift of the image $x * \delta_{-u}$ is

[2]Note that this differs from [3], in which the target object and search area were instead denoted z and x respectively.

as close as possible to a desired response $y[u]$ [14], minimizing

$$\sum_{u\in\mathcal{U}} \left(\langle x * \delta_{-u}, w\rangle - y[u]\right)^2 = \|w \star x - y\|^2 \ . \quad (4)$$

Here, $\mathcal{U} = \{0, \ldots, m-1\}^2$ is the domain of the image, $y \in \mathbb{R}^{m\times m}$ is a signal whose u-th element is $y[u]$, and δ_τ is the translated Dirac delta function $\delta_\tau[t] = \delta[t-\tau]$. In this section, we use $*$ to denote circular convolution and $\star$ to denote circular cross-correlation. Recall that convolution with the translated δ function is equivalent to translation $(x * \delta_\tau)[t] = x[t - \tau \bmod m]$. Incorporating quadratic regularization to prevent overfitting, the problem is to find

$$\arg\min_w \ \frac{1}{2n}\|w \star x - y\|^2 + \frac{\lambda}{2}\|w\|^2 \quad (5)$$

where $n = |\mathcal{U}|$ is the effective number of examples.

The optimal template w must satisfy the system of equations (obtained via the Lagrangian dual, see Appendix C, supplementary material)

$$\begin{cases} k = \frac{1}{n}(x \star x) + \lambda\delta \\ k * \alpha = \frac{1}{n} y \\ w = \alpha \star x \end{cases} \quad (6)$$

where k can be interpreted as the signal that defines a circulant linear kernel matrix, and α is a signal comprised of the Lagrange multipliers of a constrained optimization problem that is equivalent to eq. 5. The solution to eq. 6 can be computed efficiently in the Fourier domain [14],

$$\begin{cases} \widehat{k} = \frac{1}{n}(\widehat{x}^* \circ \widehat{x}) + \lambda\mathbb{1} & \text{(7a)} \\ \widehat{\alpha} = \frac{1}{n}\widehat{k}^{-1} \circ \widehat{y} & \text{(7b)} \\ \widehat{w} = \widehat{\alpha}^* \circ \widehat{x} & \text{(7c)} \end{cases}$$

where we use $\widehat{x} = Fx$ to denote the Discrete Fourier Transform of a variable, x^* to denote the complex conjugate, $\circ$ to denote element-wise multiplication and $\mathbb{1}$ to denote a signal of ones. The inverse of element-wise multiplication is element-wise scalar inversion. Notice that the operations in eq. 7 are more efficiently computed in the Fourier domain, since they involve element-wise operations instead of more expensive convolutions or matrix operators (eq. 6). Moreover, the inverse convolution problem (to find α such that $k * \alpha = \frac{1}{n}y$) is the solution to a diagonal system of equations in the Fourier domain (eq. 7b).

Back-propagation. We adopt the notation that if $x \in \mathcal{X} = \mathbb{R}^n$ is a variable in a computational graph that computes a final scalar loss $\ell \in \mathbb{R}$, then $\nabla_x \ell \in \mathcal{X}$ denotes the vector of partial derivatives $(\nabla_x\ell)_i = \partial\ell/\partial x_i$. If $y \in \mathcal{Y} = \mathbb{R}^m$ is another variable in the graph, which is

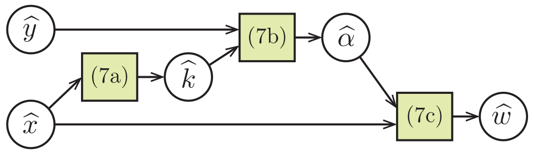

Figure 2: Internal computational graph for the Correlation Filter. The boxes denote functions, which are defined in eq. 7, and the circles denote variables.

computed directly from x according to $y = f(x)$, then the so-called *back-propagation map* for the function f is a linear map from $\nabla_y\ell \in \mathcal{Y}$ to $\nabla_x\ell \in \mathcal{X}$.

Appendix D gives a tutorial review of the mathematical background. In short, the back-propagation map is the linear map which is the adjoint of the differential. This property was used by Ionescu *et al.* [15] to compute back-propagation maps using matrix differential calculus. While they used the matrix inner product $\langle X, Y\rangle = \mathrm{tr}(X^TY)$ to find the adjoint, we use Parseval's theorem, which states that the Fourier transform is unitary (except for a scale factor) and therefore preserves inner products $\langle x, y\rangle \propto \langle \widehat{x}, \widehat{y}\rangle$.

To find the linear map for back-propagation through the Correlation Filter, we first take the differentials of the system of equations in eq. 6 that defines the template w

$$\begin{cases} dk = \frac{1}{n}(dx \star x + x \star dx) \\ dk * \alpha + k * d\alpha = \frac{1}{n}dy \\ dw = d\alpha \star x + \alpha \star dx \end{cases} \quad (8)$$

and then take the Fourier transform of each equation and rearrange to give the differential of each dependent variable in Figure 2 as a linear function (in the Fourier domain) of the differentials of its input variables

$$\begin{cases} \widehat{dk} = \frac{1}{n}(\widehat{dx}^* \circ \widehat{x} + \widehat{x}^* \circ \widehat{dx}) & \text{(9a)} \\ \widehat{d\alpha} = \widehat{k}^{-1} \circ \left[\frac{1}{n}\widehat{dy} - \widehat{dk} \circ \widehat{\alpha}\right] & \text{(9b)} \\ \widehat{dw} = \widehat{d\alpha}^* \circ \widehat{x} + \widehat{\alpha}^* \circ \widehat{dx} \ . & \text{(9c)} \end{cases}$$

Note that while these are complex equations, that is simply because they are the Fourier transforms of real equations. The derivatives themselves are all computed with respect to real variables.

The adjoints of these linear maps define the overall back-propagation map from $\nabla_w\ell$ to $\nabla_x\ell$ and $\nabla_y\ell$. We defer the derivation to Appendix B and present here the final result,

$$\begin{cases} \widehat{\nabla_\alpha\ell} = \widehat{x} \circ (\widehat{\nabla_w\ell})^* \\ \widehat{\nabla_y\ell} = \frac{1}{n}\widehat{k}^{-*} \circ \widehat{\nabla_\alpha\ell} \\ \widehat{\nabla_k\ell} = -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{\nabla_\alpha\ell} \\ \widehat{\nabla_x\ell} = \widehat{\alpha} \circ \widehat{\nabla_w\ell} + \frac{2}{n}\widehat{x} \circ \mathrm{Re}\{\widehat{\nabla_k\ell}\} \ . \end{cases} \quad (10)$$

It is necessary to compute forward Fourier transforms at the start and inverse transforms at the end. The extension to multi-channel images is trivial and given in Appendix E (supplementary material).

As an interesting aside, we remark that, since we have the gradient of the loss with respect to the "desired" response y, it is actually possible to optimize for this parameter rather than specify it manually. However, in practice we did not find learning this parameter to improve the tracking accuracy compared to the conventional choice of a fixed Gaussian response [4, 14].

4. Experiments

The principal aim of our experiments is to investigate the effect of incorporating the Correlation Filter during training. We first compare against the symmetric Siamese architecture of Bertinetto *et al.* [3]. We then compare the end-to-end trained CFNet to a variant where the features are replaced with features that were trained for a different task. Finally, we demonstrate that our method achieves state-of-the-art results.

