Deep Regression Tracking with Shrinkage Loss

Xiankai Lu[1,3⋆], Chao Ma[2⋆], Bingbing Ni[1,4], Xiaokang Yang[1,4], Ian Reid[2], and Ming-Hsuan Yang[5,6]

[1] Shanghai Jiao Tong University [2] The University of Adelaide
[3] Inception Institute of Artificial Intelligence
[4] SJTU-UCLA Joint Center for Machine Perception and Inference
[5] University of California at Merced [6] Google Inc.

Abstract. Regression trackers directly learn a mapping from regularly dense samples of target objects to soft labels, which are usually generated by a Gaussian function, to estimate target positions. Due to the potential for fast-tracking and easy implementation, regression trackers have recently received increasing attention. However, state-of-the-art deep regression trackers do not perform as well as discriminative correlation filters (DCFs) trackers. We identify the main bottleneck of training regression networks as extreme foreground-background data imbalance. To balance training data, we propose a novel shrinkage loss to penalize the importance of easy training data. Additionally, we apply residual connections to fuse multiple convolutional layers as well as their output response maps. Without bells and whistles, the proposed deep regression tracking method performs favorably against state-of-the-art trackers, especially in comparison with DCFs trackers, on five benchmark datasets including OTB-2013, OTB-2015, Temple-128, UAV-123 and VOT-2016.

Keywords: Regression networks, shrinkage loss, object tracking

1 Introduction

The recent years have witnessed growing interest in developing visual object tracking algorithms for various vision applications. Existing tracking-by-detection approaches mainly consist of two stages to perform tracking. The first stage draws a large number of samples around target objects in the previous frame and the second stage classifies each sample as the target object or as the background. In contrast, one-stage regression trackers [1–8] directly learn a mapping from a regularly dense sampling of target objects to soft labels generated by a Gaussian function to estimate target positions. One-stage regression trackers have recently received increasing attention due to their potential to be much faster and simpler than two-stage trackers. State-of-the-art one-stage trackers [1–5] are predominantly on the basis of discriminative correlation filters (DCFs) rather than deep regression networks. Despite the top performance on recent

⋆ The first two authors contribute equally to this work. The corresponding author is C. Ma (`c.ma@adelaide.edu.au`).

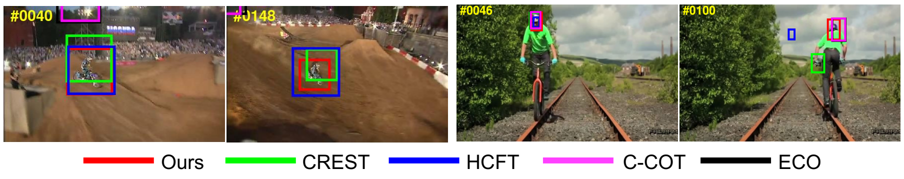


Fig. 1. Tracking results in comparison with state-of-the-art trackers. The proposed algorithm surpasses existing deep regression based trackers (CREST [8]), and performs well against the DCFs trackers (ECO [5], C-COT [4] and HCFT [3]).

benchmarks [9, 10], DCFs trackers take few advantages of end-to-end training as learning and updating DCFs are independent of deep feature extraction. In this paper, we investigate the performance bottleneck of deep regression trackers [6–8], where regression networks are fully differentiable and can be trained end-to-end. As regression networks have greater potential to take advantage of large-scale training data than DCFs, we believe that deep regression trackers can perform at least as well as DCFs trackers.

We identify the main bottleneck impeding deep regression trackers from achieving state-of-the-art accuracy as the data imbalance [11] issue in regression learning. For the two-stage trackers built upon binary classifiers, data imbalance has been extensively studied. That is, positive samples are far less than negative samples and the majority of negative samples belong to easy training data, which contribute little to classifier learning. Despite the pertinence of data imbalance in regression learning as well, we note that current one-stage regression trackers [6–8] pay little attention to this issue. As the evidence of the effectiveness, state-of-the-art DCFs trackers improve tracking accuracy by re-weighting sample locations using Gaussian-like maps [12], spatial reliability maps [13] or binary maps [14]. In this work, to break the bottleneck, we revisit the shrinkage estimator [15] in regression learning. We propose a novel shrinkage loss to handle data imbalance during learning regression networks. Specifically, we use a Sigmoid-like function to penalize the importance of easy samples coming from the background (e.g., samples close to the boundary). This not only improves tracking accuracy but also accelerates network convergence. The proposed shrinkage loss differs from the recently proposed focal loss [16] in that our method penalizes the importance of easy samples only, whereas focal loss partially decreases the loss from valuable hard samples (see Section 3.2).

We observe that deep regression networks can be further improved by best exploiting multi-level semantic abstraction across multiple convolutional layers. For instance, the FCNT [6] fuses two regression networks independently learned on the *conv4-3* and *con5-3* layers of VGG-16 [17] to improve tracking accuracy. However, independently learning regression networks on multiple convolutional layers cannot make full use of multi-level semantics across convolutional layers. In this work, we propose to apply residual connections to respectively fuse multiple convolutional layers as well as their output response maps. All the connections are fully differentiable, allowing our regression network to be trained end-to-end.

For fair comparison, we evaluate the proposed deep regression tracker using the standard benchmark setting, where only the ground-truth in the first frame is available for training. The proposed algorithm performs well against state-of-the-art methods especially in comparison with DCFs trackers. Figure 1 shows such examples on two challenging sequences.

