Unveiling the Power of Deep Tracking

Goutam Bhat[1], Joakim Johnander[1,2], Martin Danelljan[1],
Fahad Shahbaz Khan[1,3], and Michael Felsberg[1]

[1] CVL, Department of Electrical Engineering, Linköping University, Sweden
[2] Zenuity, Sweden
[3] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE

Abstract. In the field of generic object tracking numerous attempts have been made to exploit deep features. Despite all expectations, deep trackers are yet to reach an outstanding level of performance compared to methods solely based on handcrafted features. In this paper, we investigate this key issue and propose an approach to unlock the true potential of deep features for tracking. We systematically study the characteristics of both deep and shallow features, and their relation to tracking accuracy and robustness. We identify the limited data and low spatial resolution as the main challenges, and propose strategies to counter these issues when integrating deep features for tracking. Furthermore, we propose a novel adaptive fusion approach that leverages the complementary properties of deep and shallow features to improve both robustness *and* accuracy. Extensive experiments are performed on four challenging datasets. On VOT2017, our approach significantly outperforms the top performing tracker from the challenge with a relative gain of 17% in EAO.

1 Introduction

Generic object tracking is the problem of estimating the trajectory of a target in a video, given only its initial state. The problem is particularly difficult, primarily due to the limited training data available to learn an appearance model of the target *online*. Existing methods rely on rich feature representations to address this fundamental challenge. While handcrafted features have long been employed for this task, recent focus has been shifted towards deep features. The advantages of deep features being their ability to encode high-level information, invariant to complex appearance changes and clutter.

Despite the outstanding success of deep learning in a variety of computer vision tasks, its impact in generic object tracking has been limited. In fact, trackers based on handcrafted features [1, 7, 8, 22, 37] still provide competitive results, even outperforming many deep trackers on standard benchmarks [16, 36]. Moreover, contrary to the trend in image classification, object trackers do not tend to benefit from deeper and more sophisticated network architectures (see figure 1). In this work, we investigate the reasons behind the limited success of deep networks in visual object tracking.

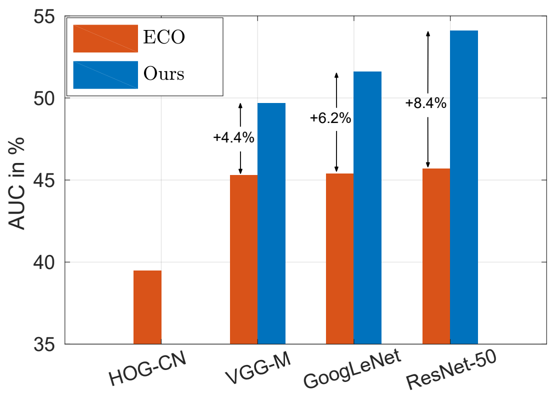


Fig. 1: Tracking performance on the Need for Speed dataset [12] when using deep features extracted from different networks. In all cases, we employ the same shallow representation, consisting of HOG and Color Names. The baseline ECO [7] does not benefit from more powerful network architectures, e.g. ResNet. Instead, our approach is able to exploit more powerful representations, achieving a consistent gain going from handcrafted features towards more powerful network architectures.

We distinguish two key challenges generally encountered when integrating deep features into visual tracking models. Firstly, compared to traditional handcrafted approaches, it is well known that deep models are data-hungry. This becomes a major obstacle in the visual tracking scenario, where training data is extremely scarce and a robust model must be learned from a single labeled frame. Even though pre-trained deep networks are frequently employed, the target model must learn the discriminative activations possessing invariance towards *unseen* appearance changes.

The second challenge for deep features is accurate target prediction. Not only is precise target localization crucial for tracking performance, it also affects the learning of the model since new frames are annotated by the tracker itself. As a result, inaccurate predictions may lead to model drift and eventual tracking failure. Deep convolutional layers generally trade spatial resolution for increased high-level invariance to account for appearance changes. Consequently, many trackers complement the deep representation with shallow-level activations [11, 23] or handcrafted features [7] for improved localization accuracy. This raises the question of how to optimally fuse the fundamentally different properties of shallow and deep features in order to achieve both accuracy *and* robustness.

Contributions: In this paper, we analyze the influential characteristics of deep and shallow features for visual tracking. This is performed by (i) systematically studying the impact of a variety of data augmentation techniques and (ii) investigating the accuracy-robustness trade-off in the discriminative learning of the target model. Our findings suggest that extensive data augmentation leads to a remarkable performance boost for the deep-feature-based model while often harming its shallow counterpart. Furthermore, we find that the deep model should be trained for robustness, while the shallow model should emphasize accurate target localization. These results indicate that the deep and shallow models should be trained *independently* and fused at a later stage. As our second contribution, we propose a novel fusion strategy to combine the deep and shallow predictions in order to exploit their complementary characteristics. This is obtained by introducing a quality measure for the predicted state, taking both accuracy and robustness into account.

Experiments are performed on five challenging benchmarks: Need for Speed, VOT2017, Temple128, UAV123 and OTB-2015. Our results clearly demonstrate that the proposed approach provides a significant improvement over the baseline tracker. Further, our approach sets a new state-of-the-art on all four tracking datasets. On the VOT2017 benchmark, our approach achieves an EAO score of 0.378, surpassing the competition winners (0.323) with a relative gain of 17%.