4.1. Evaluation criteria

Popular tracking benchmarks like VOT [17] and OTB [33, 34] have made all ground truth annotations available and do not enforce a validation/test split. However, in order to avoid overfitting to the test set in design choices and hyperparameter selection, we consider OTB-2013, OTB-50 and OTB-100 as our *test set* and 129 videos from VOT-2014, VOT-2016 and Temple-Color [20] as our *validation set*, excluding any videos which were already assigned to the test set. We perform all of our tracking experiments in Sections 4.2, 4.3 and 4.4 on the validation set with the same set of "natural" hyperparameters, which are reasonable for all methods and not tuned for any particular method.

As in the OTB benchmark [33, 34], we quantify the performance of the tracker on a sequence in terms of the average overlap (intersection over union) of the predicted and ground truth rectangles in all frames. The success rate of a tracker at a given threshold τ corresponds to the fraction of frames in which the overlap with the ground truth is at least τ. This is computed for a uniform range of 100 thresholds between 0 and 1, effectively constructing the cumulative distribution function. Trackers are compared using the area under this curve.

Mimicking the TRE (Temporal Robustness Evaluation) mode of OTB, we choose three equispaced points per sequence and run the tracker from each until the end. Differently from the OTB evaluation, when the target is *lost* (*i.e.* the overlap with the ground truth becomes zero) the tracker is terminated and an overlap of zero is reported for all remaining frames.

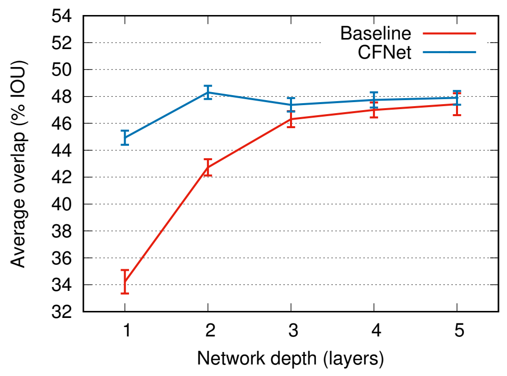

Figure 3: Tracker accuracy for different network depths, on the 129 videos of the validation set. Error bars indicate two standard deviations. Refer to section 4.2 for more details. All figures best viewed in colour.

Despite the large number of videos, we still find that the performance of similarity networks varies considerably as training progresses. To mitigate this effect, we average the final tracking results that are obtained using the parameters of the network at epochs 55, 60, . . . , 95, 100 (the final epoch) to reduce the variance. These ten results are used to estimate the standard deviation of the distribution of results, providing error bars for most figures in this section. While it would be preferable to train all networks to convergence multiple times with different random seeds, this would require significantly more resources.

4.2. Comparison to Siamese baseline

Figures 3 and 4 compare the accuracy of both methods on the validation set for networks of varying depth. The feature extraction network of depth n is terminated after the n-th linear layer, including the following ReLU but not the following pooling layer (if any).

Our baseline diverges slightly from [3] in two ways. Firstly, we reduce the total stride of the network from 8 to 4 (2 at conv1, 2 at pool1) to avoid training Correlation Filters with small feature maps. Secondly, we always restrict the final layer to 32 output channels in order to preserve the high speed of the method with larger feature maps. These changes did not have a negative effect on the tracking performance of SiamFC.

The results show that CFNet is significantly better than the baseline when shallow networks are used to compute features. Specifically, it brings a relative improvement of 31% and 13% for networks of depth one and two respectively. At depths three, four and five, the difference is much less meaningful. CFNet is relatively unaffected by the depth of the network, whereas the performance of the baseline increases steadily and significantly with depth. It seems that the ability of the Correlation Filter to adapt the distance

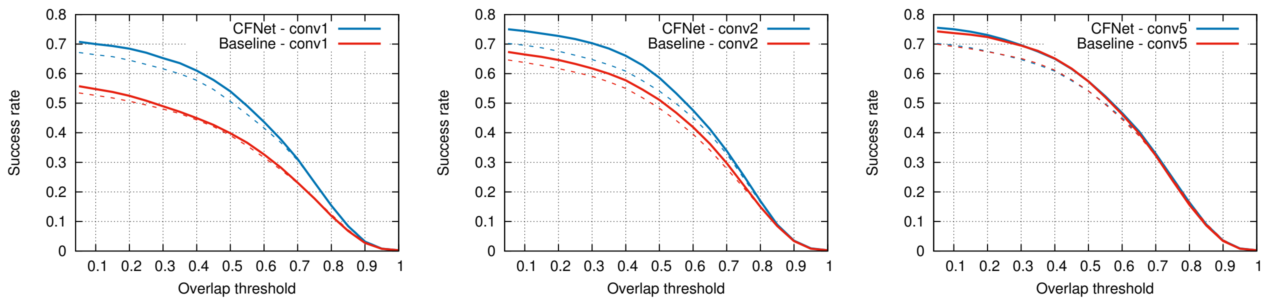

Figure 4: Success rates of rectangle overlap for individual trackers on the validation set. Solid and dotted lines represent methods that update the template with a running average learning rate of 0.01 and 0, respectively.

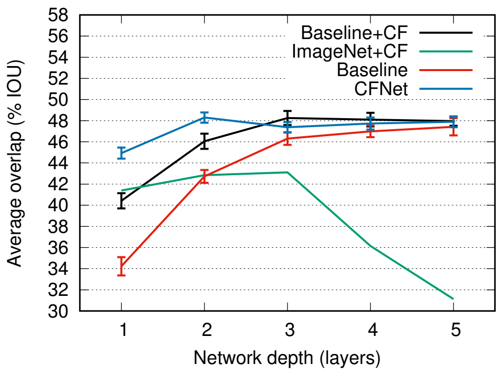

Figure 5: Accuracy of a Correlation Filter tracker when using features obtained via different methods. Error bars indicate two standard deviations. Refer to Section 4.3 for details.

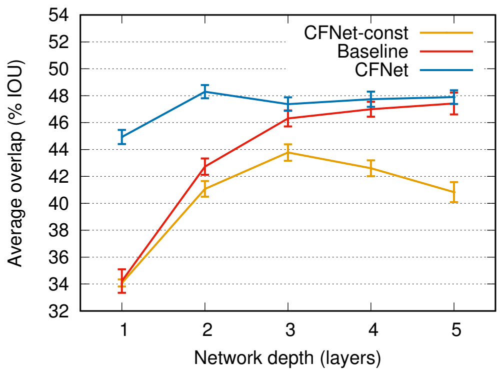

Figure 6: Comparison of CFNet to a "constant" variant of the architecture, in which the Lagrange multipliers do not depend on the image (section 4.4). Error bars indicate two standard deviations.

metric to the content of the training image is less important given a sufficiently expressive embedding function.

The CF layer can be understood to encode prior knowledge of the test-time procedure. This prior may become redundant or even overly restrictive when enough model capacity and data are available. We believe this explains the saturation of CFNet performance when more than two convolutional layers are used.

Figure 4 additionally shows that updating the template is always helpful, for both Baseline and CFNet architectures, at any depth.