The main contributions of this work are summarized below:

– We propose the novel shrinkage loss to handle the data imbalance issue in learning deep regression networks. The shrinkage loss helps accelerate network convergence as well.
– We apply residual connections to respectively fuse multiple convolutional layers as well as their output response maps. Our scheme fully exploits multi-level semantic abstraction across multiple convolutional layers.
– We extensively evaluate the proposed method on five benchmark datasets. Our method performs well against state-of-the-art trackers. We succeed in narrowing the gap between deep regression trackers and DCFs trackers.

2 Related Work

Visual tracking has been an active research topic with comprehensive surveys [18, 19]. In this section, we first discuss the representative tracking frameworks using the two-stage classification model and the one-stage regression model. We then briefly review the data imbalance issue in classification and regression learning.

Two-Stage Tracking. This framework mainly consists of two stages to perform tracking. The first stage generates a set of candidate target samples around the previously estimated location using random sampling, regularly dense sampling [20], or region proposal [21, 22]. The second stage classifies each candidate sample as the target object or as the background. Numerous efforts have been made to learn a discriminative boundary between positive and negative samples. Examples include the multiple instance learning (MIL) [23] and Struck [24, 25] methods. Recent deep trackers, such as MDNet [26], DeepTrack [27] and CNN-SVM [28], all belong to the two-stage classification framework. Despite the favorable performance on the challenging object tracking benchmarks [9, 10], we note that two-stage deep trackers suffer from heavy computational load as they directly feed samples in the image level into classification neural networks. Different from object detection, visual tracking put more emphasis on slight displacement between samples for precise localization. Two-stage deep trackers benefit little from the recent advance of ROI pooling [29], which cannot highlight the difference between highly spatially correlated samples.

One-Stage Tracking. The one-stage tracking framework takes the whole search area as input and directly outputs a response map through a learned regressor, which learns a mapping between input features and soft labels generated by a Gaussian function. One representative category of one-stage trackers are based on discriminative correlation filters [30], which regress all the circularly shifted versions of input image into soft labels. By computing the correlation as an element-wise product in the Fourier domain, DCFs trackers achieve the fastest speed thus far. Numerous extensions include KCF [31], LCT [32, 33], MCF [34], MCPF [35] and BACF [14]. With the use of deep features, DCFs trackers, such as DeepSRDCF [1], HDT [2], HCFT [3], C-COT [4] and ECO [5], have shown superior performance on benchmark datasets. In [3], Ma et al. propose to learn multiple DCFs over different convolutional layers and empirically fuse output correlation maps to locate target objects. A similar idea is exploited in [4] to combine multiple response maps. In [5], Danelljan et al. reduce feature channels to accelerate learning correlation filters. Despite the top performance, DCFs trackers independently extract deep features to learn and update correlation filters. In the deep learning era, DCFs trackers can hardly benefit from end-to-end training. The other representative category of one-stage trackers are based on convolutional regression networks. The recent FCNT [6], STCT [7], and CREST [8] trackers belong to this category. The FCNT makes the first effort to learn regression networks over two CNN layers. The output response maps from different layers are switched according to their confidence to locate target objects. Ensemble learning is exploited in the STCT to select CNN feature channels. CREST [8] learns a base network as well as a residual network on a single convolutional layer. The output maps of the base and residual networks are fused to infer target positions. We note that current deep regression trackers do not perform as well as DCFs trackers. We identify the main bottleneck as the data imbalance issue in regression learning. By balancing the importance of training data, the performance of one-stage deep regression trackers can be significantly improved over state-of-the-art DCFs trackers.

Data Imbalance. The data imbalance issue has been extensively studied in the learning community [11, 36, 37]. Helpful solutions involve data re-sampling [38–40], and cost-sensitive loss [41–43, 16]. For visual tracking, Li et al. [44] use a temporal sampling scheme to balance positive and negative samples to facilitate CNN training. Bertinetto et al. [45] balance the loss of positive and negative examples in the score map for pre-training the Siamese fully convolution network. The MDNet [26] tracker shows that it is crucial to mine the hard negative samples during training classification networks. The recent work [16] on dense object detection proposes focal loss to decrease the loss from imbalance samples. Despite the importance, current deep regression trackers [6–8] pay little attention to data imbalance. In this work, we propose to utilize shrinkage loss to penalize easy samples which have little contribution to learning regression networks. The proposed shrinkage loss significantly differs from focal loss [16] in that we penalize the loss only from easy samples while keeping the loss of hard samples unchanged, whereas focal loss partially decreases the loss of hard samples as well.

3 Proposed Algorithm

We develop our tracker within the one-stage regression framework. Figure 2 shows an overview of the proposed regression network. To facilitate regression

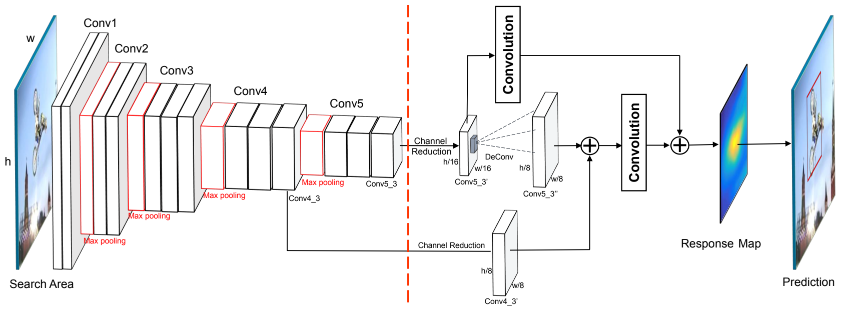


Fig. 2. Overview of the proposed deep regression network for tracking. Left: Fixed feature extractor (VGG-16). Right: Regression network trained in the first frame and updated frame-by-frame. We apply residual connections to both convolution layers and output response maps. The proposed network effectively exploits multi-level semantic abstraction across convolutional layers. With the use of shrinkage loss, our network breaks the bottleneck of data imbalance in regression learning and converges fast.

learning, we propose a novel shrinkage loss to handle data imbalance. We further apply residual connections to respectively fuse convolutional layers and their output response maps for fully exploiting multi-level semantics across convolutional layers. In the following, we first revisit learning deep regression networks briefly. We then present the proposed shrinkage loss in detail. Last, we discuss the residual connection scheme.

