ATOM: Accurate Tracking by Overlap Maximization

Martin Danelljan*,[1,2] Goutam Bhat*,[1,2] Fahad Shahbaz Khan[1,3] Michael Felsberg[1]

[1] CVL, Linköping University, Sweden [2] CVL, ETH Zürich, Switzerland [3] Inception Institute of Artificial Intelligence, UAE

Abstract

While recent years have witnessed astonishing improvements in visual tracking robustness, the advancements in tracking accuracy have been limited. As the focus has been directed towards the development of powerful classifiers, the problem of accurate target state estimation has been largely overlooked. In fact, most trackers resort to a simple multi-scale search in order to estimate the target bounding box. We argue that this approach is fundamentally limited since target estimation is a complex task, requiring high-level knowledge about the object.

*We address this problem by proposing a novel tracking architecture, consisting of dedicated target estimation and classification components. High level knowledge is incorporated into the target estimation through extensive offline learning. Our target estimation component is trained to predict the overlap between the target object and an estimated bounding box. By carefully integrating target-specific information, our approach achieves previously unseen bounding box accuracy. We further introduce a classification component that is trained online to guarantee high discriminative power in the presence of distractors. Our final tracking framework sets a new state-of-the-art on five challenging benchmarks. On the new large-scale TrackingNet dataset, our tracker **ATOM** achieves a relative gain of* 15% *over the previous best approach, while running at over* 30 *FPS. Code and models are available at* https://github.com/visionml/pytracking.

1. Introduction

Generic online visual tracking is a hard and ill-posed problem. The tracking method must learn an appearance model of the target online based on minimal supervision, often a single starting frame in the video. The model then needs to generalize to unseen aspects of the target appearance, including different poses, viewpoints, lightning conditions etc. The tracking problem can be decomposed into a *classification* task and an *estimation* task. In the former case, the aim is to robustly provide a coarse location of the target in the image by categorizing image regions into foreground and background. The second task is then to estimate the target state, often represented by a bounding box.

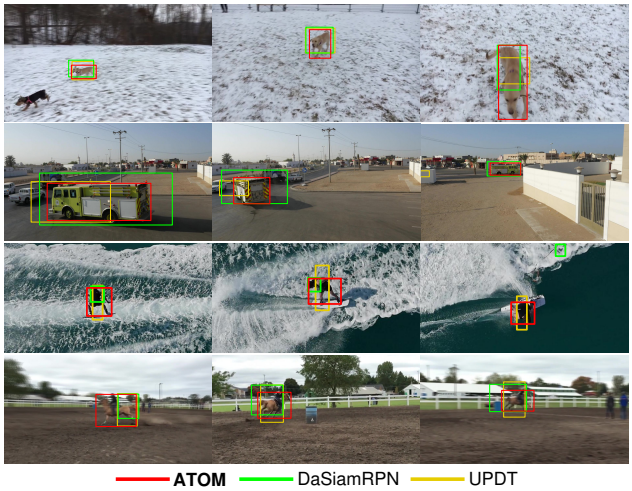


Figure 1. A comparison of our approach with state-of-the-art trackers. UPDT [3], based on correlation filters, lacks an explicit target state estimation component, performing a brute-force multi-scale search instead. Consequently, it does not handle aspect-ratio changes, which can lead to tracking failure (second row). DaSiamRPN [39] employs a bounding box regression strategy to estimate the target state, but still struggles in cases of out-of-plane rotation, deformation, etc. Our approach ATOM, employing an overlap prediction network, successfully handles these challenges and provides accurate bounding box predictions.

In recent years, the focus of tracking research has been on target classification. Much attention has been invested into constructing robust classifiers, based on e.g. correlation filters [6, 22, 31], and exploiting powerful deep feature representations [3, 34] for this task. On the other hand, target estimation has seen below expected progress. This trend is clearly observed in the recent VOT2018 challenge [17], where older trackers such as KCF [13] and MEEM [37] still obtain competitive accuracy while exhibiting vastly inferior robustness. In fact, most current state-of-the-art trackers [3, 4, 31] still rely on the classification component for target estimation by performing a multi-scale search. How-

*Both authors contributed equally.

ever, this strategy is fundamentally limited since bounding box estimation is inherently a challenging task, requiring high-level understanding of the object's pose (see figure 1).

In this work, we set out to bridge the performance gap between target classification and estimation in visual object tracking. We introduce a novel tracking architecture consisting of two components designed exclusively for target estimation and classification. Inspired by the recently proposed IoU-Net [15], we train the target estimation component to predict the Intersection over Union (IoU) overlap, i.e. the Jaccard Index [14], between the target and an estimated bounding box. Since the original IoU-Net is class-specific, and hence not suitable for generic tracking, we propose a novel architecture for integrating target-specific information into the IoU prediction. We achieve this by introducing a modulation-based network component that incorporates the target appearance in the reference image to obtain target-specific IoU estimates. This further enables our target estimation component to be trained *offline* on large-scale datasets. During tracking, the target bounding box is found by simply maximizing the predicted IoU overlap in each frame.

To develop a seamless and transparent tracking method, we also revisit the problem of target classification with the aim of avoiding unnecessary complexity. Our target classification component is simple yet powerful, consisting of a two-layer fully convolutional network head. Unlike our target estimation module, the classification component is trained *online*, providing high robustness against distractor objects in the scene. To ensure real-time performance, we address the problem of efficient online optimization, where gradient descent falls short. Instead, we employ a Conjugate-Gradient-based strategy and demonstrate how it can be easily implemented in modern deep learning frameworks. Our final tracking loop is simple, alternating between target classification, estimation, and model update.