4.3. Feature transfer experiment

The motivation for this work was the hypothesis that incorporating the CF during training will result in features that are better suited to tracking with a CF. We now compare our end-to-end trained CFNet to variants that use features from alternative sources: *Baseline+CF* and *ImageNet+CF*. The results are presented in Figure 5.

To obtain the curve *Baseline+CF* we trained a baseline Siamese network of the desired depth and then combined those features with a CF during tracking. Results show that taking the CF into account during offline training is critical at depth one and two. However, it seems redundant when more convolutional layers are added, since using features from the *Baseline* in conjunction with the CF achieves similar performance.

The *ImageNet+CF* variant employs features taken from a network trained to solve the ImageNet classification challenge [28]. The results show that these features, which are often the first choice for combining CFs with CNNs [7, 9, 22, 26, 32, 36], are significantly worse than those learned by *CFNet* and the *Baseline* experiment. The particularly poor performance of these features at deeper layers is somewhat unsurprising, since these layers are expected to have greater invariance to position when trained for classification.

4.4. Importance of adaptation

For a multi-channel CF, each channel p of the template w can be obtained as $w_p = \alpha \star x_p$, where α is itself a function of the exemplar x (Appendix C, supplementary material). To verify the importance of the online adaptation that solv-

		OTB-2013				OTB-50				OTB-100			
		OPE		TRE		OPE		TRE		OPE		TRE	
Method	**speed** (fps.)	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.	IoU	prec.
CFNet-conv1	83	57.8	77.6	58.6	77.6	48.8	65.3	51.0	67.9	53.6	71.3	55.9	72.6
CFNet-conv2	75	61.1	80.7	**64.0**	**84.8**	53.0	70.2	56.5	75.3	56.8	74.8	60.6	79.1
Baseline+CF-conv3	67	61.0	<u>82.2</u>	<u>63.1</u>	<u>83.9</u>	<u>53.8</u>	<u>72.3</u>	**57.4**	**76.7**	**58.9**	<u>77.7</u>	<u>61.1</u>	**79.8**
CFNet-conv5	43	61.1	80.3	62.6	82.5	**53.9**	**73.2**	<u>56.6</u>	<u>75.9</u>	58.6	<u>77.7</u>	60.8	78.8
Baseline-conv5	52	**61.8**	80.6	**64.0**	83.7	51.7	68.3	56.1	74.2	<u>58.8</u>	76.9	**61.6**	<u>79.7</u>
SiamFC-3s [3]		60.7	81.0	61.8	82.2	51.6	69.2	55.5	75.2	58.2	77.0	60.5	79.5
Staple [2]		60.0	79.3	61.7	80.3	50.9	68.1	54.1	72.6	58.1	**78.4**	60.4	78.9
LCT [23]		<u>61.2</u>	**86.2**	59.4	81.3	49.2	69.1	49.5	67.4	56.2	76.2	56.9	74.5
SAMF [19]		–	–	–	–	46.2	63.9	51.4	70.9	53.9	74.6	57.7	77.6
DSST [6]		55.4	74.0	56.6	73.8	45.2	60.4	48.4	64.1	51.3	68.0	–	–

Table 1: Perfomance as overlap (IoU) and precision produced by the OTB toolkit for the OTB-2013, OTB-50 and OTB-100 datasets. The **first** and <u>second</u> best results are highlighted in each column. For details refer to Section 4.5.

ing a ridge regression problem at test time should provide, we propose a "constant" version of the Correlation Filter (*CFNet-const*) where the vector of Lagrange multipliers α is instead a parameter of the network that is learned offline and remains fixed at test time.

Figure 6 compares CFNet to its constant variant. CFNet is consistently better, demonstrating that in order to improve over the baseline Siamese network it is paramount to back-propagate through the solution to the inverse convolution problem that defines the Lagrange multipliers.

4.5. Comparison with the state-of-the-art

We use the OTB-2013/50/100 benchmarks to confirm that our results are on par with the state-of-the-art. All numbers in this section are obtained using the OTB toolkit [33]. We report the results for the three best instantiations of CFNet from Figure 5 (*CFNet-conv2*, *CFNet-conv5*, *Baseline+CF-conv3*), the best variant of the baseline (*Baseline-conv5*) and the most promising single-layer network (*CFNet-conv1*). We compare our methods against state-of-the-art trackers that can operate in real-time: SiamFC-3s [3], Staple [2] and LCT [23]. We also include the recent SAMF [19] and DSST [6] for reference.

For the evaluation of this section, we use a different set of tracking hyperparameters per architecture, chosen to maximize the performance on the validation set after a random search of 300 iterations. More details are provided in the supplementary material. For the few greyscale sequences present in OTB, we re-train each architecture using exclusively greyscale images.

Both overlap (IoU) and precision scores [34] are reported for OPE (one pass) and TRE (temporal robustness) evaluations. For OPE, the tracker is simply run once on each sequence, from the start to the end. For TRE, the tracker is instead started from twenty different starting points, and run until the end from each. We observed that this ensures more robust and reliable results compared to OPE.

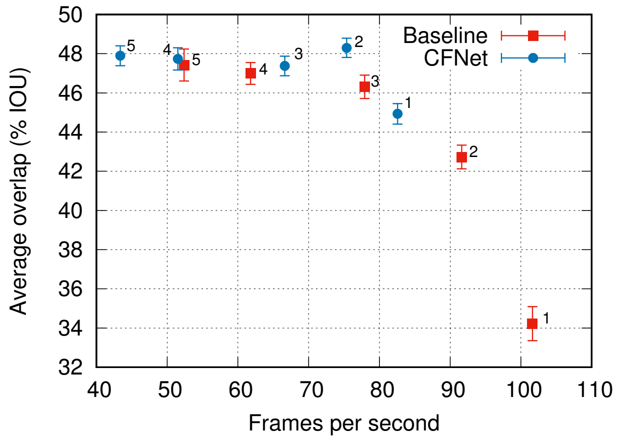

Figure 7: Tracker accuracy versus speed for CFNet and Siamese baseline. Labels indicate network depth. CFNet enables better accuracy to be obtained at higher speeds using shallower networks. Error bars indicate two standard deviations. Refer to section 4.6 for details.

Similarly to the analysis on the validation set, *CFNet-conv2* is among the top performers and its accuracy rivals that of *Baseline-conv5*, which possesses approximately 30× as many parameters. In general, our best proposed CFNet variants are superior (albeit modestly) to the state-of-the-art. In order to focus on the impact of our contribution, we decided to avoid including orthogonal improvements which can often be found in the tracking literature (*e.g.* bounding box regression [26], ensembling of multiple cues [23, 2], optical flow [29]).

4.6. Speed and practical benefits

The previous sections have demonstrated that there is a clear benefit to integrating Correlation Filters into Siamese networks when the feature extraction network is relatively shallow. Shallow networks are practically advantageous in that they require fewer operations and less memory to evaluate and store. To understand the trade-off, Figure 7 reports the speed and accuracy of both CFNet and the baseline for varying network depth[3].

