

Detect to Track and Track to Detect

Christoph Feichtenhofer
Graz University of Technology
feichtenhofer@tugraz.at

Axel Pinz
Graz University of Technology
axel.pinz@tugraz.at

Andrew Zisserman
University of Oxford
az@robots.ox.ac.uk

Abstract

Recent approaches for high accuracy detection and tracking of object categories in video consist of complex multistage solutions that become more cumbersome each year. In this paper we propose a ConvNet architecture that jointly performs detection and tracking, solving the task in a simple and effective way.

Our contributions are threefold: (i) we set up a ConvNet architecture for simultaneous detection and tracking, using a multi-task objective for frame-based object detection and across-frame track regression; (ii) we introduce correlation features that represent object co-occurrences across time to aid the ConvNet during tracking; and (iii) we link the frame level detections based on our across-frame tracklets to produce high accuracy detections at the video level. Our ConvNet architecture for spatiotemporal object detection is evaluated on the large-scale ImageNet VID dataset where it achieves state-of-the-art results. Our approach provides better single model performance than the winning method of the last ImageNet challenge while being conceptually much simpler. Finally, we show that by increasing the temporal stride we can dramatically increase the tracker speed.

1. Introduction

Object detection in images has received a lot of attention over the last years with tremendous progress mostly due to the emergence of deep Convolutional Networks [12, 19, 21, 36, 37] and their region based descendants [3, 9, 10, 31]. In the case of object detection and tracking in videos, recent approaches have mostly used detection as a first step, followed by post-processing methods such as applying a tracker to propagate detection scores over time. Such variations on the ‘tracking by detection’ paradigm have seen impressive progress but are dominated by frame-level detection methods.

Object detection in video has seen a surge in interest

Christoph Feichtenhofer is a recipient of a DOC Fellowship of the Austrian Academy of Sciences at the Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology.



Figure 1. Challenges for video object detection. Training examples for: (a) bicycle, bird, rabbit; (b) dog; (c) fox; and (d) red panda.

lately, especially since the introduction of the ImageNet [32] video object detection challenge (VID). Different from the ImageNet object detection (DET) challenge, VID shows objects in image sequences and comes with additional challenges of (i) size: the sheer number of frames that video provides (VID has around 1.3M images, compared to around 400K in DET or 100K in COCO [22]), (ii) motion blur: due to rapid camera or object motion, (iii) quality: internet video clips are typically of lower quality than static photos, (iv) partial occlusion: due to change in objects/viewer positioning, and (v) pose: unconventional object-to-camera poses are frequently seen in video. In Fig. 1, we show example images from the VID dataset; for more examples please see¹.

To solve this challenging task, recent top entries in the ImageNet [32] video detection challenge use exhaustive post-processing on top of frame-level detectors. For example, the winner [17] of ILSVRC’15 uses two multi-stage Faster R-CNN [31] detection frameworks, context suppression, multi-scale training/testing, a ConvNet tracker [38], optical-flow based score propagation and model ensembles.

In this paper we propose a unified approach to tackle the problem of object detection in realistic video. Our objective is to directly infer a ‘tracklet’ over multiple frames by simultaneously carrying out detection and tracking with a ConvNet. To achieve this we propose to extend the R-FCN [3] detector with a tracking formulation that is inspired by

¹<http://vision.cs.unc.edu/ilsvrc2015/ui/vid>

current correlation and regression based trackers [1, 13, 25]. We train a fully convolutional architecture end-to-end using a detection and tracking based loss and term our approach D&T for joint Detection and Tracking. The input to the network consists of multiple frames which are first passed through a ConvNet trunk (*e.g.* a ResNet-101 [12]) to produce convolutional features which are shared for the task of detection and tracking. We compute convolutional cross-correlation between the feature responses of adjacent frames to estimate the local displacement at different feature scales. On top of the features, we employ an RoI-pooling layer [3] to classify and regress box proposals as well as an RoI-tracking layer that regresses box transformations (translation, scale, aspect ratio) across frames. Finally, to infer long-term tubes of objects across a video we link detections based on our tracklets.

An evaluation on the large-scale ImageNet VID dataset shows that our approach is able to achieve better single-model performance than the winner of the last ILSVRC'16 challenge, despite being conceptually simple and much faster. Moreover, we show that including a tracking loss may improve feature learning for better static object detection, and we also present a very fast version of D&T that works on temporally-strided input frames.

Our code and models are available at <https://github.com/feichtenhofer/Detect-Track>

2. Related work

Object detection. Two families of detectors are currently popular: First, region proposal based detectors R-CNN [10], Fast R-CNN [9], Faster R-CNN [31] and R-FCN [3] and second, detectors that directly predict boxes for an image in one step such as YOLO [30] and SSD [23].