We perform comprehensive experiments on five challenging benchmarks: NFS [9], UAV123 [24], TrackingNet [25], LaSOT [8], and VOT2018 [17]. Our tracking approach sets a new state-of-the-art on all five datasets, achieving an absolute gain of 10% on the challenging LaSOT dataset. Moreover, we provide an analysis of our tracker, along with different network architectures for overlap prediction.

2. Related Work

In the context of visual tracking, it often makes sense to distinguish between *target classification* and *target estimation* as two separate, but related subtasks. Target classification basically aims at determining the presence of the target object at a certain image location. However, only partial information about the target state is obtained, e.g. its image coordinates. Target estimation then aims at finding the full state. In visual tracking, the target state is often represented by a bounding box, either axis aligned [9, 35] or rotated [17]. State estimation is then reduced to finding the image bounding box that best describes the target in the current frame. In the simplest case, the target is rigid and only moves parallel to the camera plane. In such a scenario, *target estimation* reduces to finding the 2D image-location of the target, and therefore does not need to be considered separately from *target classification*. In general, however, objects may undergo radical variations in pose and viewpoint, greatly complicating the task of bounding box estimation.

In the last few years, the challenge of target classification has been successfully addressed by discriminatively training powerful classifiers online [6, 13, 26]. In particular, the correlation-based trackers [7, 13, 23] have gained wide popularity. These methods rely on the diagonalizing transformation of circular convolutions, given by the Discrete Fourier Transform, to perform efficient fully convolutional training and inference. Correlation filter methods often excel at target classification by computing reliable confidence scores in a dense 2D-grid. On the other hand, accurate target estimation has long eluded such approaches. Even finding a one-parameter scale factor has turned out a formidable challenge [5, 20] and most approaches resort to the brute-force multi-scale detection strategy with its obvious computational impact. As such, the default method is to apply the classifier alone to perform full state estimation. However, target classifiers are not sensitive to all aspects of the target state, e.g. the width and height of the target. In fact, invariance to some aspects of the target state is often considered a valuable property of the discriminative model to improve robustness [2, 3, 26]. Instead of relying on the classifier, we learn a dedicated target estimation component.

Accurate estimation of an object's bounding box is a complex task, requiring high-level a-priori knowledge. The bounding box depends on the pose and viewpoint of the object, which cannot be modeled as a simple image transformation (e.g. uniform image scaling). It is therefore highly challenging, if not impossible, to learn accurate target estimation online from scratch. Many recent methods in the literature have therefore integrated prior knowledge in the form of heavy offline learning [18, 26, 39]. In particular, SiamRPN [18] and its extension [39] have been shown capable of bounding box regression thanks to extensive offline training. Yet, these Siamese tracking approaches often struggle at the problem of target classification. Unlike, for instance, correlation-based methods, most Siamese trackers do not explicitly account for distractors, since no online learning is performed. While this problem has been partly addressed using simple template update techniques [39], it has yet to reach the level of strong online-learned models. In contrast to Siamese methods, we learn the classification model online, while also utilizing extensive offline training for the target estimation task.

3. Proposed Method

In this work, we propose a novel tracking approach consisting of two components: 1) A target estimation module that is learned *offline*; and 2) A target classification module that is learned *online*. That is, following the modern trend in object detection [27, 28], we separate the subproblems of target classification and estimation. Yet, both of these tasks are integrated in a unified multi-task network architecture, shown in figure 2.

We employ the same backbone network for both the target classification and estimation tasks. For simplicity, we use a ResNet-18 model that is trained on ImageNet and refrain from fine-tuning the backbone in this work. Target estimation is performed by the IoU-predictor network. This network is trained offline on large-scale video tracking and object detection datasets, and its weights are frozen during online tracking. The IoU-predictor takes four inputs: i) backbone features from current frame, ii) bounding box estimates in the current frame, iii) backbone features from a reference frame, iv) the target bounding box in the reference frame. It then outputs the predicted Intersection over Union (IoU) score for each of the current-frame bounding box estimates. During tracking, the final bounding box is obtained by maximizing the IoU score using gradient ascent. The target estimation component is detailed in section 3.1.

Target classification is performed by another network head. Unlike the target estimation component, the classification network is entirely learned during online tracking. It is exclusively trained to discriminate the target from other objects in the scene by predicting a target confidence score based on backbone features extracted from the current frame. Both training and prediction are performed in a fully convolutional manner to ensure efficiency and coverage. However, training such a network online with conventional approaches, such as stochastic gradient descent, is suboptimal for our online purpose. We therefore propose to use an optimization strategy, based on Conjugate Gradient and Gauss-Newton, that enables fast online training. Moreover, we demonstrate how this approach can be easily implemented in common deep learning frameworks, such as PyTorch, by exploiting the back-propagation functionality. Our target classification approach is described in section 3.2 and our final tracking framework is detailed in section 3.3.

3.1. Target Estimation by Overlap Maximization

In this section, we detail how the target state estimation is performed. The aim of our state estimation component is to determine the target bounding box given a rough initial estimate. We take inspiration from the IoU-Net [15], which was recently proposed for object detection as an alternative to typical anchor-based bounding box regression techniques. In contrast to conventional approaches, the IoU-Net is trained to predict the IoU between an image object and an input bounding box candidate. Bounding box estimation is then performed by maximizing the IoU prediction.