This plot suggests that the two-layer CFNet could be the most interesting variant for practitioners requiring an accurate tracking algorithm that operates at high framerates. It runs at 75 frames per second and has less than 4% of the parameters of the five-layer baseline, requiring only 600kB to store. This may be of particular interest for embedded devices with limited memory. In contrast, methods like DeepSRDCF [7] and C-COT [9], which use out-of-the-box deep features for the Correlation Filter, run orders of magnitude slower. Even the one-layer CFNet remains competitive despite having less than 1% of the parameters of the five-layer baseline and requiring under 100kB to store.

5. Conclusion

This work proposes the Correlation Filter network, an asymmetric architecture that back-propagates gradients through an online learning algorithm to optimize the underlying feature representation. This is made feasible by establishing an efficient back-propagation map for the solution to a system of circulant equations.

Our empirical investigation reveals that, for a sufficiently deep Siamese network, adding a Correlation Filter layer does not significantly improve the tracking accuracy. We believe this is testament to the power of deep learning given sufficient training data. However, incorporating the Correlation Filter into a similarity network during training does enable shallow networks to rival their slower, deeper counterparts.

Future research may include extensions to account for adaptation over time, and back-propagating gradients through learning problems for related tasks such as one-shot learning and domain adaptation.

A. Implementation details

We follow the procedure of [3] to minimize the loss (equation 2) through SGD, with the Xavier-improved parameters initialization and using mini-batches of size 8. We use all the 3862 training videos of ImageNet Video [28], containing more than 1 million annotated frames, with multiple objects per frame. Training is conducted for 100 epochs, each sampling approximately 12 pairs (x'_i, z'_i) from each video, randomly extracted so that they are at most 100 frames apart.

During tracking, a spatial cosine window is multiplied with the score map to penalize large displacements. Tracking in scale space is achieved by evaluating the network at the scale of the previous object and at one adjacent scale on either side, with a geometric step of 1.04. Updating the scale is discouraged by multiplying the responses of the scaled object by 0.97. To avoid abrupt transitions of object size, scale is updated using a rolling average with learning rate 0.6.

[3]The speed was measured using a 4.0GHz Intel i7 CPU and an NVIDIA Titan X GPU.

B. Back-propagation for the Correlation Filter

As described in Appendix D (supplementary material), the back-propagation map is the adjoint of the linear maps that is the differential. These linear maps for the Correlation Filter are presented in eq. 9. We are free to obtain these adjoint maps in the Fourier domain since Parseval's theorem provides the preservation of inner products. Let J_1 denote the map $dx \mapsto dk$ in eq. 9a. Hence manipulation of the inner product

$$\begin{aligned}\langle F dk, F J_1(dx)\rangle &= \left\langle \widehat{dk}, \tfrac{1}{n}(\widehat{dx}^* \circ \hat{x} + \hat{x}^* \circ \widehat{dx})\right\rangle \\ &= \tfrac{1}{n}\left[\langle \widehat{dx}, \widehat{dk}^* \circ \hat{x}\rangle + \langle \widehat{dk} \circ \hat{x}, \widehat{dx}\rangle\right] \\ &= \left\langle \widehat{dx}, \tfrac{2}{n}\,\mathrm{Re}\{\widehat{dk}\} \circ \hat{x}\right\rangle \end{aligned} \tag{11}$$

gives the back-propagation map

$$\widehat{\nabla_x \ell} = \tfrac{2}{n}\hat{x} \circ \mathrm{Re}\{\widehat{\nabla_k \ell}\} \ . \tag{12}$$

Similarly, for the linear map $dk, dy \mapsto d\alpha$ in eq. 9b,

$$\begin{aligned}&\langle F d\alpha, F J_2(dk, dy)\rangle = \left\langle \widehat{d\alpha}, \hat{k}^{-1}[\tfrac{1}{n}\widehat{dy} - \widehat{dk} \circ \hat{\alpha}]\right\rangle \\ &= \left\langle \tfrac{1}{n}\hat{k}^{-*} \circ \widehat{d\alpha}, \widehat{dy}\right\rangle + \left\langle -\hat{k}^{-*} \circ \hat{\alpha}^* \circ \widehat{d\alpha}, \widehat{dk}\right\rangle \ , \end{aligned} \tag{13}$$

the back-propagation maps are

$$\widehat{\nabla_y \ell} = \tfrac{1}{n}\hat{k}^{-*} \circ \widehat{\nabla_\alpha \ell} \tag{14}$$

$$\widehat{\nabla_k \ell} = -\hat{k}^{-*} \circ \hat{\alpha}^* \circ \widehat{\nabla_\alpha \ell} \ , \tag{15}$$

and for the linear map $dx, d\alpha \mapsto dw$ in eq. 9c,

$$\begin{aligned}&\langle F dw, F J_3(dx, d\alpha)\rangle = \langle \widehat{dw}, \widehat{d\alpha}^* \circ \hat{x} + \hat{\alpha}^* \circ \widehat{dx}\rangle \\ &= \langle \widehat{d\alpha}, \widehat{dw}^* \circ \hat{x}\rangle + \langle \widehat{dw} \circ \hat{\alpha}, \widehat{dx}\rangle \ , \end{aligned} \tag{16}$$

the back-propagation maps are

$$\widehat{\nabla_\alpha \ell} = \hat{x} \circ (\widehat{\nabla_w \ell})^* \ , \tag{17}$$

$$\widehat{\nabla_x \ell} = \hat{\alpha} \circ \widehat{\nabla_w \ell} \ . \tag{18}$$

The two expressions for $\nabla_x \ell$ above are combined to give the back-propagation map for the entire Correlation Filter block in eq. 10.

C. Correlation Filter formulation

C.1. Kernel linear regression

First, consider the general linear regression problem of learning the weight vector w that best maps each of n example input vectors $x_i \in \mathbb{R}^d$ to their target $y_i \in \mathbb{R}$. The squared error can be expressed

$$\frac{1}{2n}\sum_{i=1}^{n}(x_i^T w - y_i)^2 = \frac{1}{2n}\|X^T w - y\|^2 \tag{19}$$

where $X \in \mathbb{R}^{d\times n}$ is a matrix whose columns are the example vectors and $y \in \mathbb{R}^n$ is a vector of the targets. Incorporating regularization, the problem is

$$\arg\min_w \quad \tfrac{1}{2n}\|X^T w - y\|^2 + \tfrac{\lambda}{2}\|w\|^2 \ . \tag{20}$$

Kernel linear regression can be developed by writing this as a constrained optimization problem

$$\begin{aligned} \arg\min_{w,r} \quad & \tfrac{1}{2n}\|r\|^2 + \tfrac{\lambda}{2}\|w\|^2 \\ \text{subject to} \quad & r = X^T w - y \end{aligned} \tag{21}$$

and then finding a saddle point of the Lagrangian

$$L(w,r,\upsilon) = \tfrac{1}{2n}\|r\|^2 + \tfrac{\lambda}{2}\|w\|^2 + \upsilon^T(r - X^T w + y) \ . \tag{22}$$

The final solution can be obtained from the dual variable

$$w = \tfrac{1}{\lambda}X\upsilon \tag{23}$$

and the solution to the dual problem is

$$\upsilon = \tfrac{\lambda}{n}K^{-1}y \tag{24}$$

where $K = \frac{1}{n}X^T X + \lambda I$ is the regularized kernel matrix. It is standard to introduce a scaled dual variable $\alpha = \frac{1}{\lambda}\upsilon$ that defines w as a weighted combination of examples

$$w = X\alpha = \sum_{i=1}^{n}\alpha_i x_i \quad \text{with} \quad \alpha = \frac{1}{n}K^{-1}y \ . \tag{25}$$

The kernel matrix is $n \times n$ and therefore the dual solution is more efficient than the primal solution, which requires inversion of a $d \times d$ matrix, when the number of features d exceeds the number of examples n.