2 Related Work

Deep learning has pervaded many areas of computer vision. While these techniques have also been investigated for visual tracking, it has been with limited success. The SINT method [30] learns a similarity measure offline on a video dataset, and localizes the target using the initial labeled sample. Another approach is to directly regress the relative target location given an input patch [14, 33]. Li et al. [18] tackle the tracking problem in an end-to-end fashion by training a classifier online. The FCNT [34] employs both pre-trained deep features and an online trained model. MDNet [26] further pre-trains a model offline using a multi-domain procedure. Following the end-to-end philosophy, recent works [31, 28] have investigated integrating Discriminative Correlation Filters (DCF) [3, 10] as a computational block in a deep network. The work of [31] integrate DCF into the Siamese framework [2]. Further, [28] employs DCF as a one-layer CNN for end-to-end training.

Other DCF methods focus on integrating convolutional features from a fixed pre-trained deep network [6, 7, 9, 11, 23, 27, 38]. Ma et al. [23] propose a hierarchical ensemble method of independent DCF trackers to combine multiple convolutional layers. Qi et al. [27] learn a correlation filter per feature map, and combine the individual predictions with a modified Hedge algorithm. The MCPF tracker proposed by Zhang et al. [38] combines the deep DCF with a particle filter. Danelljan et al. [11] propose the continuous convolution operator tracker (C-COT) to efficiently integrate multi-resolution shallow and deep feature maps. The subsequent ECO tracker [7] improves the C-COT tracker in terms of performance and efficiency. In this work we adopt the ECO tracking framework due to its versatility and popularity: in the most recent edition of VOT2017 [16], five of the top 10 trackers were based on either ECO or its predecessor C-COT.

3 Analyzing Deep Features for Tracking

Deep learning has brought remarkable performance improvements in many computer vision areas, such as object classification, detection and semantic segmentation. However, its impact is yet to be *fully* realized in the context of generic visual object tracking. In this section, we analyze the causes behind the below-expected performance of deep trackers and propose strategies to address them.

3.1 Motivation

In our quest to seek a better understanding of deep features for tracking, we investigate their properties in relation to the well studied shallow features. One of the well known issues of deep learning is the need for large amounts of labeled training data. Still, thousands of training samples are required to *fine-tune* a pre-trained deep network for a new task. Such amount of data is however not available in the visual tracking scenario, where initially only a single labeled frame is provided. This poses a major challenge when learning deep-feature-based models for visual tracking.

To maximally exploit the available training data, deep learning methods generally employ data augmentation strategies. Yet, data augmentation is seldom used in visual tracking. In fact, the pioneering work of Bolme et al. [3] utilized augmented gray-scale image samples to train a discriminative tracking model. Since then, state-of-the-art deep DCF tracking methods have ignored data augmentation as a strategy for acquiring additional training data. In section 3.3 we therefore perform a thorough investigation of data augmentation techniques with the aim of better understanding deep features for tracking.

Another challenge when integrating deep features is their low spatial resolution, hampering accurate localization of the target. Object trackers based on low-level handcrafted features are primarily trained for accurate target localization to avoid long-term drift. However, this might not be the optimal strategy for deep features which exhibit fundamentally different properties. Deep features generally capture high-level semantics while being invariant to, e.g., small translations and scale changes. From this perspective it may be beneficial to train the deep model to emphasize robustness rather than accuracy. This motivates us to analyze the accuracy/robustness trade-off involved in the model learning, to gain more knowledge about the properties of deep and shallow features. This analysis is performed in section 3.4.

3.2 Methodology

To obtain a clearer understanding of deep and shallow features, we aim to isolate their impact on the overall tracking performance. The analysis is therefore performed with a baseline tracker that exclusively employs either shallow or deep features. This exclusive treatment allows us to directly measure the impact of, e.g., data augmentation on both shallow and deep features separately.

We use the recently introduced ECO tracker [7] as a baseline, due to its state-of-the-art performance. For shallow features, we employ a combination of Histogram of Oriented Gradients (HOG) [5] and Color Names (CN) [35], as it has been used in numerous tracking approaches [7, 10, 19, 15, 21]. For the deep representation, we first restrict our analysis to ResNet-50, using the activations from the fourth convolutional block. Generalization to other networks is further presented in section 5.4. The entire analysis is performed on the OTB-2015 [36] dataset.

3.3 Data Augmentation

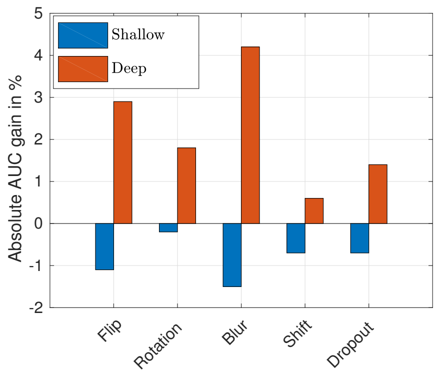
Data augmentation is a standard strategy to alleviate problems with limited training data. It can lead to a better generalization of the learned model for unseen data. However, data augmentation can also lead to lower accuracy in the context of visual tracking due to increased invariance of the model to small translations or scale changes. Therefore, it is unclear whether data augmentation is helpful in the case of tracking.