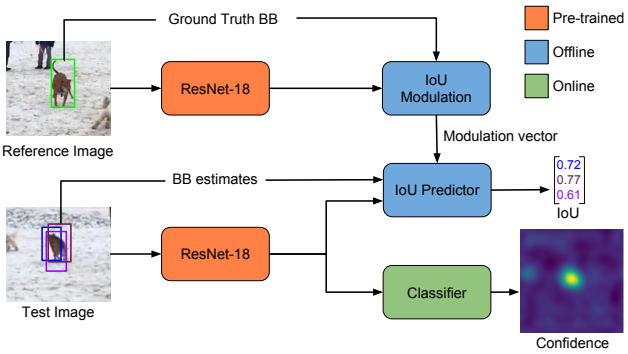


Figure 2. Overview of our network architecture for visual tracking. We augment two modules to the pretrained ResNet-18 backbone network (orange). The target estimation module (blue) is trained offline on large-scale datasets to predict the IoU overlap with the target. Using the reference frame and the initial target box, modulation vectors carrying target-specific appearance information are computed. The IoU predictor component then receives features and proposal bounding boxes in the test frame, along with the aforementioned modulation vectors. It estimates the IoU for each input box. The target classification module (green) is trained online to output target confidences in a fully convolutional manner.

To describe our target estimation component, we first briefly revisit the IoU-Net model. Given a deep feature representation of an image, $x \in \mathbb{R}^{W \times H \times D}$, and a bounding box estimate $B \in \mathbb{R}^4$ of an image object, IoU-Net predicts the IoU between B and the object. Here B is parametrized as $B = (c_x/w, c_y/h, \log w, \log h)$, where (c_x, c_y) are the image coordinates of the bounding box center. The network uses a Precise ROI Pooling (`PrPool`) [15] layer to pool the region in x given by B, resulting in a feature map x_{B} of a pre-determined size. Essentially, `PrPool` is a continuous variant of adaptive average pooling, with the key advantage of being differentiable w.r.t. the bounding box coordinates B. This allows the bounding box B to be refined by maximizing the IoU w.r.t. B through gradient ascent.

Network Architecture: For the task of object detection, independent IoU-Nets are trained in [15] for each object class. However, in tracking the target class is generally unknown. Further, unlike object detection, the target is not required to belong to any set of pre-defined classes or be represented in any existing training datasets. Class-specific IoU predictors are thus of little use for generic visual tracking. Instead, *target-specific* IoU predictions are required, by exploiting the target annotation in the first frame. Due to the high-level nature of the IoU prediction task, it is not feasible to train, or even fine-tune the IoU-Net online on a single frame. Thus, we argue that the target estimation network needs to be trained offline to learn a general representation for IoU prediction.

In the context of visual tracking, where the target object

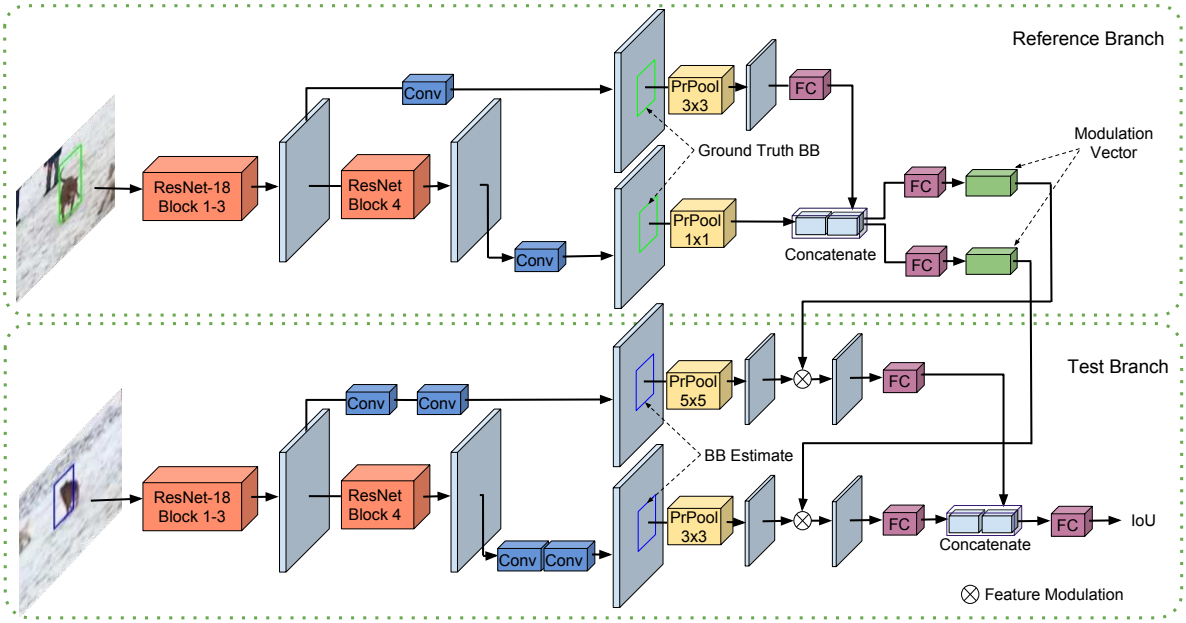


Figure 3. Full architecture of our target estimation network. ResNet-18 `Block3` and `Block4` features extracted from the test image are first passed through two `Conv` layers. Regions defined by the input bounding boxes are then pooled to a fixed size using `PrPool` layers. The pooled features are modulated by channel-wise multiplication with the coefficient vector returned by the reference branch. The features are then passed through fully-connected layers to predict the IoU. All `Conv` and `FC` layers are followed by `BatchNorm` and `ReLU`.

is unknown beforehand, the challenge is thus to construct an IoU prediction architecture that makes effective use of the reference target appearance given at test-time. Our initial experiments showed that naive approaches for fusing the reference image features with the current-frame features yield poor performance (see section 4.1). We also found Siamese architectures to provide suboptimal results. In this work, we therefore propose a *modulation-based* network architecture that predicts the IoU for an arbitrary object given only a single reference image. The proposed network is visualized in figure 3. Our network has two branches, both of which take backbone features from ResNet-18 `Block3` and `Block4` as input. The *reference branch* inputs features x_0 and the target bounding box annotation B_0 in the reference image. It returns a modulation vector $c(x_0, B_0)$, consisting of positive coefficients of size $1 \times 1 \times D_z$. As illustrated in figure 3, this branch consists of a convolutional layer followed by `PrPool` and a fully connected layer.