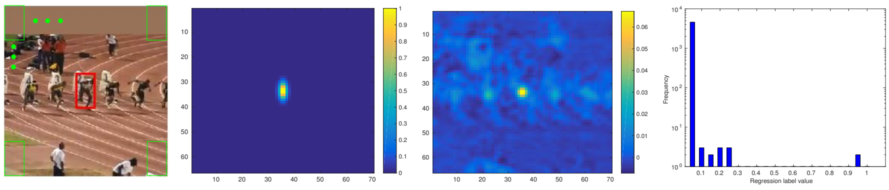
3.1 Convolutional Regression

Convolutional regression networks regress a dense sampling of inputs to soft labels which are usually generated by a Gaussian function. Here, we formulate the regression network as one convolutional layer. Formally, learning the weights of the regression network is to solve the following minimization problem:

$$\arg\min_{\mathbf{W}} \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2, \tag{1}$$

where $*$ denotes the convolution operation and $\mathbf{W}$ denotes the kernel weight of the convolutional layer. Note that there is no bias term in Eq. (1) as we set the bias parameters to 0. $\mathbf{X}$ means the input features. $\mathbf{Y}$ is the matrix of soft labels, and each label $y \in \mathbf{Y}$ ranges from 0 to 1. λ is the regularization term. We estimate the target translation by searching for the location of the maximum value of the output response map. The size of the convolution kernel $\mathbf{W}$ is either fixed (e.g., 5×5) or proportional to the size of the input features $\mathbf{X}$. Let η be the learning rate. We iteratively optimize $\mathbf{W}$ by minimizing the square loss:

$$\begin{aligned} L(\mathbf{W}) &= \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2 \\ \mathbf{W}_t &= \mathbf{W}_{t-1} - \eta \frac{\partial L}{\partial \mathbf{W}}, \end{aligned} \tag{2}$$



(a) Input patch (b) Soft labels $\mathbf{Y}$ (c) Outputs $\mathbf{P}$ (d) Hist. of $|\mathbf{P}-\mathbf{Y}|$

Fig. 3. (a) Input patch. (b) The corresponding soft labels $\mathbf{Y}$ generated by Gaussian function for training. (c) The output regression map $\mathbf{P}$. (d) The histogram of the absolute difference $|\mathbf{P}-\mathbf{Y}|$. Note that easy samples with small absolute difference scores dominate the training data.

3.2 Shrinkage Loss

For learning convolutional regression networks, the input search area has to contain a large body of background surrounding target objects (Figure 3(a)). As the surrounding background contains valuable context information, a large area of the background helps strengthen the discriminative power of target objects from the background. However, this increases the number of easy samples from the background as well. These easy samples produce a large loss in total to make the learning process unaware of the valuable samples close to targets. Formally, we denote the response map in every iteration by $\mathbf{P}$, which is a matrix of size $m \times n$. $p_{i,j} \in \mathbf{P}$ indicates the probability of the position $i \in [1, m], j \in [1, n]$ to be the target object. Let l be the absolute difference between the estimated possibility p and its corresponding soft label y, i.e., $l = |p - y|$. Note that, when the absolute difference l is larger, the sample at the location (i, j) is more likely to be the hard sample and vice versa. Figure 3(d) shows the histogram of the absolute differences. Note that easy samples with small absolute difference scores dominate the training data.

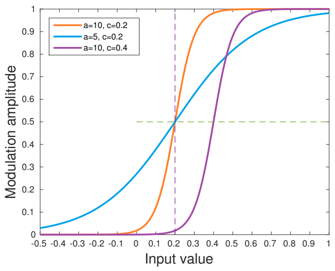
In terms of the absolute difference l, the square loss in regression learning can be formulated as:

$$L_2 = |p - y|^2 = l^2. \tag{3}$$

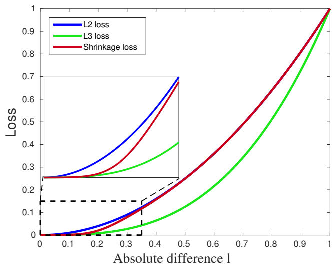
The recent work [16] on dense object detection shows that adding a modulating factor to the entropy loss helps alleviate the data imbalance issue. The modulating factor is a function of the output possibility with the goal to decrease the loss from easy samples. In regression learning, this amounts to re-weighting the square loss using an exponential form of the absolute difference term l as follows:

$$L_F = l^{\gamma} \cdot L_2 = l^{2+\gamma}. \tag{4}$$

For simplicity, we set the parameter γ to 1 as we observe that the performance is not sensitive to this parameter. Hence, the focal loss for regression learning is equal to the L_3 loss, i.e., $L_F = l^3$. Note that, as a weight, the absolute difference l, $l \in [0, 1]$, not only penalizes an easy sample (i.e., $l < 0.5$) but also penalizes a hard sample (i.e., $l > 0.5$). By revisiting the shrinkage estimator [15] and the