C.2. Single-channel Correlation Filter

Given a scalar-valued example signal x with domain $\mathcal{U}$ and corresponding target signal y, the Correlation Filter w is the scalar-valued signal

$$\arg\min_w \quad \frac{1}{2n}\|w \star x - y\|^2 + \frac{\lambda}{2}\|w\|^2 \tag{26}$$

where signals are treated as vectors in $\mathbb{R}^{\mathcal{U}}$ and the circular cross-correlation of two signals $w \star x$ is defined

$$(w \star x)[u] = \sum_{t\in\mathcal{U}} w[t]x[u + t \bmod m] \quad \forall u \in \mathcal{U} \ . \tag{27}$$

The solution from the previous section can then be used by defining X to be the matrix in $\mathbb{R}^{\mathcal{U}\times\mathcal{U}}$ such that $X^T w = w \star x$. It follows that the kernel matrix K belongs to $\mathbb{R}^{\mathcal{U}\times\mathcal{U}}$ and the dual variable α is a signal in $\mathbb{R}^{\mathcal{U}}$.

The key to the correlation filter is that the circulant structure of X enables the solution to be computed efficiently in the Fourier domain. The matrix X has elements $X[u,t] = x[u + t \bmod m]$. Since the matrix X is symmetric, the template w is obtained as cross-correlation

$$w = X\alpha = \alpha \star x \ . \tag{28}$$

The linear map defined by the kernel matrix K is equivalent to convolution with a signal k

$$Kz = k * z \quad \forall z \tag{29}$$

which is defined $k = \frac{1}{n}x \star x + \lambda\delta$, since

$$\begin{aligned} \forall z : FX^T Xz &= F((z \star x) \star x) \\ &= \hat{z} \circ \hat{x}^* \circ \hat{x} = F(z * (x \star x)) \ . \end{aligned} \tag{30}$$

Therefore the solution is defined by the equations

$$\begin{cases} k = \frac{1}{n}x \star x + \lambda\delta \\ k * \alpha = \frac{1}{n}y \\ w = \alpha \star x \end{cases} \tag{31}$$

and the template can be computed efficiently in the Fourier domain

$$\begin{cases} \hat{k} = \frac{1}{n}\hat{x}^* \circ \hat{x} + \lambda\mathbb{1} \\ \hat{\alpha} = \frac{1}{n}\hat{k}^{-1} \circ \hat{y} \\ \hat{w} = \hat{\alpha}^* \circ \hat{x} \ . \end{cases} \tag{32}$$

C.3. Multi-channel Correlation Filter

There is little advantage to the dual solution when training a single-channel Correlation Filter from the circular shifts of a single base example. However, the dual formulation is much more efficient in the multi-channel case [14].

For signals with k channels, each multi-channel signal is a collection of scalar-valued signals $x = (x_1, \ldots, x_k)$, and the data term becomes

$$\|\textstyle\sum_p w_p \star x_p - y\|^2 = \|\sum_p X_p^T w_p - y\|^2 \tag{33}$$

and each channel of the template is obtained from the dual variables

$$w_p = X_p\alpha = \alpha \star x_p \tag{34}$$

The solution to the dual problem is still $\alpha = \frac{1}{n}K^{-1}y$, however the kernel matrix is now given

$$K = \tfrac{1}{n}\textstyle\sum_p X_p^T X_p + \lambda I \tag{35}$$

and the linear map defined by this matrix is equivalent to convolution with the signal

$$k = \tfrac{1}{n}\textstyle\sum_p x_p \star x_p + \lambda\delta \ . \tag{36}$$

Therefore the solution is defined by the equations

$$\begin{cases} k = \frac{1}{n}\sum_p x_p \star x_p + \lambda\delta \\ k * \alpha = \frac{1}{n}y \\ w_p = \alpha \star x_p \quad \forall p \end{cases} \tag{37}$$

and the template can be computed efficiently in the Fourier domain

$$\begin{cases} \widehat{k} = \frac{1}{n}\sum_p \widehat{x}_p^* \circ \widehat{x}_p + \lambda\mathbb{1} \\ \widehat{\alpha} = \frac{1}{n}\widehat{k}^{-1} \circ \widehat{y} \\ \widehat{w}_p = \widehat{\alpha}^* \circ \widehat{x}_p \quad \forall p \ . \end{cases} \tag{38}$$

It is critical that the computation scales only linearly with the number of channels.

D. Adjoint of the differential

Consider a computational graph that computes a scalar loss $\ell \in \mathbb{R}$. Within this network, consider an intermediate function that computes $y = f(x)$ where $x \in \mathcal{X} = \mathbb{R}^m$ and $y \in \mathcal{Y} = \mathbb{R}^n$. Back-propagation computes the gradient with respect to the input $\nabla_x \ell \in \mathcal{X}$ from the gradient with respect to the output $\nabla_y \ell \in \mathcal{Y}$.

The derivative $\partial f(x)/\partial x$ is a matrix in $\mathbb{R}^{n\times m}$ whose ij-th element is the partial derivative $\partial f_i(x)/\partial x_j$. This matrix relates the gradients according to

$$(\nabla_x \ell)^T = \frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial y}\frac{\partial y}{\partial x} = (\nabla_y \ell)^T \frac{\partial f(x)}{\partial x} \tag{39}$$

From this it is evident that the back-propagation map is the linear map which is the adjoint of that defined by the derivative. That is, if the derivative defines the linear map

$$J(u) = \frac{\partial f(x)}{\partial x} u \tag{40}$$

then the back-propagation map is the unique linear map J^* that satisfies

$$\langle J^*(v), u\rangle = \langle v, J(u)\rangle \quad \forall u \in \mathcal{X}, v \in \mathcal{Y} \tag{41}$$

and the gradient with respect to the input is obtained $\nabla_x \ell = J^*(\nabla_y \ell)$. This is the core of reverse-mode differentiation [12].

An alternative way to obtain the linear map defined by the derivative is to use differential calculus. Whereas the *derivative* represents this linear map as a matrix with respect to the standard bases, the *differential* represents the linear map as an expression $df(x; dx)$. This is valuable for working with variables that possess more interesting structure than simple vectors. This technique has previously been used for matrix structured back-propagation [15]. In this paper, we use it for circulant structured back-propagation.