The current image, in which we want to estimate the target bounding box, is processed through the *test branch*. It first extracts a deep representation by feeding the backbone features x through two convolutional layers followed by a `PrPool` with the bounding box estimate B. As the test branch extracts general features for IoU prediction, which constitutes a more complex task, it employs more layers and higher pooling resolution compared to the reference branch (see figure 3). The resulting representation $z(x, B)$ is of size $K \times K \times D_z$, where K is spatial output size of the `PrPool` layer. The computed feature representation of the test image is then *modulated* by the coefficient vector c via a channel-wise multiplication. This creates a target-specific representation for IoU prediction, effectively incorporating the reference appearance information. The modulated representation is finally fed to the IoU predictor module g, consisting of three fully connected layers. The predicted IoU of the bounding box B is hence given by

$$\text{IoU}(B) = g\big(c(x_0, B_0) \cdot z(x, B)\big) \,. \tag{1}$$

To train the network, we minimize the prediction error of (1), given annotated data. During tracking we maximize (1) w.r.t. B to estimate the target state.

Training: From (1) it is clear that the entire IoU prediction network can be trained in an end-to-end fashion, using bounding-box-annotated image pairs. We use the training splits of the recently introduced Large-scale Single Object Tracking (LaSOT) dataset [8] and TrackingNet [25]. We sample image pairs from the videos with a maximum gap of 50 frames. Similar to [39], we augment our training data with synthetic image pairs from the COCO dataset [21] to have more diverse classes. From the reference image, we sample a square patch centered at the target, with an area of about 5^2 times the target area. From the test image, we sample a similar patch, with some perturbation in the position and scale to simulate the tracking scenario. These cropped regions are then resized to a fixed size. For each image pair we generate 16 candidate bounding boxes by adding Gaussian noise to the ground truth coordinates, while ensuring a minimum IoU of 0.1. We use image flipping and color

jittering for data augmentation. As in [15], the IoU is normalized to $[-1, 1]$.

The weights in our head network are initialized using [12]. For the backbone network, we freeze all weights during training. We use the mean-squared error loss function and train for 40 epochs with 64 image pairs per batch. The ADAM [16] optimizer is employed with initial learning rate of 10^{-3}, and using a factor 0.2 decay every 15 epochs.

3.2. Target Classification by Fast Online Learning

While the target estimation module provides accurate bounding box outputs, it lacks the ability to robustly discriminate between the target object and background distractors. We therefore complement the estimation module with a second network head, whose sole purpose is to perform this discrimination. Unlike the estimation component, the target classification module is exclusively trained online, to predict a target confidence score. Since the goal of the target classification module is to provide a rough 2D-location of the object, we wish it to be *invariant* to the size and scale of the target. Instead, it should emphasize robustness by minimizing false detections.

Model: Our target classification module is a 2-layer fully convolutional neural network, formally defined as

$$f(x; w) = \phi_2(w_2 * \phi_1(w_1 * x)) \,. \tag{2}$$

Here, x is the backbone feature map, $w = \{w_1, w_2\}$ are the parameters of the network, ϕ_1, ϕ_2 are activation functions and $*$ denotes standard multi-channel convolution. While our framework is general, allowing more complex models for this purpose, we found such a simple model sufficient and beneficial in terms of computational efficiency.

Inspired by the recent success of discriminative correlation filter (DCF) approaches, we formulate a similar learning objective based on the L^2 classification error,

$$L(w) = \sum_{j=1}^{m} \gamma_j \|f(x_j; w) - y_j\|^2 + \sum_k \lambda_k \|w_k\|^2 \,. \tag{3}$$

Each training sample feature map x_j is annotated by the classification confidences $y_j \in \mathbb{R}^{W \times H}$, set to a sampled Gaussian function centered at the target location. The impact of each training sample is controlled by the weight γ_j, while the amount of regularization on w_k is set by λ_k.

Online Learning: A brute-force approach to minimize (3) would be to apply standard gradient descent or its stochastic twin. These approaches are easily implemented in modern deep learning libraries, but are not well suited for online learning due to their slow convergence rates. We therefore develop a more sophisticated optimization strategy that is tailored for such online learning problems, yet requiring only little added implementation complexity. First, we define the residuals of the problem as $r_j(w) = \sqrt{\gamma_j}(f(x_j; w) - y_j)$ for $j \in \{1, \dots, m\}$ and $r_{m+k}(w) = \sqrt{\lambda_k} w_k$ for $k = 1, 2$. The loss (3) is then equivalently written as the squared L^2 norm of the residual vector $L(w) = \|r(w)\|^2$, where $r(w)$ is the concatenation of all residuals $r_j(w)$. We utilize the quadratic Gauss-Newton approximation $\tilde{L}_w(\Delta w) \approx L(w + \Delta w)$, obtained from a first order Taylor expansion of the residuals $r(w + \Delta w) \approx r_w + J_w \Delta w$ at the current estimate w,

$$\tilde{L}_w(\Delta w) = \Delta w^{\mathrm{T}} J_w^{\mathrm{T}} J_w \Delta w + 2 \Delta w^{\mathrm{T}} J_w^{\mathrm{T}} r_w + r_w^{\mathrm{T}} r_w \,. \tag{4}$$

Here, we have defined $r_w = r(w)$ and $J_w = \frac{\partial r}{\partial w}$ is the Jacobian of r at w. The new variable Δw represents the increment in the parameters w.