Our approach builds on R-FCN [3] which is a simple and efficient framework for object detection on region proposals with a fully convolutional nature. In terms of accuracy it is competitive with Faster R-CNN [31] which uses a multi-layer network that is evaluated per-region (and thus has a cost growing linearly with the number of candidate RoIs). R-FCN reduces the cost for region classification by pushing the region-wise operations to the end of the network with the introduction of a position-sensitive RoI pooling layer which works on convolutional features that encode the spatially subsampled class scores of input RoIs.

Tracking Tracking is also an extensively studied problem in computer vision with most recent progress devoted to trackers operating on deep ConvNet features. In [26] a ConvNet is fine-tuned at test-time to track a target from the same video via detection and bounding box regression. Training on the examples of a test sequence is slow and also not applicable in the object detection setting. Other methods use pre-trained ConvNet features to track and have

achieved strong performance either with correlation [1, 25] or regression trackers on heat maps [38] or bounding boxes [13]. The regression tracker in [13] is related to our method. It is based on a Siamese ConvNet that predicts the location in the second image of the object shown in the center of the previous image. Since this tracker predicts a bounding box instead of just the position, it is able to model changes in scale and aspect of the tracked template. The major drawback of this approach is that it only can process a single target template and it also has to rely on significant data augmentation to learn all possible transformations of tracked boxes. The approach in [1] is an example of a correlation tracker and inspires our method. The tracker also uses a fully-convolutional Siamese network that takes as input the tracking template and the search image. The ConvNet features from the last convolutional layer are correlated to find the target position in the response map. One drawback of many correlation trackers [1, 25] is that they only work on single targets and do not account for changes in object scale and aspect ratio.

Video object detection. Action detection is also a related problem and has received increased attention recently, mostly with methods building on two-stream ConvNets [35]. In [11] a method is presented that uses a two-stream R-CNN [10] to classify regions and link them across frames based on the action predictions and their spatial overlap. This method has been adopted by [33] and [27] where the R-CNN was replaced by Faster R-CNN with the RPN operating on two streams of appearance and motion information.

One area of interest is learning to detect and localize in each frame (*e.g.* in video co-localization) with only weak supervision. The YouTube Object Dataset [28], has been used for this purpose, *e.g.* [15, 20].

Since the object detection from video task has been introduced at the ImageNet challenge, it has drawn significant attention. In [18] tubelet proposals are generated by applying a tracker to frame-based bounding box proposals. The detector scores across the video are re-scored by a 1D CNN model. In their corresponding ILSVRC submission the group [17] added a propagation of scores to nearby frames based on optical flows between frames and suppression of class scores that are not among the top classes in a video. A more recent work [16] introduces a tubelet proposal network that regresses static object proposals over multiple frames, extracts features by applying Faster R-CNN which are finally processed by an encoder-decoder LSTM. In deep feature flow [40] a recognition ConvNet is applied to key frames only and an optical flow ConvNet is used for propagating the deep feature maps via a flow field to the rest of the frames. This approach can increase detection speed by a factor of 5 at a slight accuracy cost. The approach is error-prone due largely to two aspects: First,

propagation from the key frame to the current frame can be erroneous and, second, the key frames can miss features from current frames. Very recently a new large-scale dataset for video object detection has been introduced [29] with single objects annotations over video sequences.

3. D&T Approach

In this section we first give an overview of the Detect and Track (D&T) approach (Sect. 3.1) that generates tracklets given two (or more) frames as input. We then give the details, starting with the baseline R-FCN detector [3] (Sect. 3.2), and formulating the tracking objective as cross-frame bounding box regression (Sect. 3.3); finally, we introduce the correlation features (Sect. 3.4) that aid the network in the tracking process.

Sect. 4 shows how we link across-frame tracklets to tubes over the temporal extent of a video, and Sect. 5 describes how we apply D&T to the ImageNet VID challenge.

3.1. D&T overview

We aim at jointly detecting and tracking (D&T) objects in video. Fig. 2 illustrates our D&T architecture. We build on the R-FCN [3] object detection framework which is fully convolutional up to region classification and regression, and extend it for multi-frame detection and tracking. Given a set of two high-resolution input frames our architecture first computes convolutional feature maps that are shared for the tasks of detection and tracking (*e.g.* the features of a ResNet-101[12]). An RPN is used to propose candidate regions in each frame based on the objectness likelihood for pre-defined candidate boxes (*i.e.* “anchors”[31]). Based on these regions, RoI pooling is employed to aggregate position-sensitive score and regression maps, produced from intermediate convolutional layers, to classify boxes and refine their coordinates (regression), respectively.

We extend this architecture by introducing a regressor that takes the intermediate position-sensitive regression maps from both frames (together with correlation maps, see below) as input to an RoI tracking operation which outputs the box transformation from one frame to the other. The correspondence between frames is thus simply accomplished by pooling features from both frames, at the same proposal region. We train the RoI tracking task by extending the multi-task objective of R-FCN with a tracking loss that regresses object coordinates across frames. Our tracking loss operates on ground truth objects and evaluates a soft L1 norm [9] between coordinates of the predicted track and the ground truth track of an object.