E. Back-propagation for multi-channel case

The differentials of the equations that define the multi-channel CF in eq. 37 are

$$\begin{cases} dk = \frac{1}{n}\sum_p (dx_p \star x_p + x_p \star dx_p) \\ dk * \alpha + k * d\alpha = \frac{1}{n}dy \\ dw_p = d\alpha \star x_p + \alpha \star dx_p \quad \forall p \ , \end{cases} \tag{42}$$

and taking the Fourier transforms of these equations gives

$$\begin{cases} \widehat{dk} = \frac{1}{n}\sum_p \left(\widehat{dx}_p^* \circ \widehat{x}_p + \widehat{x}_p^* \circ \widehat{dx}_p\right) \\ \widehat{d\alpha} = \widehat{k}^{-1} \circ \left[\frac{1}{n}\widehat{dy} - \widehat{dk} \circ \widehat{\alpha}\right] \\ \widehat{dw}_p = \widehat{d\alpha}^* \circ \widehat{x}_p + \widehat{\alpha}^* \circ \widehat{dx}_p \quad \forall p \ . \end{cases} \tag{43}$$

Now, to find the adjoint of the map $dx \mapsto dk$, we re-arrange the inner product

$$\begin{aligned} \langle F dk, F J_1(dx)\rangle &= \left\langle \widehat{dk}, \tfrac{1}{n}\textstyle\sum_p \left(\widehat{dx}_p^* \circ \widehat{x}_p + \widehat{x}_p^* \circ \widehat{dx}_p\right)\right\rangle \\ &= \tfrac{1}{n}\textstyle\sum_p \left[\langle \widehat{dx}_p, \widehat{dk}^* \circ \widehat{x}_p\rangle + \langle \widehat{dk} \circ \widehat{x}_p, \widehat{dx}_p\rangle\right] \\ &= \textstyle\sum_p \langle \widehat{dx}_p, \tfrac{2}{n}\operatorname{Re}\{\widehat{dk}\} \circ \widehat{x}_p\rangle \end{aligned} \tag{44}$$

to give the back-propagation map

$$\widehat{\nabla_{x_p}\ell} = \tfrac{2}{n}\widehat{x}_p \circ \operatorname{Re}\{\widehat{\nabla_k \ell}\} \quad \forall p \ . \tag{45}$$

The linear map $dk, dy \mapsto d\alpha$ is identical to the single-channel case. To find the adjoint of the map $dx, d\alpha \mapsto dw$, we examine the inner-product

$$\begin{aligned} \langle dw, J_3(dx, d\alpha)\rangle &= \textstyle\sum_p \langle \widehat{dw}_p, \widehat{d\alpha}^* \circ \widehat{x}_p + \widehat{\alpha}^* \circ \widehat{dx}_p\rangle \\ &= \left\langle \widehat{d\alpha}, \textstyle\sum_p \widehat{dw}_p^* \circ \widehat{x}_p\right\rangle + \textstyle\sum_p \langle \widehat{dw}_p \circ \widehat{\alpha}, \widehat{dx}_p\rangle \ , \end{aligned} \tag{46}$$

giving the back-propagation maps

$$\widehat{\nabla_\alpha \ell} = \textstyle\sum_p \widehat{x}_p \circ (\widehat{\nabla_{w_p}\ell})^* \ , \tag{47}$$

$$\widehat{\nabla_{x_p}\ell} = \widehat{\alpha} \circ \widehat{\nabla_{w_p}\ell} \quad \forall p \ . \tag{48}$$

Finally, combining these results gives the procedure for back-propagation in the multi-channel case

$$\begin{cases} \widehat{\nabla_\alpha \ell} = \sum_p \widehat{x}_p \circ (\widehat{\nabla_{w_p}\ell})^* \\ \widehat{\nabla_y \ell} = \frac{1}{n}\widehat{k}^{-*} \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_k \ell} = -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_{x_p}\ell} = \widehat{\alpha} \circ \widehat{\nabla_{w_p}\ell} + \frac{2}{n}\widehat{x}_p \circ \operatorname{Re}\{\widehat{\nabla_k \ell}\} \quad \forall p \ . \end{cases} \tag{49}$$

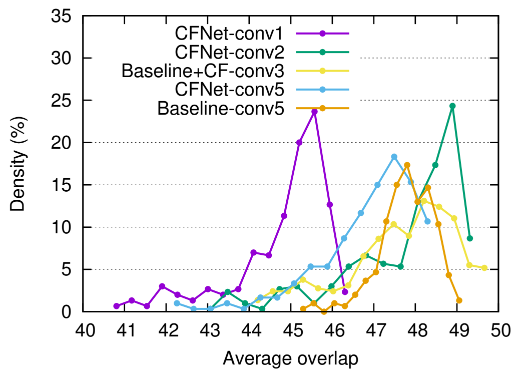

Figure 8: Empirical distribution of the average overlap for the hyperparameter search.

Again, it is important that the computation scales only linearly with the number of channels.

F. Hyperparameter optimization

The hyperparameters that define the simplistic tracking algorithm have a significant impact on the tracking accuracy. These include parameters such as the penalty for changes in scale and position and the learning rate of the template average. Choosing hyperparameters is a difficult optimization problem: we cannot use gradient descent because the function is highly discontinuous, and each function evaluation is expensive because it involves running a tracker on every sequence from multiple starting points.

For the experiments of the main paper, where we sought to make a fair comparison of different architectures, we therefore used a *natural* choice of hyperparameters that were not optimized for any particular architecture. Ideally, we would use the optimal hyperparameters for each variant, except it would have been computationally prohibitive to perform this optimization for every point in every graph in the main paper (multiple times for the points with error bars).

To achieve results that are competitive with the state-of-the-art, however, it is necessary to optimize the parameters of the tracking algorithm (on a held-out validation set).

To find optimal hyperparameters, we use random search with a uniform distribution on a reasonable range for each parameter. Specifically, we sample 300 random vectors of hyperparameters and run the evaluation described in Section 4.1 on the 129 videos of our validation set. Each method is then evaluated once on the test sets (OTB-2013, OTB-50 and OTB-100) using the hyperparameter vector which gave the best results on the validation set (specified in Table 2). We emphasize that, even though the ground-truth labels are available for the videos in the benchmarks, we do not choose hyperparameters to optimize the results on the benchmarks, as this would not give a meaningful estimate of the generalization ability of the method.

Note that this random search is performed after training and is only used to choose parameters for the online tracking algorithm. The same network is used for all random samples. The training epoch with the best tracking results on the validation set (with natural tracking parameters) is chosen.

Figure 8 shows, for each method, the empirical distribution of results (in terms of average overlap) that is induced by the distribution of tracking parameters in random search.

G. Detailed results on the OTB benchmarks

Figures 9 to 14 show the curves produced by the OTB toolkit[4] for OTB-2013/50/100, of which we presented a summary in the main paper.

Acknowledgements. This research was supported by Apical Ltd., EPSRC grant Seebibyte EP/M013774/1 and ERC grants ERC-2012-AdG 321162-HELIOS, HELIOS-DFR00200, “Integrated and Detailed Image Understanding” (EP/L024683/1) and ERC 677195-IDIU.