(a) Modulating factor



(b) Loss comparison

Fig. 4. (a) Modulating factors in (5) with different hyper-parameters. (b) Comparison between the square loss (L_2), focal loss (L_3) and the proposed shrinkage loss for regression learning. The proposed shrinkage loss only decreases the loss from easy samples ($l < 0.5$) and keeps the loss from hard samples ($l > 0.5$) unchanged.

cost-sensitive weighting strategy [37] in learning regression networks, instead of using the absolute difference l as weight, we propose a modulating factor with respect to l to re-weight the square loss to penalize easy samples only. The modulating function is with the shape of a Sigmoid-like function as:

$$f(l) = \frac{1}{1 + \exp\left(a \cdot (c - l)\right)}, \tag{5}$$

where a and c are hyper-parameters controlling the shrinkage speed and the localization respectively. Figure 4(a) shows the shapes of the modulating function with different hyper-parameters. When applying the modulating factor to weight the square loss, we have the proposed shrinkage loss as:

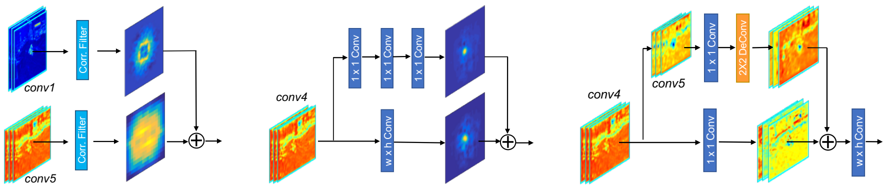
$$L_S = \frac{l^2}{1 + \exp\left(a \cdot (c - l)\right)}. \tag{6}$$

As shown in Figure 4(b), the proposed shrinkage loss only penalizes the importance of easy samples (when $l < 0.5$) and keeps the loss of hard samples unchanged (when $l > 0.5$) when compared to the square loss (L_2). The focal loss (L_3) penalizes both the easy and hard samples.

When applying the shrinkage loss to Eq. (1), we take the cost-sensitive weighting strategy [37] and utilize the values of soft labels as an importance factor, e.g., $\exp(\mathbf{Y})$, to highlight the valuable rare samples. In summary, we rewrite Eq. (1) with the shrinkage loss for learning regression networks as:

$$L_S(\mathbf{W}) = \frac{\exp(\mathbf{Y}) \cdot \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2}{1 + \exp(a \cdot (c - (\mathbf{W} * \mathbf{X} - \mathbf{Y})))} + \lambda \|\mathbf{W}\|^2. \tag{7}$$

We set the value of a to be 10 to shrink the weight function quickly and the value of c to be 0.2 to suit for the distribution of l, which ranges from 0 to 1. Extensive comparison with the other losses shows that the proposed shrinkage loss not only improves the tracking accuracy but also accelerates the training speed (see Section 5.3).



(a) ECO [5] (b) CREST [8] (c) Ours

Fig. 5. Different schemes to fuse convolutional layers. ECO [5] independently learns *correlation filters* over multiple convolutional layers. CREST [8] learns a base and a residual regression network over a single convolutional layer. We first fuse multiple convolutional layers using residual connection and then perform regression learning. Our regression network makes full use of multi-level semantics across multiple convolutional layers rather than merely integrating response maps as ECO and CREST.

3.3 Convolutional Layer Connection

It has been known that CNN models consist of multiple convolutional layers emphasizing different levels of semantic abstraction. For visual tracking, early layers with fine-grained spatial details are helpful in precisely locating target objects; while the later layers maintain semantic abstraction that are robust to significant appearance changes. To exploit both merits, existing deep trackers [3, 5, 6] develop independent models over multiple convolutional layers and integrate the corresponding output response maps with empirical weights. For learning regression networks, we observe that semantic abstraction plays a more important role than spatial detail in dealing with appearance changes. The FCNT exploit both the *conv4* and *conv5* layers and the CREST [8] merely uses the *conv4* layer. Our studies in Section 5.3 also suggest that regression trackers perform well when using the *conv4* and *conv5* layers as the feature backbone. For integrating the response maps generated over convolutional layers, we use a residual connection block to make full use of multiple-level semantic abstraction of target objects. In Figure 3, we compare our scheme with the ECO [5] and CREST [8] methods. The DCFs tracker ECO [5] independently learns *correlation filters* over the *conv1* and *conv5* layers. The CREST [8] learns a base and a residual regression network over the *conv4* layer. The proposed method in Figure 3(c) fuses the *conv4* and *conv5* layers before learning the regression networks. Here we use the deconvolution operation to upsample the *conv*5 layer before connection. We reduce feature channels to ease the computational load as in [46, 47]. Our connection scheme resembles the Option C of constructing the residual network [46]. Ablation studies affirm the effectiveness of this scheme to facilitate regression learning (see Section 5.3).

4 Tracking Framework

We detail the pipeline of the proposed regression tracker. In Figure 2, we show an overview of the proposed deep regression network, which consists of model initialization, target object localization, scale estimation and model update. For training, we crop a patch centered at the estimated location in the previous frame. We use the VGG-16 [17] model as the backbone feature extractor. Specifically, we take the output response of the $conv4\_3$ and $conv5\_3$ layers as features to represent each patch. The fused features via residual connection are fed into the proposed regression network. During tracking, given a new frame, we crop a search patch centered at the estimated position in the last frame. The regression networks take this search patch as input and output a response map, where the location of the maximum value indicates the position of target objects. Once obtaining the estimated position, we carry out scale estimation using the scale pyramid strategy as in [48]. To make the model adaptive to appearance variations, we incrementally update our regression network frame-by-frame. To alleviate noisy updates, the tracked results and soft labels in the last T frames are used for the model update.