Such a tracking formulation can be seen as a multi-object extension of the single target tracker in [13] where a ConvNet is trained to infer an object’s bounding box from features of the two frames. One drawback of such an approach is that it does not exploit translational equivariance

which means that the tracker has to learn all possible translations from training data. Thus such a tracker requires exceptional data augmentation (artificially scaling and shifting boxes) during training [13].

A tracking representation that is based on correlation filters [2, 4, 14] can exploit the translational equivariance as correlation is equivariant to translation. Recent correlation trackers [1, 25] typically work on high-level ConvNet features and compute the cross correlation between a tracking template and the search image (or a local region around the tracked position from the previous frame). The resulting correlation map measures the similarity between the template and the search image for all circular shifts along the horizontal and vertical dimension. The displacement of a target object can thus be found by taking the maximum of the correlation response map.

Different from typical correlation trackers that work on single target templates, we aim to track multiple objects simultaneously. We compute correlation maps for all positions in a feature map and let RoI tracking additionally operate on these feature maps for better track regression. Our architecture is able to be trained end-to-end taking as input frames from a video and producing object detections and their tracks. The next sections describe how we structure our architecture for end-to-end learning of object detection and tracklets.

3.2. Object detection and tracking in R-FCN

Our architecture takes frames $I^t \in \mathbb{R}^{H_0 \times W_0 \times 3}$ at time t and pushes them through a backbone ConvNet (*i.e.* ResNet-101 [12]) to obtain feature maps $x_l^t \in \mathbb{R}^{H_l \times W_l \times D_l}$ where W_l , H_l and D_l are the width, height and number of channels of the respective feature map output by layer l . As in R-FCN [3] we reduce the effective stride at the last convolutional layer from 32 pixels to 16 pixels by modifying the conv5 block to have unit spatial stride, and also increase its receptive field by dilated convolutions [24].

Our overall system builds on the R-FCN [3] object detector which works in two stages: first it extracts candidate regions of interest (RoI) using a Region Proposal Network (RPN) [31]; and, second, it performs region classification into different object categories and background by using a position-sensitive RoI pooling layer [3]. The input to this RoI pooling layer comes from an extra convolutional layer with output x_{cls}^t that operates on the last convolutional layer of a ResNet [12]. The layer produces a bank of $D_{cls} = k^2(C + 1)$ position-sensitive score maps which correspond to a $k \times k$ spatial grid describing relative positions to be used in the RoI pooling operation for each of the C categories and background. Applying the softmax function to the outputs leads to a probability distribution p over $C + 1$ classes for each RoI. In a second branch R-FCN puts a sibling convolutional layer with output x_{reg}^t after the

Figure 2. Architecture of our Detect and Track (D&T) approach (see Section 3 for details).

last convolutional layer for bounding box regression, again a position-sensitive RoI pooling operation is performed on this bank of $D_{cls} = 4k^2$ maps for class-agnostic bounding box prediction of a box $b = (b_x, b_y, b_w, b_h)$.

Let us now consider a pair of frames t, t^+ , sampled at time t and t^+ , given as input to the network. We introduce an inter-frame bounding box regression layer that performs position sensitive RoI pooling on the concatenation of the bounding box regression features $\{x_{reg}^t, x_{reg}^{t^+}\}$ to predict the transformation $t^+ = (x^+, y^+, w^+, h^+)$ of the RoIs from t to t^+ . The correlation features, that are also used by the bounding box regressors, are described in section 3.4. Fig. 3 shows an illustration of this approach.

3.3. Multitask detection and tracking objective

To learn this regressor, we extend the multi-task loss of Fast R-CNN [9], consisting of a combined classification L_{cls} and regression loss L_{reg} , with an additional term that scores the tracking across two frames L_{tra} . For a single iteration and a batch of N RoIs the network predicts softmax probabilities $\{p_i\}_{i=1}^N$, regression offsets $\{b_i\}_{i=1}^N$, and cross-frame RoI-tracks $\{t_i^{t^+}\}_{i=1}^{N_{tra}}$. Our overall objective function is written as:

$$\begin{aligned} L(\{p_i\}, \{b_i\}, \{t_i^{t^+}\}) = & \frac{1}{N} \sum_{i=1}^N L_{cls}(p_{i,c}) \\ & + \frac{1}{N_{fg}} \sum_{i=1}^N [c_i > 0] L_{reg}(b_i, b_i) \\ & + \frac{1}{N_{tra}} \sum_{i=1}^{N_{tra}} L_{tra}(t_i^{t^+}, t_i^{t^+}). \end{aligned} \quad (1)$$

The ground truth class label of an RoI is defined by c_i and its predicted softmax score is $p_{i,c}$. b_i is the ground truth regression target, and $t_i^{t^+}$ is the track regression target. The indicator function $[c_i > 0]$ is 1 for fore-

ground RoIs and 0 for background RoIs (with $c_i = 0$). $L_{cls}(p_{i,c}) = -\log(p_{i,c})$ is the cross-entropy loss for box classification, and L_{reg} & L_{tra} are bounding box and track regression losses defined as the smooth L1 function in [9]. The tradeoff parameter is set to $\lambda = 1$ as in [3, 9]. The assignment of RoIs to ground truth is as follows: a class label c and regression targets b are assigned if the RoI overlaps with a ground-truth box at least by 0.5 in intersection-over-union (IoU) and the tracking target t^{t^+} is assigned only to ground truth targets which are appearing in both frames. Thus, the first term of (1) is active for all N boxes in a training batch, the second term is active for N_{fg} foreground RoIs and the last term is active for N_{tra} ground truth RoIs which have a track correspondence across the two frames.