The Gauss-Newton subproblem (4) forms a positive definite quadratic function, allowing the deployment of specialized machinery such as the Conjugate Gradient (CG) method. While a full description of CG is outside the scope of this paper (see [29] for a full treatment), intuitively it finds an optimal search direction p and step length α in each iteration. Since the CG algorithm consists of simple vector operations, it can be implemented with only a few lines of python code. The only challenging aspect of CG is the evaluation of the operation $J_w^{\mathrm{T}} J_w p$ for a search direction p.

We note that CG has been successfully deployed in some DCF tracking approaches [4, 7, 31]. However, these implementations rely on hand-coding all operations in order to implement $J_w^{\mathrm{T}} J_w p$, requiring much tedious work and derivations even for a simple model (2). This approach also lacks flexibility since any minor modification of the architecture (2), such as adding a layer or changing a non-linearity, may require comprehensive re-derivation and implementation work. In this paper, we therefore demonstrate how to implement CG for (4) by exploiting the backpropagation functionality of modern deep learning frameworks, such as PyTorch. Our implementation only requires the user to supply the function $r(w)$ for evaluating the residuals, which is easy to implement. Our algorithm is therefore applicable to any shallow learning problem of the form (3).

To find a strategy for evaluating $J_w^{\mathrm{T}} J_w p$, we first consider a vector u of the same size as the residuals $r(w)$. By computing the gradient of their inner product, we obtain $\frac{\partial}{\partial w}(r(w)^{\mathrm{T}} u) = \frac{\partial r}{\partial w}^{\mathrm{T}} u = J_w^{\mathrm{T}} u$. In fact, this is the standard operation of the backpropagation procedure, namely to apply the transposed Jacobian at each node in the computational graph, starting at the output. We can thus define backpropagation of a *scalar* function s with respect to a variable v as $\texttt{BackProp}(s, v) = \frac{\partial s}{\partial v}$. Now, as shown above, we have $\texttt{BackProp}(r^{\mathrm{T}} u, w) = J_w^{\mathrm{T}} u$. However, this only accounts for the second product in $J_w^{\mathrm{T}} J_w p$. We first have to compute $J_w p$, which involves the application of the Jacobian itself (not its transpose). Thankfully, the Jacobian of the function $u \mapsto J_w^{\mathrm{T}} u$ is trivially J_w^{T}, since the function is

Algorithm 1 Classification component optimization.

Input: Net weights w, residual function $r(w)$, N_{GN}, N_{CG}

for $i = 1, \dots, N_{\text{GN}}$ **do**
  $r \leftarrow r(w)\,, \quad u \leftarrow r$
  $h \leftarrow \texttt{BackProp}(r^{\text{T}}u, w)$ # Treat u as constant
  $g \leftarrow -h\,, \quad p \leftarrow 0\,, \quad \rho_1 \leftarrow 1\,, \quad \Delta w \leftarrow 0$
  for $n = 1, \dots, N_{\text{CG}}$ **do**
    $\rho_2 \leftarrow \rho_1\,, \quad \rho_1 \leftarrow g^{\text{T}}g\,, \quad \beta \leftarrow \frac{\rho_1}{\rho_2}$
    $p \leftarrow g + \beta p$
    $q_1 \leftarrow \texttt{BackProp}(h^{\text{T}}p, u)$ # Treat p as constant
    $q_2 \leftarrow \texttt{BackProp}(r^{\text{T}}q_1, w)$ # Treat q_1 as constant
    $\alpha \leftarrow \frac{\rho_1}{q_2^{\text{T}}p}$
    $g \leftarrow g - \alpha q_2$
    $\Delta w \leftarrow \Delta w + \alpha p$
  end for
  $w \leftarrow w + \Delta w$
end for

linear. We can therefore transpose it by applying backpropagation. By letting $h := J_w^{\text{T}}u = \texttt{BackProp}(r^{\text{T}}u, w)$, we get $J_w p = \frac{\partial}{\partial u}(h^{\text{T}}p) = \texttt{BackProp}(h^{\text{T}}p, u)$.

Given the above mentioned result, we outline the entire optimization procedure in algorithm 1. It applies N_{GN} Gauss-Newton iterations, each encompassing N_{CG} Conjugate Gradient iterations for minimizing the resulting subproblem (4). Each CG iteration requires two `BackProp` calls for evaluating $q_1 = J_w p$ and $q_2 = J_w^{\text{T}}q_1$, respectively. There is a need for computing $h = J_w^{\text{T}}u$ once in the outer loop. Note that in each call to `BackProp` in algorithm 1, one of the vectors in the inner product is treated as constant, i.e. gradients are not propagated through it. This is highlighted as comments in algorithm 1 for clarity. It is noteworthy that the optimization algorithm is virtually parameter free, only the number of iterations need to be set. In comparison to gradient descent, the CG-based method adaptively computes the learning rate α and momentum β in each iteration. Observe that g is the negative gradient of (4).

3.3. Online Tracking Approach

Our tracker **ATOM** is implemented in Python, using PyTorch. It runs at over 30 FPS on an Nvidia GT-1080 GPU.

Feature extraction: We use ResNet-18 pretrained on ImageNet as our backbone network. For target classification, we employ block 4 features, while the target estimation component uses both block 3 and 4 as input. Features are always extracted from patches of size 288×288 from image regions corresponding to 5 times the estimated target size. Note that ResNet-18 feature extraction is shared and only performed on a single image patch every frame.