We separately investigate the impact of different data augmentation techniques on both shallow as well as deep features. We consider the following data augmentation techniques:
Flip: The sample is horizontally flipped.
Rotation: Rotation from a fixed set of 12 angles ranging from $-60°$ to $60°$.
Shift: Shift of n pixels both horizontally and vertically prior to feature extraction. The resulting feature map is shifted back n/s pixels where s is the stride of the feature extraction.
Blur: Blur with a Gaussian filter. This is expected to simulate motion blur and scale variations, which are both commonly encountered in tracking scenarios.
Dropout: Channel-wise dropout of the sample. This is performed by randomly setting 20% of the feature channels to zero. As usual, the remaining feature channels are amplified in order to preserve the sample energy.

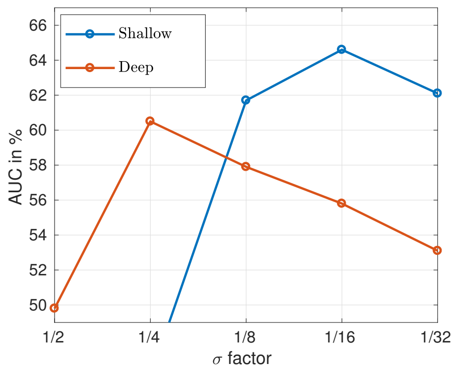
Figure 2a shows the impact of data augmentation on tracking performance (in AUC score [36]). It can be seen that the deep features consistently benefit from data augmentation. All augmentations, except for 'shift', give over 1% improvement in tracking performance. The maximum improvement is obtained using 'blur' augmentation, where a gain of 4% is obtained over the baseline, which employs no data augmentation. Meanwhile, shallow features do not benefit from data augmentation. This surprising difference in behavior of deep and shallow features can be explained by their opposing properties. Deep features capture higher level semantic information that is invariant to the applied augmentations like 'flip', and can thus gain from the increased training data. On the other hand, the shallow features capture low-level information that is hampered by augmentations like 'flip' or 'blur'. The use of data augmentation thus harms the training in this case.

3.4 Robustness/Accuracy Trade-off

When comparing the performance of trackers, there are two important criteria: accuracy and robustness. The former is the measure of *how accurately* the target is localized during tracking. Robustness, on the other hand, is the tracker's resilience to failures in challenging scenarios and its ability to recover. In other words, robustness is a measure of *how often* the target is successfully localized. Generally, both accuracy and robustness are of importance, and a satisfactory



(a) Impact of data augmentation



(b) Impact of label score function width

Fig. 2: Impact of data augmentation (a) and label score function width (b) on shallow (blue) and deep (red) features on OTB-2015. The results are reported as area-under-the-curve (AUC). While deep features significantly benefit from data augmentation, the results deteriorate for shallow features. Similarly, a sharp label function is beneficial for the shallow features, whereas the deep features benefit from a wide label function.

trade-off between these properties is sought since they are weakly correlated [17]. This trade-off can be controlled in the construction and training of the tracker.

In a discriminative tracking framework, the appearance model can be learned to emphasize the accuracy criterion by only extracting positive samples very close to the target location. That is, only very accurate locations are treated as positive samples of the target appearance. Instead, increasing the region from which target samples are extracted allows for more positive training data. This has the potential of promoting the generalization and robustness of the model, but can also result in poor discriminative power when the variations in the target samples become too large.

We analyze the effect of training the tracking model for various degrees of accuracy-robustness trade-off when using either shallow or deep features. In DCF-based trackers, such as the baseline ECO, the size of the region from which positive training samples are extracted is controlled by the width of the label score function. ECO employs a Gaussian function for this task, with standard deviation proportional to the target size with a factor of σ. We analyze different values of σ for both shallow and deep features. Figure 2b shows the results of this experiment. We observe that the deep features are utilized best when trained with higher value of σ, with $\sigma = \frac{1}{4}$ giving the best results. This behavior can be attributed to the invariance property of the deep features. Since they are invariant to small translations, training deep features to get higher accuracy might lead to a suboptimal model. The shallow features on the other hand, perform best when trained with a low σ and give inferior results when trained with a higher σ. This is due to the fact that the shallow features capture low level, higher resolution features, and hence are well suited to give high accuracy. Furthermore, due to their large variance to small transformations, the model is

Table 1: Impact of data augmentation (denoted Aug) and wider label score function (denoted σ) on deep features. Results are shown in terms of AUC score on the OTB-2015 dataset. Both data augmentation and wider label scores provide significant improvement. The best result is obtained when employing both techniques.

	ResNet	ResNet+Aug	ResNet+σ	ResNet+Aug+σ
AUC	56.2	61.5	60.5	**62.0**

unable to handle the larger number of positive training samples implied by a higher σ, resulting in poor performance.

3.5 Observations

The aforementioned results from section 3.3 and section 3.4 show that the deep model significantly improves by the use of data augmentation *and* by training for increased robustness instead of accuracy. We further evaluate the combined effects of data augmentation and higher σ on the deep model. Table 1 shows the results on the OTB-2015 dataset in terms of the AUC measure. The baseline tracker (left) does not employ data augmentation and uses the default value $\sigma = \frac{1}{12}$. Combining all the data augmentation techniques evaluated in section 3.3 provides an improvement of 5.3% in AUC over the baseline. Training with a σ-parameter of $\frac{1}{4}$ further improves the results by 0.5%. Thus our analysis indicates the benefit of using both data augmentation as well as a wider label function when training the deep-feature-based model.