For track regression we use the bounding box regression parametrisation of R-CNN [9, 10, 31]. For a single object we have ground truth box coordinates $B^t = (B_x^t, B_y^t, B_w^t, B_h^t)$ in frame t , and similarly B^{t^+} for frame t^+ , denoting the horizontal & vertical centre coordinates and its width and height. The tracking regression values for the target $t^{t^+} = \{x^{t^+}, y^{t^+}, w^{t^+}, h^{t^+}\}$ are then

$$x^{t^+} = \frac{B_x^{t^+} - B_x^t}{B_w^t}, \quad y^{t^+} = \frac{B_y^{t^+} - B_y^t}{B_h^t} \quad (2)$$

$$w^{t^+} = \log\left(\frac{B_w^{t^+}}{B_w^t}\right), \quad h^{t^+} = \log\left(\frac{B_h^{t^+}}{B_h^t}\right). \quad (3)$$

3.4. Correlation features for object tracking

Different from typical correlation trackers on single target templates, we aim to track multiple objects simultaneously. We compute correlation maps for all positions in a feature map and let RoI pooling operate on these feature maps for track regression. Considering all possible circular shifts in a feature map would lead to large output dimensionality and also produce responses for too large displace-

Figure 3. Schematic of our approach for two frames at time t and $t + 1$. The inputs are first passed through a fully-convolutional network to produce feature maps. A correlation layer operates on multiple feature maps of different scales (only the coarsest scale is shown in the figure) and estimates local feature similarity for various offsets between the two frames. Finally, position sensitive RoI-pooling [3] operates on the convolutional features of the individual frames to produce per-frame detections and also on a stack of individual frame-features as well as the between frame correlation features to output regression offsets of the boxes across the two frames (RoI-tracking).

ments. Therefore, we restrict correlation to a local neighbourhood. This idea was originally used for optical flow estimation in [5], where a correlation layer is introduced to aid a ConvNet in matching feature points between frames. The correlation layer performs point-wise feature comparison of two feature maps x_i^t, x_i^{t+1}

$$x_{\text{corr}}^{t,t+1}(i,j,p,q) = x_i^t(i,j), x_i^{t+1}(i+p,j+q) \quad (4)$$

where $-d \leq p \leq d$ and $-d \leq q \leq d$ are offsets to compare features in a square neighbourhood around the locations i, j in the feature map, defined by the maximum displacement, d . Thus the output of the correlation layer is a feature map of size $x_{\text{corr}} \in \mathbb{R}^{H_i \times W_i \times (2d+1) \times (2d+1)}$. Equation (4) can be seen as a correlation of two feature maps within a local square window defined by d . We compute this local correlation for features at layers conv3, conv4 and conv5 (we use a stride of 2 in i, j to have the same size in the conv3 correlation). We show an illustration of these features for two sample sequences in Fig. 4.

To use these features for track-regression, we let RoI pooling operate on these maps by stacking them with the bounding box features in Sect. 3.2 $\{x_{\text{corr}}^{t,t+1}, x_{\text{reg}}^t, x_{\text{reg}}^{t+1}\}$.

4. Linking tracklets to object tubes

One drawback of high-accuracy object detection is that high-resolution input images have to be processed which

puts a hard constraint on the number of frames a (deep) architecture can process in one iteration (due to memory limitations in GPU hardware). Therefore, a tradeoff between the number of frames and detection accuracy has to be made. Since video possesses a lot of redundant information and objects typically move smoothly in time we can use our inter-frame tracks to link detections in time and build long-term object tubes. To this end, we adopt an established technique from action localization [11, 27, 33], which is used to link frame detections in time to tubes.

Consider the class detections for a frame at time t , $D_i^{t,c} = \{x_i^t, y_i^t, w_i^t, h_i^t, p_{i,c}^t\}$, where $D_i^{t,c}$ is a box indexed by i , centred at (x_i^t, y_i^t) with width w_i^t and height h_i^t , and $p_{i,c}^t$ is the softmax probability for class c . Similarly, we also have tracks $T_i^{t,t+1} = \{x_i^t, y_i^t, w_i^t, h_i^t; x_i^{t+1}, y_i^{t+1}, w_i^{t+1}, h_i^{t+1}\}$ that describe the transformation of the boxes from frame t to $t + 1$. We can now define a class-wise linking score that combines detections and tracks across time

$$s_c(D_i^{t,c}, D_j^{t+1,c}, T^{t,t+1}) = p_{i,c}^t + p_{j,c}^{t+1} + (D_i^t, D_j, T^{t,t+1}) \quad (5)$$

where the pairwise score is

$$(D_i^t, D_j, T^{t,t+1}) = \begin{cases} 1, & \text{if } D_i^t, D_j \in T^{t,t+1}, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Here, the pairwise term evaluates to 1 if the IoU over-