Classification Model: The first layer in our classification head (2) consists of a 1×1 convolutional layer w_1, which reduces the feature dimensionality to 64. As in [4], the purpose of this layer is to limit memory and computational requirements. The second layer employs a 4×4 kernel w_2 with a single output channel. We set ϕ_1 to identity since we did not observe any benefit of using a non-linearity at this layer. We use a continuously differentiable parametric exponential linear unit (PELU) [33] as output activation: $\phi_2(t) = t, t \geq 0$ and $\phi_2(t) = \alpha(e^{\frac{t}{\alpha}} - 1), t \leq 0$. Setting $\alpha = 0.05$ allows us to ignore easy negative examples in the loss (3). We found the continuous differentiability of ϕ_2 to be advantageous for optimization.

In the first frame, we perform data augmentation by applying varying degrees of translation, rotation, blur, and dropout, similar to [3], resulting in 30 initial training samples x_j. We then apply algorithm 1 with $N_{\text{GN}} = 6$ and $N_{\text{CG}} = 10$ to optimize the parameters w. Subsequently, we only optimize the final layer w_2, using $N_{\text{GN}} = 1$ and $N_{\text{CG}} = 5$ every 10th frame. In every frame, we add the extracted feature map x_j as a training sample, annotated by a Gaussian y_j centered at the estimated target location. The weights γ_j in (3) are updated with a learning rate of 0.01.

Target Estimation: We first extract features at the previously estimated target location and scale. We then apply the classification model (2) and find the 2D-position with the maximum confidence score. Together with the previously estimated target width and height, this generates the initial bounding box B. While it is possible to perform state estimation using this single proposal, we found that local maxima are better avoided using multiple random initializations. We therefore generate a set of 10 initial proposals by adding uniform random noise to B. The predicted IoU (1) of each box is maximized using 5 gradient ascent iterations with a step length of 1. The final prediction is obtained by taking the mean of the 3 bounding boxes with highest IoU. No further post-processing or filtering, as in e.g. [18] is performed. This refined state also annotates the training sample (x_j, y_j), as described earlier. Note that the modulation vector $c(x_0, B_0)$ in (1) is precomputed in the first frame.

Hard Negative Mining: To further robustify our classification component in the presence of distractors, we adopt a hard negative mining strategy, common in many visual trackers [26, 39]. If a distractor peak is detected in the classification scores, we double the learning rate of this training sample and instantly run a round of optimization with standard settings ($N_{\text{GN}} = 1, N_{\text{CG}} = 5$). We also determine the target as lost if the score falls below 0.25. While the hard negative strategy is not fundamental to our framework, it provides some additional robustness (see section 4.2).

4. Experiments

We evaluate the proposed tracker **ATOM** on five benchmarks: Need for Speed (NFS) [9], UAV123 [24], TrackingNet [25], LaSOT [8], and VOT2018 [17]. Detailed results are provided in the supplementary material.

4.1. IoU Prediction Architecture Analysis

Here, we study the impact of various architectural choices for the IoU prediction module, presented in section 3.1. Our analysis is performed on the combined UAV123 [24] and NFS (30 FPS version) [9] datasets, summing to 223 videos. These datasets contain a high variety of videos that are challenging in many aspects, such as deformation, view change, occlusion, fast motion and distractors. We evaluate the trackers based on the overlap precision metric (OP_T), defined as the percentage of frames having bounding box IoU overlap larger than a threshold T with the ground truth. We also report the area-under-the-curve (AUC) score, defined as $\text{AUC} = \int_0^1 \text{OP}_T \, \mathrm{d}T$. In all experiments, we report the average result over 5 runs.

Reference image: We compare with a baseline approach that excludes target specific information by removing the reference branch in our architecture. That is, the baseline network only uses the test frame to predict the IoU. The results of this investigation are shown in table 1. Excluding the reference frame deteriorates the results by over 5.5% AUC score. This demonstrates the importance of exploiting target-specific appearance information in order to accurately predict the IoU for an arbitrary object.

Integration of target appearance: We investigate different network architectures for integrating the reference image features for IoU prediction. We compare our feature modulation based method, presented in section 3.1, with two alternative architectures. **Concatenation:** Activations from the reference and test branches are concatenated before the final IoU prediction layers. **Siamese:** Using identical architecture for both branches and performing final IoU prediction as a scalar product of their outputs. All the networks are trained using the same setup, with ResNet18 `Block3` and `Block4` features as input. For a fair comparison, we ensure that all networks have the same depth and similar number of trainable parameters. Results are shown in table 1. Naively concatenating the features from the reference image and the test image achieves an AUC of 56.3%. Our Siamese-based architecture obtains better results, with an AUC of 61.7% and $\text{OP}_{0.50}$ of 75.1%. Our modulation-based method further improves the results, giving an absolute gain of 1.2% in $\text{OP}_{0.50}$ and achieves an AUC of 62.3%.

	Baseline (Block 3&4)	Modulation (Block 3&4)	Concatenation (Block 3&4)	Siamese (Block 3&4)	Modulation (Block 3)	Modulation (Block 4)
$\text{OP}_{0.50}(\%)$	68.3	**76.3**	67.5	75.1	73.4	73.6
$\text{OP}_{0.75}(\%)$	38.6	**48.4**	37.9	47.6	44.5	38.9
AUC (%)	56.7	**62.3**	56.3	61.7	60.3	58.5

Table 1. Analysis of different architectures for IoU prediction on the combined NFS and UAV123 datasets. For each method, we indicate in parenthesis the backbone feature layers that are used as input. The baseline approach, which does not employ a reference branch to integrate target specific information, provides poor results. Among the different architectures, the modulation based approach, using both block 3 and 4, achieves the best results.