Results from section 3.3 and section 3.4 thus highlight the complementary properties of deep and shallow features. Their corresponding models need to be trained differently, in terms of data and annotations, in order to best exploit their true potential. We therefore argue that the shallow and deep models should be trained independently. However, this raises the question of how to fuse these models in order to leverage their complementary properties, which we address in the next section.

4 Adaptive Fusion of Model Predictions

As previously discussed, the deep and shallow models possess different characteristics regarding accuracy and robustness. This is demonstrated in figure 3, showing detection scores from the shallow and deep models for an example frame. We propose a novel adaptive fusion approach that aims at fully exploiting their complementary nature, based on a quality measure described in section 4.1. In section 4.2 we show how to infer the deep and shallow weights and obtain the final target prediction.

4.1 Prediction Quality Measure

Our aim is to find a quality measure for the target prediction, given the detection score y over the search region of the image. We see the score y as a function over

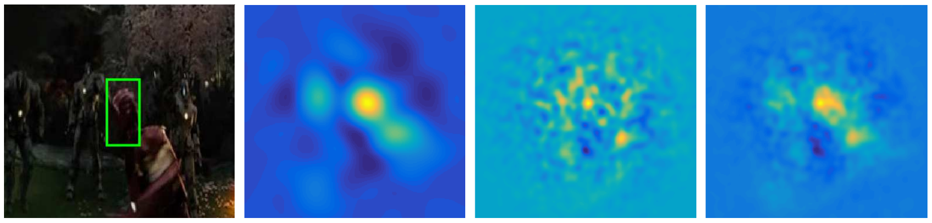


Fig. 3: Visualization of the detection scores produced by the deep and shallow models for a sample frame (a). The deep score (b) contains a robust mode with high confidence, but it only allows for coarse localization. Meanwhile, the shallow score (c) has sharp peaks enabling accurate localization, but also contains distractor modes. Our approach fuses these scores by adaptively finding the optimal weights for each model, producing a sharp and unambiguous score function (d).

the image coordinates, where $y(t) \in \mathbb{R}$ is the target prediction score at location $t \in \mathbb{R}^2$. We require that the quality measure rewards both the accuracy and robustness of the target prediction. The former is related to the *sharpness* of the detection score around the prediction. A sharper peak indicates more accurate localization capability. The robustness of the prediction is derived from the margin to distractor peaks. If the margin is small, the prediction is ambiguous. On the other hand, a large margin indicates that the confidence of the prediction is significantly higher than at other candidate locations.

We propose the minimal weighted confidence margin as a quality measure of a candidate target prediction t^*,

$$\xi_{t^*}\{y\} = \min_t \frac{y(t^*) - y(t)}{\Delta(t - t^*)} \quad . \tag{1}$$

The confidence margin in the numerator is computed as the difference between the confidence score $y(t^*)$ at the candidate prediction t^* and the score $y(t)$ at a location t. The margin is weighted by the distance between t^* and the location t, computed by a distance measure $\Delta : \mathbb{R}^2 \to [0,1]$ satisfying $\Delta(0) = 0$ and $\lim_{|\tau|\to\infty} \Delta(\tau) = 1$. We also assume Δ to be twice continuously differentiable and have a positive definite Hessian at $\tau = 0$. For our purpose, we use

$$\Delta(\tau) = 1 - e^{-\frac{\kappa}{2}|\tau|^2} \quad . \tag{2}$$

Here, κ is a parameter controlling the rate of transition $\Delta(\tau) \to 1$ when $|\tau|$ is increasing. As we will see, κ has a direct interpretation related to the behavior of the quality measure (1) close to the target prediction $t \approx t^*$. From the definition (1) it follows that $\xi_{t^*}\{y\} \geq 0$ if and only if $y(t^*)$ is a global maximum of y.

To verify that the proposed quality measure (1) has the desired properties of promoting both accuracy and robustness, we analyze the cases (a) where t is far from the prediction t^* and (b) when $t \to t^*$. In the former case, we obtain

$|t - t^*| \gg 0$ implying that $\Delta(t - t^*) \approx 1$. In this case, we have that

$$\xi_{t^*}\{y\} \leq \frac{y(t^*) - y(t)}{\Delta(t - t^*)} \approx y(t^*) - y(t)\,, \quad \text{whenever } |t - t^*| \gg 0 \ . \tag{3}$$

As a result, the quality measure $\xi_{t^*}\{y\}$ is approximately bounded by the score-difference to the most significant distractor peak $y(t)$ outside the immediate neighborhood of the prediction t^*. Hence, a large quality measure $\xi_{t^*}\{y\}$ ensures that there are no distractors making the prediction ambiguous. Conversely, if there exists a secondary detection peak $y(t)$ with a similar score $y(t) \approx y(t^*)$, then the quality of the prediction is low $\xi_{t^*}\{y\} \approx 0$.