5 Experiments

In this section, we first introduce the implementation details. Then, we evaluate the proposed method on five benchmark datasets including OTB-2013 [49], OTB-2015 [9], Temple128 [50], UAV123 [51] and VOT-2016 [10] in comparison with state-of-the-art trackers. Last, we present extensive ablation studies on different types of losses as well as their effect on the convergence speed.

5.1 Implementation Details

We implement the proposed Deep Shrinkage Loss Tracker (DSLT) in Matlab using the Caffe toolbox [52]. All experiments are performed on a PC with an Intel i7 4.0GHz CPU and an NVIDIA TITAN X GPU. We use VGG-16 as the backbone feature extractor. We apply a 1×1 convolution layer to reduce the channels of $conv4\_3$ and $conv5\_3$ from 512 to 128. We train the regression networks with the Adam [53] algorithm. Considering the large gap between maximum values of the output regression maps over different layers, we set the learning rate η to $8e$-7 in $conv5\_3$ and $2e$-8 in $conv4\_3$. During online update, we decrease the learning rates to $2e$-7 and $5e$-9, respectively. The length of frames T for model update is set to 7. The soft labels are generated by a two-dimensional Gaussian function with a kernel width proportional (0.1) to the target size. For scale estimation, we set the ratio of scale changes to 1.03 and the levels of scale pyramid to 3. The average tracking speed including all training process is 5.7 frames per second. The source code is available at https://github.com/chaoma99/DSLT.

5.2 Overall Performance

We extensively evaluate our approach on five challenging tracking benchmarks. We follow the protocol of the benchmarks for fair comparison with state-of-the-art trackers. For the OTB [49, 9] and Temple128 [50] datasets, we report the

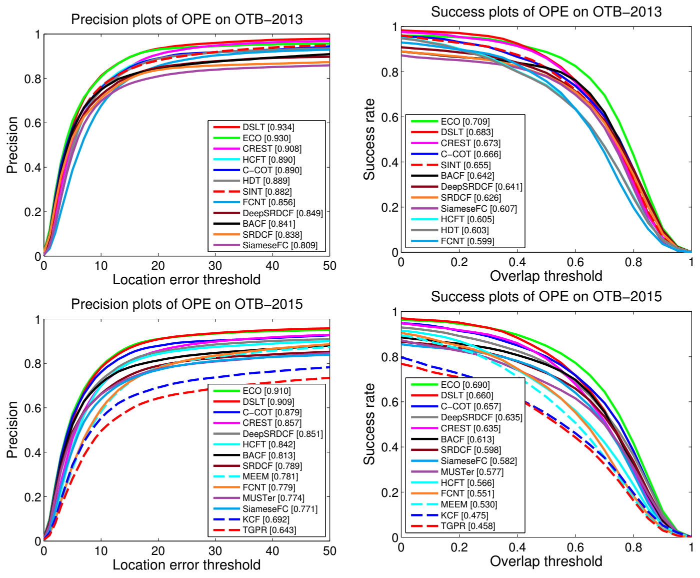


Fig. 6. Overall performance on the OTB-2013 [49] and OTB-2015 [9] datasets using one-pass evaluation (OPE). Our tracker performs well against state-of-the-art methods.

results of one-pass evaluation (OPE) with distance precision (DP) and overlap success (OS) plots. The legend of distance precision plots contains the thresholded scores at 20 pixels, while the legend of overlap success plots contains area-under-the-curve (AUC) scores for each tracker. See the complete results on all benchmark datasets in the supplementary document.

OTB Dataset. There are two versions of this dataset. The OTB-2013 [49] dataset contains 50 challenging sequences and the OTB-2015 [9] dataset extends the OTB-2013 dataset with additional 50 video sequences. All the sequences cover a wide range of challenges including occlusion, illumination variation, rotation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter and low resolution. We fairly compare the proposed DSLT with state-of-the-art trackers, which mainly fall into three categories: (i) one-stage regression trackers including CREST [8], FCNT [6], GOTURN [54], SiameseFC [45]; (ii) one-stage DCFs trackers including ECO [5], C-COT [4], BACF [14], DeepSRDCF [1], HCFT [3], HDT [2], SRDCF [12], KCF [31], and MUSTer [55]; and (iii) two-stage trackers including MEEM [56], TGPR [57], SINT [58], and CNN-SVM [28]. As shown in Figure 6, the proposed DSLT achieves the best distance precision (93.4%) and the second best overlap success (68.3%) on OTB-2013. Our DSLT outperforms the state-of-the-art deep regression trackers (CREST [8] and FCNT [6]) by a large margin. We attribute the favorable performance of our DSLT to two reasons. First, the proposed shrinkage loss effectively alleviate the data imbalance issue in regression learning. As a

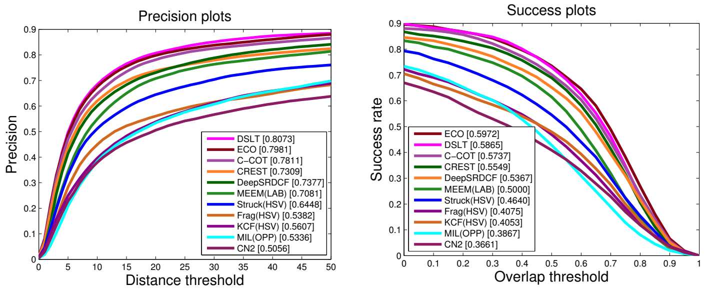


Fig. 7. Overall performance on the Temple Color 128 [50] dataset using one-pass evaluation. Our method ranks first in distance precision and second in overlap success.