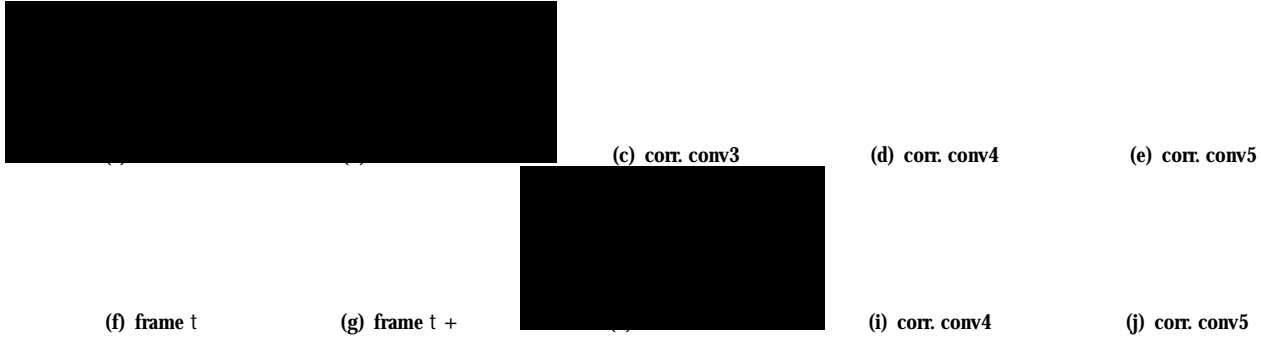


Figure 4. Correlation features for two frames of two validation videos. For the frames in (a) & (b), we show in (c),(d) and (e) the correlation maps computed by using features from conv3, conv4 and conv5, respectively. The feature maps are shown as arrays with the centre map corresponding to zero offsets p, q between the frames and the neighbouring rows and columns correspond to shifted correlation maps of increasing p, q . We observe that the airplane moves to the top-right; hence the feature maps corresponding to $p = 2, q = 3$ show strong responses (highlighted in red). Note that the features at conv4 and conv5 have the same resolution, whereas at conv3 we use stride 2 correlation sampling to produce equal sized outputs. In (h),(i) and (j) we show additional multiscale correlation maps for the frames in (f) & (g) which are affected by camera motion resulting in correlation patterns that correctly estimate this at the lower layer (conv3 corr. responds on the grass and legs of the animal (h)), and also handles the independent motion of the animals at the higher conv5 corr (j).

lap a track correspondences $T^{t,t+}$ with the detection boxes D_i^t, D_j^{t+} is larger than 0.5. This is necessary, because the output of the track regressor does not have to exactly match the output of the box regressor.

The optimal path across a video can then be found by maximizing the scores over the duration T of the video [11]

$$\bar{D}_c = \underset{D}{\operatorname{argmax}} \frac{1}{T} \sum_{t=1}^T s_c(D^t, D^{t+}, T^{t,t+}). \quad (7)$$

Eq. (7) can be solved efficiently by applying the Viterbi algorithm [11]. Once the optimal tube \bar{D}_c is found, the detections corresponding to that tube are removed from the set of regions and (7) is applied again to the remaining regions.

After having found the class-specific tubes \bar{D}_c for one video, we re-weight all detection scores in a tube by adding the mean of the $\alpha = 50\%$ highest scores in that tube. We found that overall performance is largely robust to that parameter, with less than 0.5% mAP variation when varying 10% α to 100%. Our simple tube-based re-weighting aims to boost the scores for positive boxes on which the detector fails. Using the highest scores of a tube for re-weighting acts as a form of non-maximum suppression. It is also inspired by the hysteresis tracking in the Canny edge detector. Our reweighting assumes that the detector fails at most in half of a tubes frames, and improves robustness of the tracker, though the performance is quite insensitive to the proportion chosen (α). Note that our approach enforces the tube to span the whole video and, for simplicity, we do not prune any detections in time. Removing detections with subsequent low scores along a tube (e.g. [27, 33]) could clearly improve the results, but we leave that for future work. In the following section our approach is applied to the video object detection task.

5. Experiments

5.1. Dataset sampling and evaluation

We evaluate our method on the ImageNet [32] object detection from video (VID) dataset² which contains 30 classes in 3862 training and 555 validation videos. The objects have ground truth annotations of their bounding box and track ID in a video. Since the ground truth for the test set is not publicly available, we measure performance as mean average precision (mAP) over the 30 classes on the validation set by following the protocols in [16, 17, 18, 40], as is standard practice.