References

[1] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, pages 523–531, 2016. 2

[2] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, pages 1401–1409, 2016. 2, 7

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional Siamese networks for object tracking. In *ECCV Workshops*, pages 850–865, 2016. 1, 2, 3, 5, 7, 8

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2, 3, 5

[5] K. Chen and W. Tao. Once for all: A two-flow convolutional neural network for visual tracking. *arXiv preprint arXiv:1604.07507*, 2016. 1

[6] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 2, 3, 7

[7] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshops*, pages 58–66, 2015. 2, 3, 6, 8

[4] The precision plots in this version of the paper are slightly different to those in the version submitted to CVPR. Whereas in the CVPR version, we adopted the “area under curve” precision metric, here we have used the standard precision metric with a single threshold of 20 pixels. This has little effect on the ordering of the trackers and all observations remained valid.

	avg. overlap	**best overlap**	scale step	scale penalty	scale l.r.	win. weight	template l.r.
CFNet-conv1	44.8	46.5	1.0355	0.9825	0.700	0.2375	0.0058
CFNet-conv2	**47.8**	<u>49.5</u>	1.0575	0.9780	0.520	0.2625	0.0050
Baseline+CF-conv3	<u>47.7</u>	**49.9**	1.0340	0.9820	0.660	0.2700	0.0080
CFNet-conv5	46.9	48.5	1.0310	0.9815	0.525	0.2000	0.0110
Baseline-conv5	**47.8**	49.2	1.0470	0.9825	0.680	0.1750	0.0102

Table 2: Average and best overlap scores over 300 random sets of hyperparameters. Values of hyperparameters associated to the best performance are also reported. These parameters describe: the geometric step to use in scale search, the multiplicative penalty to apply for changing scale, the learning rate for updating the scale, the weight of an additive cosine window that penalizes translation, and the learning rate for the template average.

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, pages 4310–4318, 2015. 2, 3

[9] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, pages 472–488, 2016. 2, 6, 8

[10] J. A. Fernandez and B. Vijayakumar. Zero-aliasing correlation filters. In *International Symposium on Image and Signal Processing and Analysis 2013*, pages 101–106, 2013. 2

[11] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. 2

[12] A. Griewank and A. Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM, 2008. 10

[13] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, pages 749–765. Springer, 2016. 1, 2

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 37(3):583–596, 2015. 2, 3, 4, 5, 9

[15] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix back-propagation for deep networks with structured layers. In *ICCV*, pages 2965–2973, 2015. 2, 4, 10

[16] H. Kiani Galoogahi, T. Sim, and S. Lucey. Correlation filters with limited boundaries. In *CVPR*, pages 4630–4638, 2015. 2, 3

[17] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, G. Häger, A. Lukežič, G. Fernández, et al. The Visual Object Tracking VOT2016 challenge results. In *ECCV Workshops*. Springer, 2016. 5

[18] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. In *CVPR Workshops*, pages 33–40, 2016. 1

[19] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV*, pages 254–265, 2014. 2, 7

[20] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: Algorithms and benchmark. *IEEE Transactions on Image Processing*, 24(12):5630–5644, 2015. 5

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[22] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, pages 3074–3082, 2015. 2, 3, 6

[23] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, pages 5388–5396, 2015. 2, 3, 7

[24] D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015. 2

[25] I. Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016. 2

[26] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR 2016*, pages 4293–4302, 2016. 1, 6, 7

[27] A. Rodriguez, V. N. Boddeti, B. V. K. V. Kumar, and A. Mahalanobis. Maximum margin correlation filter: A new approach for localization and classification. *IEEE Transactions on Image Processing*, 22(2):631–643, 2013. 2

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6, 8

[29] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, pages 1420–1429, 2016. 1, 2, 7

[30] J. Valmadre, S. Sridharan, and S. Lucey. Learning detectors quickly with stationary statistics. In *ACCV*, pages 99–114. Springer, 2014. 3

[31] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, 2016. 2

[32] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015. 1, 2, 6

[33] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013. 5, 7

[34] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 5, 7

[35] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*, pages 2528–2535, 2010. 2

[36] M. Zhai, M. J. Roshtkhari, and G. Mori. Deep learning of appearance models for online object tracking. *arXiv preprint arXiv:1607.02568*, 2016. 1, 6

[37] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529–1537, 2015. 2

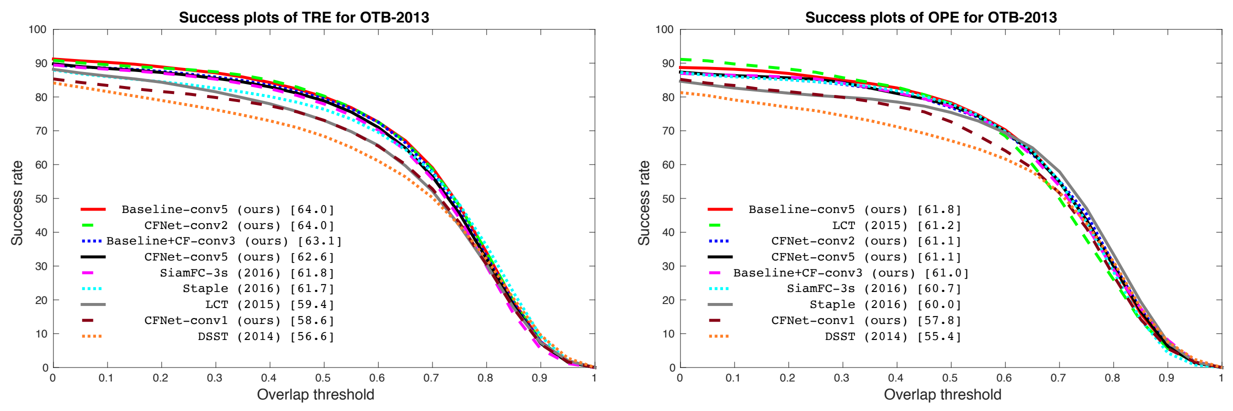

Figure 9: OTB-2013 success rate.

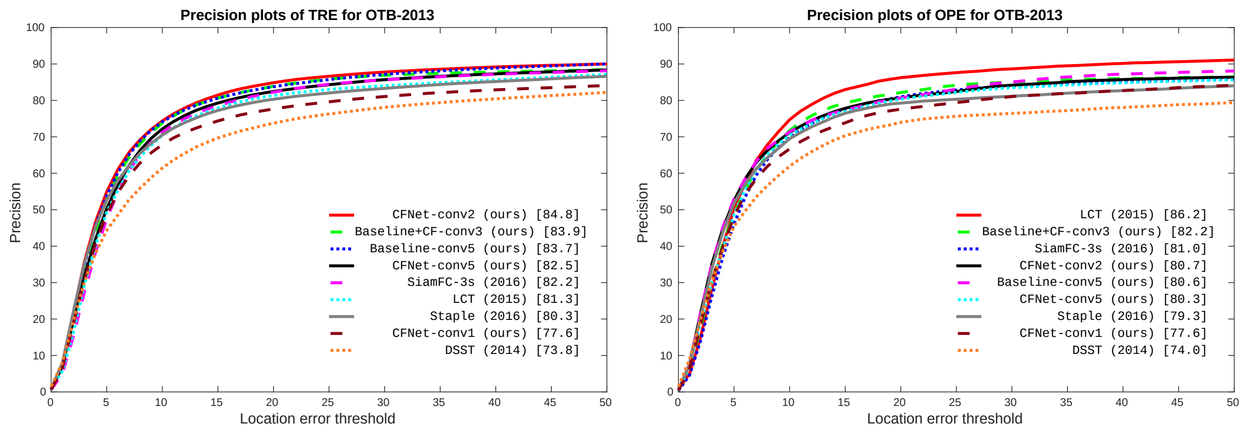

Figure 10: OTB-2013 precision.