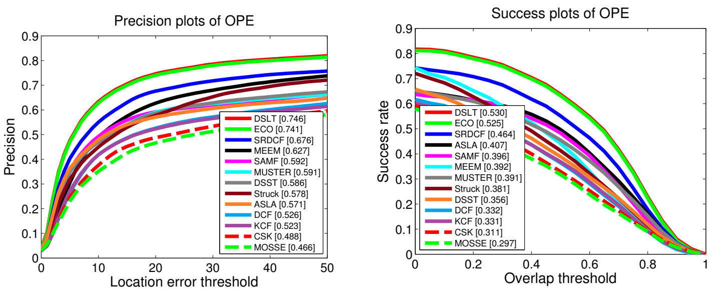


Fig. 8. Overall performance on the UAV-123 [51] dataset using one-pass evaluation (OPE). The proposed DSLT method ranks first.

result, the proposed DSLT can automatically mine the most discriminative samples and eliminate the distraction caused by easy samples. Second, we exploit the residual connection scheme to fuse multiple convolutional layers to further facilitate regression learning as multi-level semantics across convolutional layers are fully exploited. As well, our DSLT performs favorably against all DCFs trackers such as C-COT, HCFT and DeepSRDCF. Note that ECO achieves the best results by exploring both deep features and hand-crafted features. On OTB-2015, our DSLT ranks second in both distance precision and overlap success.

Temple Color 128 Dataset. This dataset [50] consists of 128 colorful video sequences. The evaluation setting of Temple 128 is same to the OTB dataset. In addition to the aforementioned baseline methods, we fairly compare with all the trackers including Struck [24], Frag [59], KCF [31], MEEM [56], MIL [23] and CN2 [47] evaluated by the authors of Temple 128. Figure 7 shows that the proposed method achieves the best distance precision by a large margin compared to the ECO, C-COT and CREST trackers. Our method ranks second in terms of overlap success. It is worth mentioning that our regression tracker performs well in tracking small targets. Temple-128 contains a large number of small target objects. Our method achieves the best precision of 80.73%, far better than the state-of-the-art.

Table 1. Overall performance on VOT-2016 in comparison to the top 7 trackers. EAO: Expected average overlap. AR: Accuracy rank. RR: Robustness rank.

	ECO [5]	C-COT [4]	Staple [63]	CREST [8]	DeepSRDCF [1]	MDNet [26]	SRDCF [12]	DSLT (ours)
EAO	0.3675	0.3310	0.2952	0.2990	0.2763	0.2572	0.2471	0.3321
AR	1.72	1.63	1.82	2.09	1.95	1.78	1.90	1.91
RR	1.73	1.90	1.95	1.95	2.85	2.88	3.18	2.15

UAV123 Dataset. This dataset [51] contains 123 video sequences obtained by unmanned aerial vehicles (UAVs). We evaluate the proposed DSLT with several representative methods including ECO [5], SRDCF [12], KCF [31], MUSTer [55], MEEM [56], TGPR [57], SAMF [60], DSST [58], CSK [61], Struck [24], and TLD [62]. Figure 8 shows that the performance of the proposed DSLT is slightly superior to ECO in terms of distance precision and overlap success rate.

VOT-2016 Dataset. The VOT-2016 [10] dataset contains 60 challenging videos, which are annotated by the following attributes: occlusion, illumination change, motion change, size change, and camera motion. The overall performance is measured by the expected average overlap (EAO), accuracy rank (AR) and robustness rank (RR). The main criteria, EAO, takes into account both the per-frame accuracy and the number of failures. We compare our method with state-of-the-art trackers including ECO [5], C-COT [4], CREST [8], Staple [63], SRDCF [12], DeepSRDCF [1], MDNet [26]. Table 1 shows that our method performs slightly worse than the top performing ECO tracker but significantly better than the others such as the recent C-COT and CREST trackers. The VOT-2016 report [10] suggests a strict state-of-the-art bound as 0.251 with the EAO metric. The proposed DSLT achieves a much higher EAO of 0.3321.

5.3 Ablation Studies

We first analyze the contributions of the loss function and the effectiveness of the residual connection scheme. We then discuss the convergence speed of different losses in regression learning.

Loss Function Analysis. First, we replace the proposed shrinkage loss with square loss (L_2) or focal loss (L_3). We evaluate the alternative implementations on the OTB-2015 [9] dataset. Overall, the proposed DSLT with shrinkage loss significantly advances the square loss (L_2) and focal loss (L_3) by a large margin. We present the qualitative results on two sequences in Figure 9 where the trackers with L_2 loss or L_3 loss both fail to track the targets undergoing large appearance changes, whereas the proposed DSLT can locate the targets robustly. Figure 10 presents the quantitative results on the OTB-2015 dataset. Note that the baseline tracker with L_2 loss performs much better than CREST [8] in both distance precision (87.0% vs. 83.8%) and overlap success (64.2% vs. 63.2%). This clearly proves the effectiveness of the convolutional layer connection scheme,