In the other case we study how the measure (1) promotes an accurate prediction by analyzing the limit $t \to t^*$. We assume that the detection score function y is defined over a continuous domain $\Omega \subset \mathbb{R}^2$ and is twice continuously differentiable. This assumption is still valid for discrete scores y by applying suitable interpolation. The ECO framework in fact outputs scores with a direct continuous interpretation, parametrized by its Fourier coefficients. In any case, we assume the prediction t^* to be a local maximum of y. We denote the gradient and Hessian of y at t as $\nabla y(t)$ and $\mathbf{H}y(t)$ respectively. Since t^* is a local maximum we conclude $\nabla y(t^*) = 0$ and $0 \geq \lambda_1^* \geq \lambda_2^*$, where λ_1^*, λ_2^* are the eigenvalues of $\mathbf{H}y(t^*)$. Using (2), we obtain the result[4]

$$\xi_{t^*}\{y\} \leq \frac{|\lambda_1^*|}{\kappa}\,. \tag{4}$$

Note that the eigenvalue $|\lambda_1^*|$ represents the minimum curvature of the score function y at the peak t^*. Thus, $|\lambda_1^*|$ is a measure of the *sharpness* of the peak t^*. The quality bound (4) is proportional to the sharpness $|\lambda_1^*|$. A high quality value $\xi_{t^*}\{y\}$ therefore ensures that the peak is distinctive, while a flat peak will result on a low quality value. The parameter κ controls the trade-off between the promotion of robustness and accuracy of the prediction. From (4) it follows that κ represents the sharpness $|\lambda_1^*|$ that yields a quality of at most $\xi_{t^*}\{y\} = 1$.

Our approach can be generalized to scale transformations and other higher-dimensional state spaces by extending t to the entire state vector. In this paper, we employ 2-dimensional translation and 1-dimensional scale transformations. In the next section, we show that (1) can be used for jointly finding the prediction t^* and the optimal importance weights for the shallow and deep scores.

4.2 Target Prediction

We present a fusion approach based on the quality measure (1), that combines the deep and shallow model predictions to find the optimal state. Let y_d and y_s denote the scores based on deep and shallow features respectively. The fused score is obtained as a weighted combination of the two scores

$$y_\beta(t) = \beta_\mathrm{d} y_\mathrm{d}(t) + \beta_\mathrm{s} y_\mathrm{s}(t) \ , \tag{5}$$

[4] See the supplementary material for a derivation.

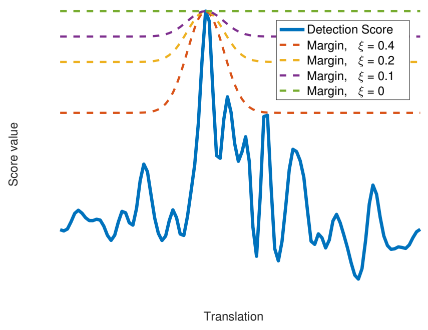


Fig. 4: An illustration of our fusion approach, based on solving the optimization problem (7). A one-dimensional detection score $y_\beta(t)$ (blue curve) is plotted for a particular choice of model weights β, with the candidate state t^* corresponding to the global maximum. The left-hand side of (7c) (dashed curves) is plotted for different values of the slack variable ξ, representing the margin. We find the maximum value of ξ satisfying the inequality (7c), which in this case is $\xi = 0.4$.

where $\beta = (\beta_\mathrm{d}, \beta_\mathrm{s})$ are the weights for the deep and shallow scores, respectively. Our aim is to jointly estimate the score weights β and the target state t^* that maximize the quality measure (1). This is achieved by minimizing the loss

$$\text{minimize:} \quad L_{t^*}(\beta) = -\xi_{t^*}\{y_\beta\} + \mu\left(\beta_\mathrm{d}^2 + \beta_\mathrm{s}^2\right) \tag{6a}$$
$$\text{subject to:} \quad \beta_\mathrm{d} + \beta_\mathrm{s} = 1\,, \quad \beta_\mathrm{d} \geq 0\,, \quad \beta_\mathrm{s} \geq 0\ . \tag{6b}$$

Note that we have added a regularization term, controlled by the parameter μ, penalizing large deviations in the weights. The score weights themselves are constrained to be non-negative and sum up to one.

To optimize (6), we introduce a slack variable $\xi = \xi_{t^*}\{y_\beta\}$, resulting in the equivalent minimization problem

$$\text{minimize:} \quad L_{t^*}(\xi, \beta) = -\xi + \mu\left(\beta_\mathrm{d}^2 + \beta_\mathrm{s}^2\right) \tag{7a}$$
$$\text{subject to:} \quad \beta_\mathrm{d} + \beta_\mathrm{s} = 1\,, \quad \beta_\mathrm{d} \geq 0\,, \quad \beta_\mathrm{s} \geq 0 \tag{7b}$$
$$y_\beta(t^*) - \xi\varDelta(t^* - t) \geq y_\beta(t)\,, \quad \forall t \in \varOmega\ . \tag{7c}$$

A visualization of this reformulated problem and the constraint (7c) is shown in figure 4. For any fixed state t^*, (7) corresponds to a Quadratic Programming (QP) problem, which can be solved using standard techniques. In practice, we sample a finite set of candidate states $\varOmega$ based on local maxima from the deep and shallow scores. Subsequently, (7) is optimized for each state $t^* \in \varOmega$ by solving a three-parameter QP problem, adding minimal computational overhead. We then select the candidate state t^* with lowest overall loss (7a) as our final prediction.