	ATOM	Multi-Scale	No Classif.	GD	GD++	No HN
$\text{OP}_{0.50}(\%)$	**76.3**	66.2	52.3	74.5	74.8	75.9
$\text{OP}_{0.75}(\%)$	**48.4**	26.0	35.1	47.4	47.3	48.1
AUC (%)	**62.3**	53.7	43.0	60.9	61.1	61.9

Table 2. Impact of each component in the proposed approach on the combined NFS and UAV123 datasets. We compare the target estimation component with the brute-force multi-scale approach and analyze the impact of our classification module, online optimization strategy, and hard-negative mining scheme.

Backbone feature layers: We evaluate the impact of using different feature blocks from the backbone ResNet-18 (table 1). Using features from only `Block3` leads to an AUC of 60.3%, while only `Block4` gives an AUC of 58.5%. Fusing features from both the blocks leads to a significant improvement, giving an AUC score of 62.3%. This indicates that `Block3` and `Block4` features have complementary information useful for predicting the IoU.

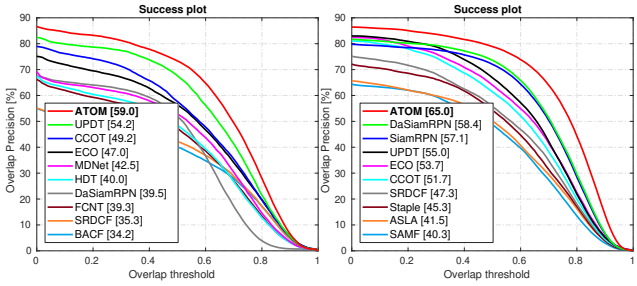
4.2. Ablation Study

We perform an ablation study to demonstrate the impact of each component in the proposed method. We use the same dataset and the evaluation criteria as in section 4.1.

Target Estimation: We compare our target state estimation component, presented in section 3.1, with a brute-force **multi-scale** search approach employing *only* the classification model. This approach mimics the common practice in correlation filter based methods, extracting features at 5 scales with a scale ratio of 1.02. The classification component is then evaluated on all scales, selecting the location and scale with the highest confidence score as the new target state. Results are shown in table 2. Our approach significantly outperforms the multi-scale method by 8.6% in AUC. Further, our approach almost doubles the percentage of highly accurate bounding box predictions, as measured by $\text{OP}_{0.75}$. These results highlight the importance of treating target state estimation as a high-level visual task.

Target Classification: We investigate the impact of the target classification component (section 3.2) by excluding it from our tracking framework. **No Classif** in table 2 only employs the target estimation module for tracking, using a larger search region. The resulting method achieves an AUC of 43.0%, almost 20% less than our approach.

Online Optimization: We investigate the impact of the optimization strategy presented in algorithm 1, by comparing it with gradient descent. We use carefully tuned learning rate and momentum parameters for the gradient descent approach. In the version termed **GD**, we run the same number of `BackProp` operations as in our algorithm, obtaining the same speed as of our tracker. We also compare with **GD++**, running 5 times as many iterations as in **GD**, thus running at significantly slower frame rates. In both cases, the proposed Gauss-Newton approach outperforms gradient descent by



(a) NFS (b) UAV123

Figure 4. Success plots on NFS (a) and UAV123 (b). In both cases, our approach improves the state-of-the-art by a large margin.

	Staple [1]	SAMF [20]	CSRDCF [22]	ECO [4]	DaSiam-RPN [39]	SiamFC [2]	CFNet [34]	MDNet [26]	UPDT [3]	**ATOM**
Precision (%)	47.0	47.7	48.0	49.2	41.3	53.3	53.3	**56.5**	55.7	**64.8**
Norm. Prec. (%)	60.3	59.8	62.2	61.8	60.2	66.6	65.4	**70.5**	70.2	**77.1**
Success (%)	52.8	50.4	53.4	55.4	56.8	57.1	57.8	60.6	**61.1**	**70.3**

Table 3. State-of-the-art comparison on the TrackingNet test set in terms of precision, normalized precision, and success. Our approach significantly outperforms UPDT, achieving a relative gain of 15% in terms of success.

more than 1.2% AUC score (Table 2). Note that even a 5-fold increase of iterations does not provide any significant improvement (only 0.2%), indicating slow convergence.

Hard Negative Mining: We evaluate our method without the **H**ard **N**egative mining component (section 3.3), resulting in an AUC of 61.9%. This suggests the hard negative mining adds some robustness (0.4% AUC) to our tracker.

4.3. State-of-the-art Comparison

We present the comparison of our tracker with state-of-the-art methods on five challenging tracking datasets.

Need For Speed [9]: We evaluate on the 30 FPS version of the dataset. Figure 4a shows the success plot over all the 100 videos, reporting AUC scores in the legend. CCOT [7] and UPDT [3], both based on correlation filters, achieve AUC scores of 49.2% and 54.2% respectively. Our tracker significantly outperforms UPDT with a relative gain of 9%.

UAV123 [24]: Figure 4b displays the success plot over all the 123 videos. DaSiamRPN [39] and its predecessor SiamRPN [18] employ a target estimation component based on bounding box regression. Compared to other approaches, DaSiamRPN achieves a superior AUC of 58.4%, owing to its accuracy. Our tracker, employing an overlap maximization strategy for target estimation, significantly outperforms DaSiamRPN by achieving an AUC of 65.0%.