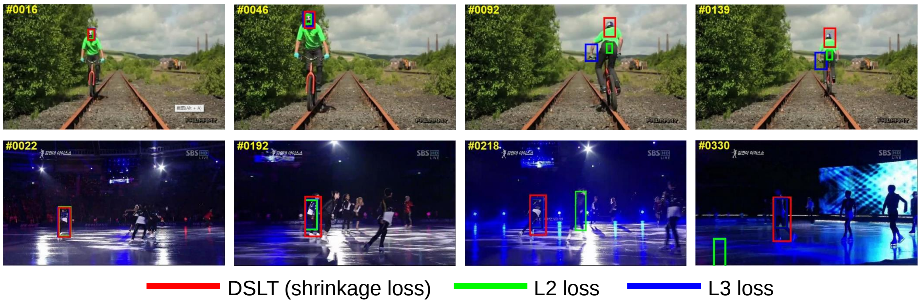


Fig. 9. Quantitative results on the *Biker* and *Skating1* sequences. The proposed DSLT with shrinkage loss can locate the targets more robustly than L_2 loss and L_3 loss.

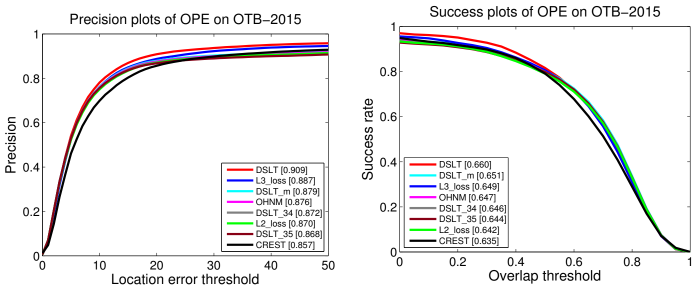


Fig. 10. Ablation studies with different losses and different layer connections on the OTB-2015 [9] dataset.

which applies residual connection to both convolutional layers and output regression maps rather than only to the output regression maps as CREST does. In addition, we implement an alternative approach using online hard negative mining (OHNM) [26] to completely exclude the loss from easy samples. We empirically set the mining threshold to 0.01. Our DSLT outperforms the OHNM method significantly. Our observation is thus well aligned to [16] that easy samples still contribute to regression learning but they should not dominate the whole gradient. In addition, the OHNM method manually sets a threshold, which is hardly applicable to all videos.

Feature Analysis. We further evaluate the effectiveness of convolutional layers. We first remove the connections between convolutional layers. The resulted DSLT_m algorithm resembles the CREST. Figure 10 shows that DSLT_m has performance drops of around 0.3% (DP) and 0.1% (OS) when compared to the DSLT. This affirms the importance of fusing features before regression learning. In addition, we fuse *conv*3_3 with *conv*4_3 or *conv*5_3. The inferior performance of DSLT_34 and DSLT_35 shows that semantic abstraction is more important than spatial detail for learning regression networks. As the kernel size of the

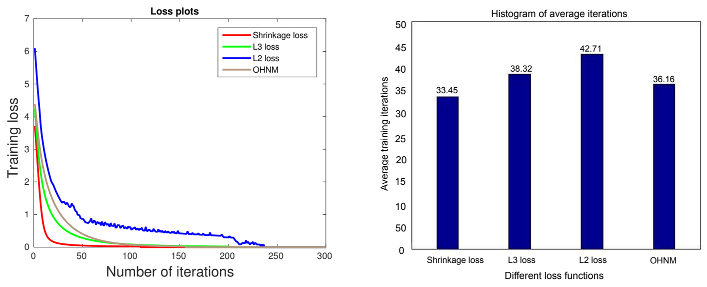


Fig. 11. Training loss plot (left) and average training iterations per sequence on the OTB-2015 dataset (right). The shrinkage loss converges the fastest and requires the least number of iterations to converge.

convolutional regression layer is proportional to the input feature size, we do not evaluate earlier layers for computational efficiency.

Convergence Speed. Figure 11 compares the convergence speed and the required training iterations using different losses on the OTB-2015 dataset [9]. Overall, the training loss using the shrinkage loss descends quickly and stably. The shrinkage loss thus requires the least iterations to converge during tracking.

6 Conclusion

We revisit one-stage trackers based on deep regression networks and identify the bottleneck that impedes one-stage regression trackers from achieving state-of-the-art results, especially when compared to DCFs trackers. The main bottleneck lies in the data imbalance in learning regression networks. We propose the novel shrinkage loss to facilitate learning regression networks with better accuracy and faster convergence speed. To further improve regression learning, we exploit multi-level semantic abstraction of target objects across multiple convolutional layers as features. We apply the residual connections to both convolutional layers and their output response maps. Our network is fully differentiable and can be trained end-to-end. We succeed in narrowing the performance gap between one-stage deep regression trackers and DCFs trackers. Extensive experiments on five benchmark datasets demonstrate the effectiveness and efficiency of the proposed tracker when compared to state-of-the-art algorithms.

Acknowledgments. This work is supported in part by the National Key Research and Development Program of China (2016YFB1001003), NSFC (61527804, 61521062, U1611461, 61502301, and 61671298), the 111 Program (B07022), and STCSM (17511105401 and 18DZ2270700). C. Ma and I. Reid acknowledge the support of the Australian Research Council through the Centre of Excellence for Robotic Vision (CE140100016) and Laureate Fellowship (FL130100102). B. Ni is supported by China's Thousand Youth Talents Plan. M.-H. Yang is supported by NSF CAREER (1149783).