5 Experiments

5.1 Implementation Details

Our tracker is implemented in Matlab using MatConvNet [32]. Based on the analysis in section 3.4, we select $\sigma_\mathrm{d} = 1/4$ and $\sigma_\mathrm{s} = 1/16$ for deep and shallow label functions respectively, when training the models. As concluded in section 3.3, we employ the proposed data augmentation techniques only for deep features. For the fusion method presented in section 4, the regularization parameter μ in (6) is set to 0.15. We set the κ parameter in the distance measure (2) to be inversely proportional to the target size with a factor of 8. All parameters were set using a *separate* validation set, described in the next section. We then use the same set of parameters for all datasets, throughout all experiments.

5.2 Evaluation Methodology

We evaluate our method on four challenging benchmarks: the recently introduced Need For Speed (NFS) [12], VOT2017 [16], UAV123 [24], and Temple128 [20]. NFS consists of 100 high frame rate (240 fps) videos as well as their 30 fps versions. We use the 30 fps version of the dataset for our experiments. Mean overlap precision (OP) and area-under-the-curve (AUC) scores are used as evaluation measures. The OP score is computed as the percentage of frames in a video where the intersection-over-union (IOU) overlap with the ground-truth exceeds a certain threshold. The mean OP over all videos is plotted over the range of IOU thresholds $[0, 1]$ to get the success plot. The area under this plot gives the AUC score. We refer to [36] for details. Due to the stochastic nature of the dropout augmentation, we run our tracker 10 times on each sequence and report average scores to robustly estimate the performance on *all* datasets. Details about VOT2017, UAV123 and Temple128 are provided in section 5.5.
Validation set: We use a subset of the popular OTB-2015 dataset [36] as our validation set for tuning all hyperparameters. The OTB-2015 dataset has been commonly used for evaluation by the tracking community. However, the dataset has saturated in recent years with several trackers [7, 26] achieving over 90% OP score at threshold 0.5 due to the majority of relatively easy videos. Instead, we are primarily interested in advancing tracking performance in the challenging and unsolved cases, where deep features are of importance. We therefore construct a subset of hard videos from OTB-2015 to form our validation set, termed OTB-H. To find the *hardest* videos in OTB-2015, we consider the per-video results of four deep-feature-based trackers with top overall performance on the dataset: ECO [7], C-COT [11], MDNet [26], and TCNN [25]. We first select sequences for which the average IOU is less than 0.6 for at least two of the four trackers. We further remove sequences overlapping with the VOT2017 dataset. The resulting OTB-H contains 23 sequences, which we use as the validation set when setting all the parameters. The remaining 73 easier videos form the OTB-E dataset that we use in our ablative studies as a test set along with NFS dataset.

5.3 Ablative Study

We first investigate the impact of the observations from section 3 in a tracking framework employing both deep and shallow features. To independently evaluate our contributions, we fuse the model predictions as in (5) with fixed weights β. By varying these weights, we can further analyze the contribution of the deep

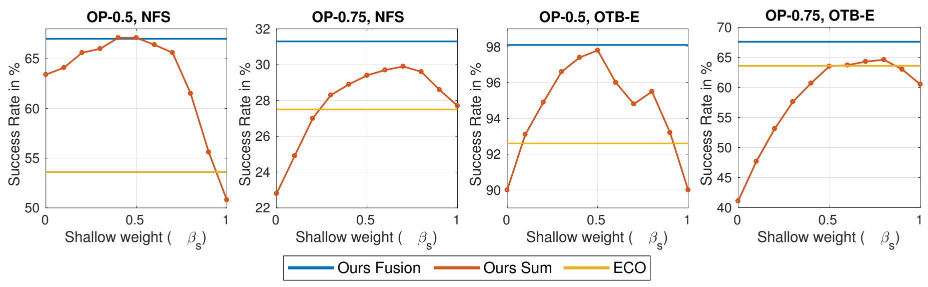


Fig. 5: Analysis of tracking robustness and accuracy using the OP scores at IOU thresholds of 0.5 and 0.75 respectively on the NFS and OTB-E datasets. We plot the performance of our approach using sum-fusion with fixed weights (red) for a range of different shallow weights β_s. These results are also compared with the baseline ECO (orange) and our adaptive fusion (blue). For a wide range of β_s values, our sum-fusion approach outperforms the baseline ECO in robustness on both datasets. Our adaptive fusion achieves the best performance both in terms of accuracy *and* robustness.

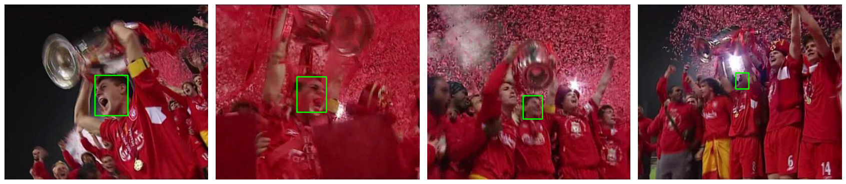
and shallow models to the final tracking accuracy and robustness. We employ the widely used PASCAL criterion as an indicator of robustness. It measures the percentage of successfully tracked frames using an IOU threshold of 0.5, equivalent to OP at 0.5. Furthermore, we consider a localization to be *accurate* if its IOU is higher than 0.75, since this is the upper half $[0.75, 1]$ of the IOU range $[0.5, 1]$ representing successfully tracked frames.