The 30 object categories in ImageNet VID are a subset of the 200 categories in the ImageNet DET dataset. Thus we follow previous approaches [16, 17, 18, 40] and train our R-FCN detector on an intersection of ImageNet VID and DET set (only using the data from the 30 VID classes). Since the DET set contains large variations in the number of samples per class, we sample at most 2k images per class from DET. We also subsample the VID training set by using only 10 frames from each video. The subsampling reduces the effect of dominant classes in DET (e.g. there are 56K images for the dog class in the DET training set) and very long video sequences in the VID training set.

5.2. Training and testing

RPN. Our RPN is trained as originally proposed [31]. We attach two sibling convolutional layers to the stride-reduced ResNet-101 (Sect. 3.2) to perform proposal classification and bounding box regression at 15 anchors corresponding to 5 scales and 3 aspect ratios. As in [31] we also extract proposals from 5 scales and apply non-maximum suppression (NMS) with an IoU threshold of 0.7 to select the top

²<http://www.image-net.org/challenges/LSVRC/>

300 proposals in each frame for training/testing our R-FCN detector. We found that pre-training on the full ImageNet DET set helps to increase the recall; thus, our RPN is first pre-trained on the 200 classes of ImageNet DET before fine-tuning on only the 30 classes which intersect ImageNet DET and VID. Our 300 proposals per image achieve a mean recall of 96.5% on the ImageNet VID validation set.

R-FCN. Our R-FCN detector is trained similar to [3, 40]. We use the stride-reduced ResNet-101 with dilated convolution in conv5 (see Sect. 3.2) and online hard example mining [34]. A randomly initialized 3×3 , dilation 6 convolutional layer is attached to conv5 for reducing the feature dimension to 512 [40] (in the original R-FCN this is a 1×1 convolutional layer without dilation and an output dimension of 1024). For object detection and box regression, two sibling 1×1 convolutional layers provide the $D_{cls} = k^2(C + 1)$ and $D_{reg} = 4k^2$ inputs to the position-sensitive RoI pooling layer. We use a $k \times k = 7 \times 7$ spatial grid for encoding relative positions as in [3].

In both training and testing, we use single scale images with shorter dimension of 600 pixels. We use a batch size of 4 in SGD training and a learning rate of 10^{-3} for 60K iterations followed by a learning rate of 10^{-4} for 20K iterations. For testing we apply NMS with IoU threshold of 0.3.

D & T. For training our D&T architecture we start with the R-FCN model from above and further fine-tune it on the full ImageNet VID training set with randomly sampling a set of two adjacent frames from a different video in each iteration. In each other iteration we also sample from the ImageNet DET training set to avoid biasing our model to the VID training set. When sampling from the DET set we send the same two frames through the network as there are no sequences available. Besides not forgetting the images from the DET training set, this has an additional beneficial effect of letting our model prefer small motions over large ones (e.g. the tracker in [13] samples motion augmentation from a Laplacian distribution with zero mean to bias a regression tracker on small displacements). Our correlation features (4) are computed at layers conv3, conv4 and conv5 with a maximum displacement of $d = 8$ and a stride of 2 in i, j for the the conv3 correlation. For training, we use a learning rate of 10^{-4} for 40K iterations and 10^{-5} for 20K iterations at a batch size of 4. During testing our architecture is applied to a sequence with temporal stride τ , predicting detections D and tracklets T between them. For object-centred tracks, we use the regressed frame boxes as input of the ROI-tracking layer. We perform non-maximum suppression with bounding-box voting [8] before the tracklet linking step to reduce the number of detections per image and class to 25. These detections are then used in eq. (7) to extract tubes and the corresponding detection boxes are re-weighted as outlined in Sect. 4 for evaluation.

5.3. Results

We show experimental results for our models and the current state-of-the-art in Table 1.

Frame level methods. First we compare methods working on single frames without any temporal processing. Our R-FCN baseline achieves 74.2% mAP which compares favourably to the best performance of 73.9% mAP in [40]. We think our slightly better accuracy comes from the use of 15 anchors for RPN instead of the 9 anchors in [40]. The Faster R-CNN models working as single frame baselines in [18], [16] and [17] score with 45.3%, 63.0% and 63.9%, respectively. We think their lower performance is mostly due to the difference in training procedure and data sampling, and not originating from a weaker base ConvNet, since our frame baseline with a weaker ResNet-50 produces 72.1% mAP (vs. the 74.2% for ResNet-101). Next, we are interested in how our model performs after fine-tuning with the tracking loss, operating via RoI tracking on the correlation and track regression features (termed D (& T) loss in Table 1). The resulting performance for single-frame testing is 75.8% mAP. This 1.6% gain in accuracy shows that merely adding the tracking loss can aid the per-frame detection. A possible reason is that the correlation features propagate gradients back into the base ConvNet and therefore make the features more sensitive to important objects in the training data. We see significant gains for classes like panda, monkey, rabbit or snake which are likely to move a lot.