References

1. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: ICCV Workshops. (2015)
2. Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.: Hedged deep tracking. In: CVPR. (2016)
3. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV. (2015)
4. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV. (2016)
5. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: CVPR. (2017)
6. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: ICCV. (2015)
7. Wang, L., Ouyang, W., Wang, X., Lu, H.: STCT: sequentially training convolutional networks for visual tracking. In: CVPR. (2016)
8. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R.W.H., Yang, M.H.: Crest: Convolutional residual learning for visual tracking. In: ICCV. (2017)
9. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. TPAMI **37**(9) (2015)
10. Kristan, M., Leonardis, A., Matas, J., et al.: The visual object tracking VOT2016 challenge results. In: ECCV Workshops. (2016)
11. He, H., Garcia, E.A.: Learning from imbalanced data. TKDE **21**(9) (2009)
12. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV. (2015)
13. Lukezic, A., Vojir, T., Zajc, L.C., Matas, J., Kristan, M.: Discriminative correlation filter with channel and spatial reliability. In: CVPR. (2017)
14. Kiani Galoogahi, H., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: ICCV. (2017)
15. Copas, J.B.: Regression, prediction and shrinkage. Journal of the Royal Statistical Society **45** (1983)
16. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollr, P.: Focal loss for dense object detection. In: ICCV. (2017)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015)
18. Salti, S., Cavallaro, A., di Stefano, L.: Adaptive appearance modeling for video tracking: Survey and evaluation. TIP **21**(10) (2012)
19. Smeulders, A.W.M., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: An experimental survey. TPAMI **36**(7) (2014)
20. Wang, N., Shi, J., Yeung, D.Y., Jia, J.: Understanding and diagnosing visual tracking systems. In: ICCV. (2015)
21. Hua, Y., Alahari, K., Schmid, C.: Online object tracking with proposal selection. In: ICCV. (2015)
22. Zhu, G., Porikli, F., Li, H.: Beyond local search: Tracking objects everywhere with instance-specific proposals. In: CVPR. (2016)
23. Babenko, B., Yang, M., Belongie, S.J.: Robust object tracking with online multiple instance learning. TPAMI **33**(8) (2011)
24. Hare, S., Saffari, A., Torr, P.H.: Struck: Structured output tracking with kernels. In: ICCV. (2011)

25. Ning, J., Yang, J., Jiang, S., Zhang, L., Yang, M.: Object tracking via dual linear structured SVM and explicit feature map. In: CVPR. (2016)
26. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. (2016)
27. Li, H., Li, Y., Porikli, F.: Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In: BMVC. (2014)
28. Hong, S., You, T., Kwak, S., Han, B.: Online tracking by learning discriminative saliency map with convolutional neural network. In: ICML. (2015)
29. Girshick, R.B.: Fast R-CNN. In: ICCV. (2015)
30. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR. (2010)
31. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI **37**(3) (2015)
32. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. In: CVPR. (2015)
33. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Adaptive correlation filters with long-term and short-term memory for object tracking. IJCV (2018) 1–26
34. Wang, M., Liu, Y., Huang, Z.: Large margin object tracking with circulant feature maps. In: CVPR. (2017)
35. Zhang, T., Xu, C., Yang, M.H.: Multi-task correlation particle filter for robust object tracking. In: CVPR. (2017)
36. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: ICDM. (2003)
37. Kukar, M., Kononenko, I.: Cost-sensitive learning with neural networks. In: ECAI. (1998)
38. Huang, C., Li, Y., Loy, C.C., Tang, X.: Learning deep representation for imbalanced classification. In: CVPR. (2016)
39. Dong, Q., Gong, S., Zhu, X.: Class rectification hard mining for imbalanced deep learning. In: ICCV. (2017)
40. Maciejewski, T., Stefanowski, J.: Local neighbourhood extension of SMOTE for mining imbalanced data. In: CIDM. (2011)
41. Khan, S.H., Hayat, M., Bennamoun, M., Sohel, F.A., Togneri, R.: Cost-sensitive learning of deep feature representations from imbalanced data. TNNLS (99) (2017)
42. Tang, Y., Zhang, Y., Chawla, N.V., Krasser, S.: Svms modeling for highly imbalanced classification. IEEE Trans. Cybern **39**(1) (2009)
43. Ting, K.M.: A comparative study of cost-sensitive boosting algorithms. In: ICML. (2000)
44. Li, H., Li, Y., Porikli, F.M.: Robust online visual tracking with a single convolutional neural network. In: ACCV. (2014)
45. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: ECCV Workshops. (2016)
46. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
47. Danelljan, M., Khan, F.S., Felsberg, M., van de Weijer, J.: Adaptive color attributes for real-time visual tracking. In: CVPR. (2014)
48. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC. (2014)
49. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR. (2013)
50. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: Algorithms and benchmark. TIP **24**(12) (2015)

51. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for UAV tracking. In: ECCV. (2016)
52. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: ACMMM. (2014)
53. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)
54. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 FPS with deep regression networks. In: ECCV. (2016)
55. Hong, Z., Chen, Z., Wang, C., Mei, X., Prokhorov, D.V., Tao, D.: Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In: CVPR. (2015)
56. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: ECCV. (2014)
57. Gao, J., Ling, H., Hu, W., Xing, J.: Transfer learning based visual tracking with gaussian processes regression. In: ECCV. (2014)
58. Tao, R., Gavves, E., Smeulders, A.W.M.: Siamese instance search for tracking. In: CVPR. (2016)
59. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR. (2006)
60. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: ECCV 2014 Workshops. (2014) 254–265
61. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.P.: Exploiting the circulant structure of tracking-by-detection with kernels. In: ECCV. (2012) 702–715
62. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7) (2012) 1409–1422
63. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: CVPR. (2016)