Figure 5 plots the accuracy and robustness indicators, as described above, on NFS and OTB-E for different values of the shallow model weight β_s. In all cases, the deep weight is set to $\beta_d = 1 - \beta_s$. We also show the performance of the baseline ECO, using the *same* set of deep and shallow features. We observe that our tracker with a fixed sum-fusion outperforms the baseline ECO for a wide range of weights β_s. This demonstrates the importance of employing specifically tailored training procedures for deep and shallow features, as observed in section 3.5.

Despite the above improvements obtained by our analysis of deep and shallow features, we note that optimal robustness and accuracy are mutually exclusive, and cannot be obtained even by careful selection of the weight parameter β_s. While shallow features (large β_s) are beneficial for accuracy, deep features (small β_s) are crucial for robustness. Figure 5 also shows the results of our proposed adaptive fusion approach (section 4), where the model weights β are dynamically computed in each frame. Compared to using a sum-fusion with fixed weights, our adaptive approach achieves improved accuracy *without* sacrificing robustness. Figure 6 shows a qualitative example of our adaptive fusion approach.

5.4 Generalization to Other Networks

With the advent of deep learning, numerous network architectures have been proposed in recent years. Here, we investigate the generalization capabilities of our findings across different deep networks. Table 2 shows the performance of the proposed method and baseline ECO on three popular architectures: VGG-M [4], GoogLeNet [29], and ResNet-50 [13]. The results are reported in terms of AUC



(a) $\beta_d = 0.01$ $\beta_s = 0.99$ **(b)** $\beta_d = 0.90$ $\beta_s = 0.10$ **(c)** $\beta_d = 0.10$ $\beta_s = 0.90$ **(d)** $\beta_d = 0.87$ $\beta_s = 0.13$

Fig. 6: Qualitative example of our fusion approach. The adaptively computed model weights β_d, β_s are shown for four frames from the *Soccer* sequence. The shallow model is prominent early in the sequence (a), before any significant appearance change. Later, when encountered with occlusions, clutter and out-of-plane rotations (b,d), our fusion emphasizes the deep model due to its superior robustness. In (c), where the target undergoes scale changes, our fusion exploits the shallow model for better accuracy.

Table 2: Generalization of our tracker across different network architectures. Results are shown in terms of AUC scores on the NFS and OTB-E dataset. The baseline ECO fails to exploit the power of more sophisticated architectures. Instead, our approach provides consistent gains over ECO when moving towards more advanced networks.

	VGG-M		**GoogLeNet**		**ResNet-50**	
	OTB-E	NFS	OTB-E	NFS	OTB-E	NFS
ECO	74.8	45.3	74.4	45.4	74.3	45.7
Ours	74.2	49.7	76.0	51.6	78.0	54.1

scores on NFS and OTB-E datasets. ECO fails to exploit more sophisticated deeper architectures: GoogLeNet and ResNet. In case of ResNet, our approach achieves a significant gain of 3.7% and 8.4% on OTB-E and NFS datasets respectively. These results show that our analysis in section 3 and the fusion approach proposed in section 4 generalizes across different network architectures.

5.5 State-of-the-Art

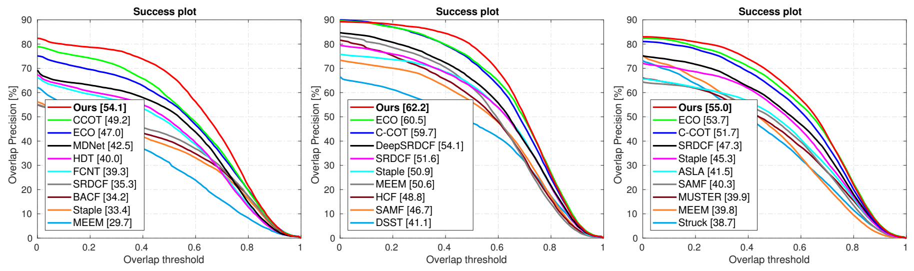
Here, we compare our tracker with state-of-the-art methods on four challenging tracking datasets. Further details are provided in the supplementary material.

VOT2017 Dataset [16]: On VOT2017, containing 60 videos, tracking performance is evaluated both in terms of accuracy (average overlap during successful tracking) and robustness (failure rate). The Expected Average Overlap (EAO) measure, which merges both accuracy and robustness, is then used to obtain the overall ranking. The evaluation metrics are computed as an average over 15 runs (see [16] for further details). The results in table 3 are presented in terms of EAO, robustness, and accuracy. Our approach significantly outperforms the top ranked method LSART with a relative gain of 17%, achieving an EAO score of 0.378. In terms of robustness, our approach obtains a relative gain of 17% compared to LSART. Furthermore, we achieve the best results in terms of accuracy, demonstrating the overall effectiveness of our approach.

Need For Speed Dataset [12]: Figure 7a shows the success plot over all the 100 videos. The AUC scores are reported in the legend. Among previous methods,

Table 3: Comparison with the state-of-the-art in terms of expected average overlap (EAO), robustness (failure rate), and accuracy on the VOT2017 benchmark. We compare with the top-10 trackers in the competition. Our tracker obtains a significant relative gain of 17% in EAO, compared to the top-ranked method (LSART).