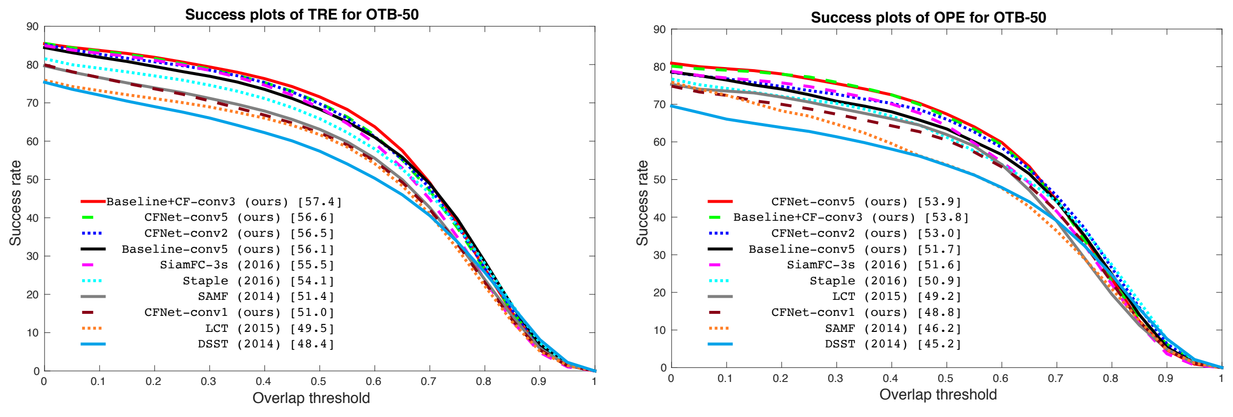

Figure 11: OTB-50 success rate.

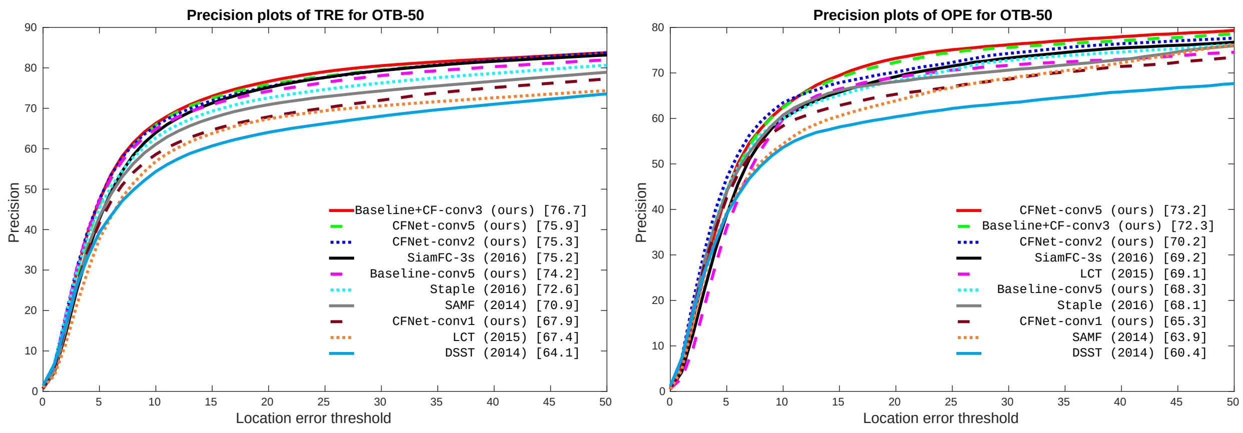

Figure 12: OTB-50 precision.

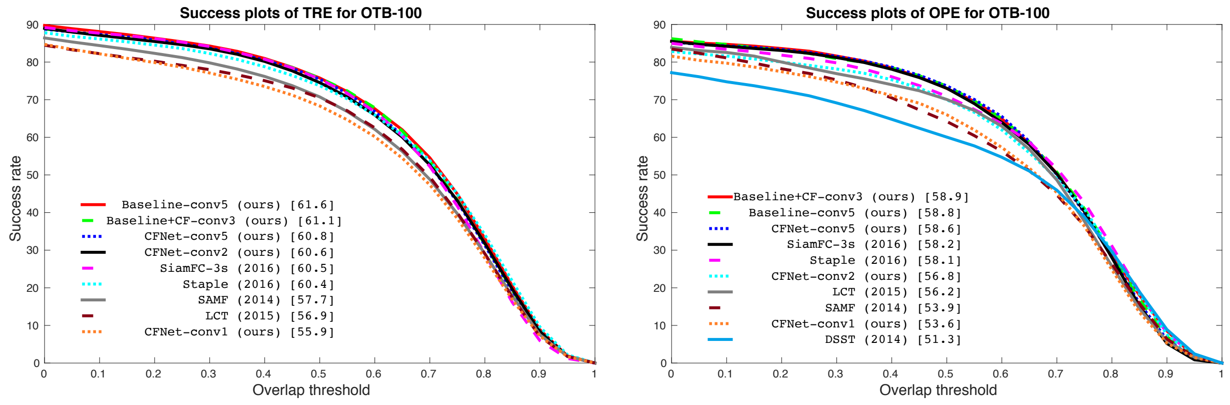

Figure 13: OTB-100 success rate.

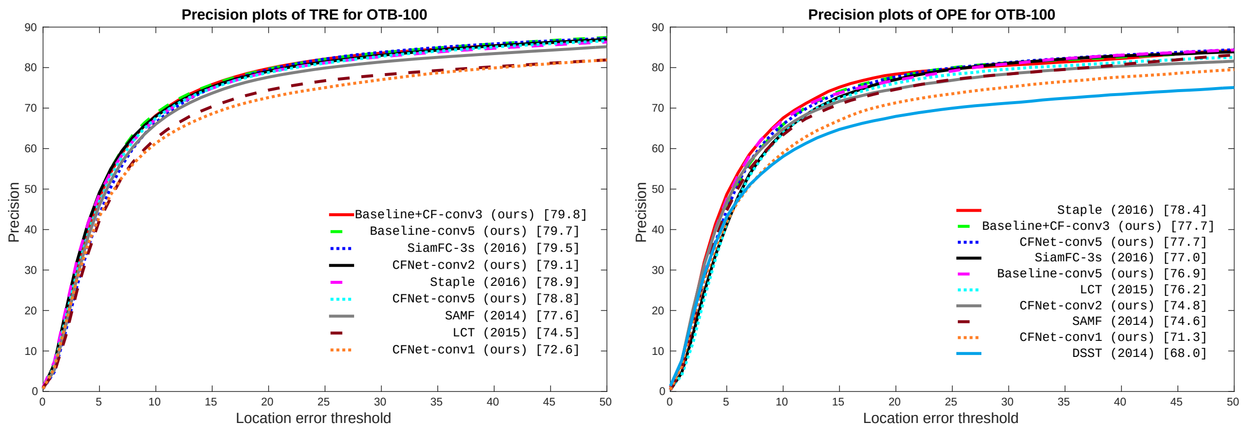

Figure 14: OTB-100 precision.