TrackingNet [25]: This is a recently introduced large-scale dataset consisting of real-world videos sampled from YouTube. The trackers are evaluated using an online evaluation server on a test set of 511 videos. Table 3 shows the results in terms of precision, normalized precision, and success. In terms of precision and success, MDNet [26] achieves scores of 56.5% and 60.6% respectively. Our tracker outperforms MDNet with relative gains of 14% and 16% in terms of precision and success respectively.

	STRCF [19]	SINT [32]	ECO [4]	DSiam [10]	StructSiam [38]	SiamFC [2]	VITAL [30]	MDNet [26]	DaSiam-RPN[39]	**ATOM**
Norm. Prec. (%)	34.0	35.4	33.8	40.5	41.8	42.0	45.3	46.0	**49.6**	**57.6**
Success (%)	30.8	31.4	32.4	33.3	33.5	33.6	39.0	39.7	**41.5**	**51.5**

Table 4. State-of-the-art comparison on the LaSOT dataset in terms of normalized precision and success.

	DLSTpp [17]	SASiamR [11]	CPT [17]	DeepSTRCF [19]	DRT [31]	RCO [17]	UPDT [3]	DaSiam-RPN [39]	MFT [17]	LADCF [36]	**ATOM**
EAO	0.325	0.337	0.339	0.345	0.356	0.376	0.378	0.383	0.385	**0.389**	**0.401**
Robustness	0.224	0.258	0.239	0.215	0.201	**0.155**	0.184	0.276	**0.140**	0.159	0.204
Accuracy	0.543	0.566	0.506	0.523	0.519	0.507	0.536	**0.586**	0.505	0.503	**0.590**

Table 5. State-of-the-art comparison on the public VOT2018 dataset in terms of expected average overlap (EAO), robustness (tracking failure), and accuracy. Our tracker outperforms all the previous methods in terms of EAO.

LaSOT [8]: We evaluate our approach on the test split consisting of 280 videos. Table 4 shows the results in terms of normalized precision and success. Among previous approaches, DaSiamRPN achieves the best success scores. Our approach significantly outperforms DaSiamRPN with an absolute gain of 10.0% in success.

VOT2018 [17]: This dataset consists of 60 videos and the performance is evaluated in terms of robustness (failure rate) and accuracy (average overlap in the course of successful tracking). The two measures are merged in a single metric, Expected Average Overlap (EAO), which provides the overall performance ranking. Table 5 shows the comparison of our approach with the top-10 trackers in the VOT2018 competition [17]. Among the top trackers, only DaSiamRPN uses an explicit target state estimation component, achieving higher accuracy compared to its DCF-based counterparts like LADCF [36] and MFT. Our approach ATOM achieves the best accuracy, while having competitive robustness. Further, our tracker obtains the best EAO score of 0.401, with a relative gain of 3% over LADCF.

5. Conclusions

We propose a novel tracking architecture with explicit components for target estimation and classification. The estimation component is trained offline on large-scale datasets to predict the IoU overlap between the target and a bounding box estimate. Our architecture integrates target-specific knowledge by performing feature modulation. The classification component consists of a two-layer fully convolutional network head and is trained online using a dedicated optimization approach. Comprehensive experiments are performed on four tracking benchmarks. Our approach provides accurate target estimation while being robust against distractor objects in the scene, outperforming previous methods on all four datasets.

Acknowledgments: This work was supported by SSF (SymbiCloud), Swedish Research Council (EMC^2, grant 2018-04673), ELLIIT, and WASP.

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016. 8

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV workshop*, 2016. 2, 8

[3] G. Bhat, J. Johnander, M. Danelljan, F. S. Khan, and M. Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 1, 2, 6, 8

[4] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. ECO: efficient convolution operators for tracking. In *CVPR*, 2017. 1, 5, 6, 8

[5] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *TPAMI*, 39(8):1561–1575, 2017. 2

[6] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 1, 2

[7] M. Danelljan, A. Robinson, F. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 2, 5, 8

[8] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019. 2, 4, 6, 8

[9] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017. 2, 6, 7, 8

[10] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, 2017. 8

[11] A. He, C. Luo, X. Tian, and W. Zeng. Towards a better match in siamese network based visual object tracker. In *ECCV workshop*, 2018. 8

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 5

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015. 1, 2

[14] P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912. 2

[15] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018. 2, 3, 5

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 5

[17] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pfugfelder, L. C. Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, G. Fernandez, and et al. The sixth visual object tracking vot2018 challenge results. In *ECCV workshop*, 2018. 1, 2, 6, 8

[18] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 2, 6, 8

[19] F. Li, C. Tian, W. Zuo, L. Zhang, and M. Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, 2018. 8

[20] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV workshop*, 2014. 2, 8

[21] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 4

[22] A. Lukezic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *IJCV*, 126(7):671–688, 2018. 1, 8

[23] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 2

[24] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 2, 6, 7, 8

[25] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 2, 4, 6, 8

[26] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 2, 6, 8

[27] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017. 3

[28] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[29] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994. 5

[30] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang. VITAL: Visual tracking via adversarial learning. In *CVPR*, 2018. 8

[31] C. Sun, D. Wang, H. Lu, and M. Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018. 1, 5, 8

[32] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 8

[33] L. Trottier, P. Giguère, and B. Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. In *ICMLA*, 2017. 6

[34] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 1, 8

[35] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 2

[36] T. Xu, Z. Feng, X. Wu, and J. Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual tracking. *CoRR*, abs/1807.11348, 2018. 8

[37] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. 1

[38] Y. Zhang, L. Wang, J. Qi, D. K. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *ECCV*, 2018. 8

[39] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 1, 2, 4, 6, 8