	MCPF	SiamDCF	CSRDCF	CCOT	MCCT	Gnet	ECO	CFCF	CFWCR	LSART	**Ours**
EAO	0.248	0.249	0.256	0.267	0.270	0.274	0.280	0.286	0.303	0.323	**0.378**
Robustness	0.427	0.473	0.356	0.318	0.323	0.276	0.276	0.281	0.267	0.218	**0.182**
Accuracy	0.510	0.500	0.491	0.494	0.525	0.502	0.483	0.509	0.484	0.493	**0.532**



(a) NFS **(b)** Temple128 **(c)** UAV123

Fig. 7: Success plots on the NFS (a), Temple128 (b), and UAV123 (c) datasets. Our tracker significantly outperforms the state-of-the-art on all datasets.

CCOT [11] and ECO [7] achieve AUC scores of 49.2% and 47.0% respectively. Our approach significantly outperforms CCOT with a relative gain of 10%.
Temple128 Dataset [20]: Figure 7b shows the success plot over all 128 videos. Among the existing methods, ECO achieves an AUC score of 60.5%. Our approach outperforms ECO with an AUC score of 62.2%.
UAV123 Dataset [24]: This dataset consists of 123 aerial tracking videos captured from a UAV platform. Figure 7c shows the success plot. Among the existing methods, ECO achieves an AUC score of 53.7%. Our approach outperforms ECO by setting a new state-of-the-art, with an AUC of 55.0%.

6 Conclusions

We perform a systematic analysis to identify key causes behind the below-expected performance of deep features for visual tracking. Our analysis shows that individually tailoring the training for shallow and deep features is crucial to obtain both high robustness and accuracy. We further propose a novel fusion strategy to combine the deep and shallow appearance models leveraging their complementary characteristics. Experiments are performed on four challenging datasets. Our experimental results clearly demonstrate the effectiveness of the proposed approach, leading to state-of-the-art performance on all datasets.

Acknowledgments: This work was supported by Swedish Foundation for Strategic Research (SymbiCloud), Swedish Research Council (EMC^2, starting grant 2016-05543), CENIIT grant (18.14), Swedish National Infrastructure for Computing, and Wallenberg AI, Autonomous Systems and Software Program.

References

1. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: CVPR (2016)
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV workshop (2016)
3. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR (2010)
4. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
6. Danelljan, M., Bhat, G., Gladh, S., Khan, F.S., Felsberg, M.: Deep motion and appearance cues for visual tracking. Pattern Recognition Letters (2018). https://doi.org/10.1016/j.patrec.2018.03.009
7. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: efficient convolution operators for tracking. In: CVPR (2017)
8. Danelljan, M., Hager, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence (2016)
9. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: ICCV Workshop (2015)
10. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV (2015)
11. Danelljan, M., Robinson, A., Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV (2016)
12. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1134–1143. IEEE (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: ECCV (2016)
15. Hong, Z., Chen, Z., Wang, C., Mei, X., Prokhorov, D., Tao, D.: Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 749–758 (2015)
16. Kristan, M., Leonardis, A., Matas, J., Felsberg, Pflugfelder, R., M., Čehovin, L., Vojír, T.and Hger, G., et al. The visual object tracking vot2017 challenge results. In: ICCV workshop (2017)
17. Kristan, M., Matas, J., Leonardis, A., Vojír, T., Pflugfelder, R.P., Fernández, G., Nebehay, G., Porikli, F., Cehovin, L.: A novel performance evaluation methodology for single-target trackers. TPAMI **38**(11), 2137–2155 (2016)
18. Li, H., Li, Y., Porikli, F.: Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In: BMVC (2014)
19. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: European Conference on Computer Vision. pp. 254–265. Springer (2014)

20. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: Algorithms and benchmark. TIP **24**(12), 5630–5644 (2015)
21. Lukežič, A., Vojíř, T., Čehovin, L., Matas, J., Kristan, M.: Discriminative correlation filter with channel and spatial reliability. arXiv preprint arXiv:1611.08461 (2016)
22. Lukežič, A., Vojíř, T., Zajc, L.Č., Matas, J., Kristan, M.: Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision pp. 1–18
23. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV (2015)
24. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: ECCV (2016)
25. Nam, H., Baek, M., Han, B.: Modeling and propagating cnns in a tree structure for visual tracking. CoRR **abs/1608.07242** (2016)
26. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR (2016)
27. Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.H.: Hedged deep tracking. In: CVPR (2016)
28. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R., Yang, M.H.: CREST: Convolutional residual learning for visual tracking. In: ICCV (2017)
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
30. Tao, R., Gavves, E., Smeulders, A.W.M.: Siamese instance search for tracking. In: CVPR (2016)
31. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: CVPR (2017)
32. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab. CoRR **abs/1412.4564** (2014)
33. Wang, C., Galoogahi, H.K., Lin, C., Lucey, S.: Deep-lk for efficient adaptive object tracking. In: ICRA (2018)
34. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: ICCV (2015)
35. van de Weijer, J., Schmid, C., Verbeek, J.J., Larlus, D.: Learning color names for real-world applications. TIP **18**(7), 1512–1524 (2009)
36. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. TPAMI **37**(9), 1834–1848 (2015)
37. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: ECCV (2014)
38. Zhang, T., Xu, C., Yang, M.H.: Multi-task correlation particle filter for robust object tracking. In: CVPR (2017)