Video level methods. Next, we investigate the effect of multi-frame input during testing. In Table 1 we see that linking our detections to tubes based on our tracklets, D&T ($\tau = 1$), raises performance substantially to 79.8% mAP. Some class-AP scores can be boosted significantly (e.g. cattle by 9.6, dog by 5.5, cat by 6, fox by 7.9, horse by 5.3, lion by 9.4, motorcycle by 6.4 rabbit by 8.9, red panda by 6.3 and squirrel by 8.5 points AP). This gain is mostly for the following reason: if an object is captured in an unconventional pose, is distorted by motion blur, or appears at a small scale, the detector might fail; however, if its tube is linked to other potentially highly scoring detections of the same object, these failed detections can be recovered (even though we use a very simple re-weighting of detections across a tube). The only class that loses AP is whale (-2.6 points) and this has an obvious explanation: in most validation snippets the whales successively emerge and submerge from the water and our detection rescoring based on tubes would assign false positives when they submerge for a couple of frames.

When comparing our 79.8% mAP against the current state of the art, we make the following observations. The method in [18] achieves 47.5% by using a temporal convolutional network on top of the still image detector. An extended work [16] uses an encoder-decoder LSTM on top

Methods	airplane	antelope	bear	bicycle	bird	bus	car	cattle	dog	d. cat	elephant	fox	g. panda	hamster	horse	lion
TCN [18]	72.7	75.5	42.2	39.5	25.0	64.1	36.3	51.1	24.4	48.6	65.6	73.9	61.7	82.4	30.8	34.4
TPN+LSTM [16]	84.6	78.1	72.0	67.2	68.0	80.1	54.7	61.2	61.6	78.9	71.6	83.2	78.1	91.5	66.8	21.6
Winner ILSVRC'15 [17]	83.7	85.7	84.4	74.5	73.8	75.7	57.1	58.7	72.3	69.2	80.2	83.4	80.5	93.1	84.2	67.8
D (R-FCN)	87.4	79.4	84.5	67.0	72.1	84.6	54.6	72.9	70.9	77.3	76.7	89.7	77.6	88.5	74.8	57.9
D (& T loss)	89.4	80.4	83.8	70.0	71.8	82.6	56.8	71.0	71.8	76.6	79.3	89.9	83.3	91.9	76.8	57.3
D&T (= 1)	90.2	82.3	87.9	70.1	73.2	87.7	57.0	80.6	77.3	82.6	83.0	97.8	85.8	96.6	82.1	66.7
D&T (= 10)	89.1	79.8	87.5	68.8	72.9	86.1	55.7	78.6	76.4	83.4	82.9	97.0	85.0	96.0	82.2	66.0

Methods	lizard	monkey	motorcycle	rabbit	red panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra	mAP (%)
TCN [18]	54.2	1.6	61.0	36.6	19.7	55.0	38.9	2.6	42.8	54.6	66.1	69.2	26.5	68.6	47.5
TPN+LSTM [16]	74.4	36.6	76.3	51.4	70.6	64.2	61.2	42.3	84.8	78.1	77.2	61.5	66.9	88.5	68.4
Winner ILSVRC'15 [17]	80.3	54.8	80.6	63.7	85.7	60.5	72.9	52.7	89.7	81.3	73.7	69.5	33.5	90.2	73.8
Winner ILSVRC'16 [39]	(single model performance)														76.2
D (R-FCN)	76.8	50.1	80.2	61.3	79.5	51.9	69.0	57.4	90.2	83.3	81.4	68.7	68.4	90.9	74.2
D (& T loss)	79.0	54.1	80.3	65.3	85.3	56.9	74.1	59.9	91.3	84.9	81.9	68.3	68.9	90.9	75.8
D&T (= 1)	83.4	57.6	86.7	74.2	91.6	59.7	76.4	68.4	92.6	86.1	84.3	69.7	66.3	95.2	79.8
D&T (= 10)	83.1	57.9	79.8	72.7	90.0	59.4	75.6	65.4	90.5	85.6	83.3	68.3	66.5	93.2	78.6

Table 1. Performance comparison on the ImageNet VID validation set. The average precision (in %) for each class and the mean average precision over all classes is shown. corresponds to the temporal sampling stride.

of a Faster R-CNN object detector which works on proposals from a tubelet proposal network, and produces 68.4% mAP. The ILSVRC 2015 winner [17] combines two Faster R-CNN detectors, multi-scale training/testing, context suppression, high confidence tracking [38] and optical-flow-guided propagation to achieve 73.8%. And the winner from ILSVRC2016 [39] uses a cascaded R-FCN detector, context inference, cascade regression and a correlation tracker [25] to achieve 76.19% mAP validation performance with a single model (multi-scale testing and model ensembles boost their accuracy to 81.1%).

Online capabilities and runtime. The only component limiting online application is the tube rescoring (Sect. 4). We have evaluated an online version which performs only causal rescoring across the tracks. The performance for this method is 78.7% mAP, compared to the noncausal method (79.8% mAP). Since the correlation layer and track regressors are operating fully convolutional (no additional per-ROI computation is added except at the ROI-tracking layer), the extra runtime cost for testing a 1000x600 pixel image is 14ms (i.e. 141ms vs 127ms without correlation and ROI-tracking layers) on a Titan X GPU. The (unoptimized) tube linking (Sect. 4) takes on average 46ms per frame on a single CPU core).

Temporally strided testing. In a final experiment we look at larger temporal strides during testing, which has recently been found useful for the related task of video action recognition [6, 7]. Our D & T architecture is evaluated only at every t^{th} frame of an input sequence and tracklets have to link detections over larger temporal strides. The performance for a temporal stride of $= 10$ is 78.6% mAP which is 1.2% below the full-frame evaluation. We think that such a minor drop is remarkable as the duration for processing a video is now roughly reduced by a factor of 10.

A potential point of improvement is to extend the detector to operate over multiple frames of the sequence. We

found that such an extension did not have a clear beneficial effect on accuracy for short temporal windows (i.e. augmenting the detection scores at time t with the detector output at the tracked proposals in the adjacent frame at time $t + 1$ only raises the accuracy from 79.8 to **80.0%** mAP). Increasing this window to frames at $t \pm 1$ by bidirectional detection and tracking from the t^{th} frame did not lead to any gain. Interestingly, when testing with a temporal stride of $= 10$ and augmenting the detections from the current frame at time t with the detector output at the tracked proposals at $t + 10$ raises the accuracy from 78.6 to 79.2% mAP.

We conjecture that the insensitivity of the accuracy for short temporal windows originates from the high redundancy of the detection scores from the centre frames with the scores at tracked locations. The accuracy gain for larger temporal strides, however, suggests that more complementary information is integrated from the tracked objects; thus, a potentially promising direction for improvement is to detect and track over multiple temporally strided inputs.

6. Conclusion

We have presented a unified framework for simultaneous object detection and tracking in video. Our fully convolutional D&T architecture allows end-to-end training for detection and tracking in a joint formulation. In evaluation, our method achieves accuracy competitive with the winner of the last ImageNet challenge while being simple and efficient. We demonstrate clear mutual benefits of jointly performing the task of detection and tracking, a concept that can foster further research in video analysis.

Acknowledgments. This work was partly supported by the Austrian Science Fund (FWF P27076) and by EPSRC Programme Grant Seebibyte EP/M013774/1.

References

- [1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV VOT Workshop*, 2016. **2, 3**
- [2] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Proc. CVPR*, 2010. **3**
- [3] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. **1, 2, 3, 4, 5, 7**
- [4] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proc. ECCV*, 2016. **3**
- [5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. **5**
- [6] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. **8**
- [7] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016. **8**
- [8] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proc. CVPR*, 2015. **7**
- [9] R. B. Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. **1, 2, 3, 4**
- [10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014. **1, 2, 4**
- [11] G. Gkioxari and J. Malik. Finding action tubes. In *Proc. CVPR*, 2015. **2, 5, 6**
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. **1, 2, 3**
- [13] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 FPS with deep regression networks. In *Proc. ECCV*, 2016. **2, 3, 7**
- [14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE PAMI*, 37(3):583–596, 2015. **3**
- [15] A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *Proc. ECCV*, 2014. **2**
- [16] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *Proc. CVPR*, 2017. **2, 6, 7, 8**
- [17] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. *arXiv preprint*, 2016. **1, 2, 6, 7, 8**
- [18] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *Proc. CVPR*, 2016. **2, 6, 7, 8**
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. **1**
- [20] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised object discovery and tracking in video collections. In *Proc. CVPR*, 2015. **2**
- [21] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. **1**
- [22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, 2014. **1**
- [23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proc. ECCV*, 2016. **2**
- [24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015. **3**
- [25] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proc. ICCV*, 2015. **2, 3, 8**
- [26] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proc. CVPR*, 2016. **2**
- [27] X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *Proc. ECCV*, 2016. **2, 5, 6**
- [28] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Proc. CVPR*, 2012. **2**
- [29] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. YouTube-BoundingBoxes: A Large High-Precision Human-Annotated Data Set for Object Detection in Video. *ArXiv e-prints*, 2017. **3**
- [30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 2016. **2**
- [31] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE PAMI*, 2016. **1, 2, 3, 4, 6**
- [32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. **1, 6**
- [33] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *Proc. BMVC*, 2016. **2, 5, 6**
- [34] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proc. CVPR*, 2016. **7**
- [35] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. **2**
- [36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. **1**
- [37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. **1**
- [38] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proc. ICCV*, 2015. **1, 2, 8**
- [39] J. Yang, H. Shuai, Z. Yu, R. Fan, Q. Ma, Q. Liu, and J. Deng. ILSVRC2016 object detection from video: Team NUIST. <http://image-net.org/challenges/talks/2016/ImageNet%202016%20VID.pptx>, 2016. **8**
- [40] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proc. CVPR*, 2017. **2, 6, 7**