Towards High Performance Video Object Detection

Xizhou Zhu[1,2*] Jifeng Dai[2] Lu Yuan[2] Yichen Wei[2]

[1]University of Science and Technology of China [2]Microsoft Research

ezra0408@mail.ustc.edu.cn {jifdai,luyuan,yichenw}@microsoft.com

Abstract

There has been significant progresses for image object detection in recent years. Nevertheless, video object detection has received little attention, although it is more challenging and more important in practical scenarios.

Built upon the recent works [37, 36], this work proposes a unified approach based on the principle of multi-frame end-to-end learning of features and cross-frame motion. Our approach extends prior works with three new techniques and steadily pushes forward the performance envelope (speed-accuracy tradeoff), towards high performance video object detection.

1. Introduction

Recent years have witnessed significant progress in object detection [17] in still images. However, directly applying these detectors to videos faces new challenges. First, applying the deep networks on all video frames introduces unaffordable computational cost. Second, recognition accuracy suffers from deteriorated appearances in videos that are seldom observed in still images, such as motion blur, video defocus, rare poses, etc.

There has been few works on video object detection. The recent works [37, 36] suggest that principled multi-frame end-to-end learning is effective towards addressing above challenges. Specifically, data redundancy between consecutive frames is exploited in [37] to reduce the expensive feature computation on most frames and improve the speed. Temporal feature aggregation is performed in [36] to improve the feature quality and recognition accuracy. These works are the foundation of the ImageNet Video Object Detection Challenge 2017 winner [6].

The two works focus on different aspects and presents their own drawbacks. *Sparse feature propagation* (see Eq. (1)) is used in [37] to save expensive feature computation on most frames. Features on these frames are propagated from sparse key frames cheaply. The propagated features, however, are only approximated and error-prone, thus hurting the recognition accuracy. Multi-frame *dense feature aggregation* (see Eq. (2)) is performed in [36] to improve feature quality on all frames and detection accuracy as well. Nevertheless, it is much slower due to repeated motion estimation, feature propagation and aggregation.

The two works are complementary in nature. They also share the same principles: motion estimation module is built into the network architecture and end-to-end learning of all modules is performed over multiple frames.

Built on these progresses and principles, this work presents a unified approach that is faster, more accurate, and more flexible. Specifically, three new techniques are proposed. First, *sparsely recursive feature aggregation* is used to retain the feature quality from aggregation but as well reduce the computational cost by operating only on sparse key frames. This technique combines the merits of both works [37, 36] and performs better than both.

Second, *spatially-adaptive partial feature updating* is introduced to recompute features on non-key frames wherever propagated features have bad quality. The feature quality is learnt via a novel formulation in the end-to-end training. This technique further improves the recognition accuracy.

Last, *temporally-adaptive key frame scheduling* replaces the previous fixed key frame scheduling. It predicts the usage of a key frame accordingly to the predicted feature quality above. It makes the key frame usage more efficient.

The proposed techniques are unified with the prior works [37, 36] under a unified viewpoint. Comprehensive experiments show that the three techniques steadily pushes forward the performance (speed-accuracy trade-off) envelope, towards high performance video object detection. For example, we achieve 77.8% mAP score at speed of 15.22 frame per second. It establishes the new state-of-the-art.

2. From Image to Video Object Detection

Object detection in static images has achieved significant progress in recent years using deep CNN [17]. State-of-the-art detectors share the similar methodology and network

*This work is done when Xizhou Zhu is intern at Microsoft Research Asia

architecture, consisting of two *conceptual* steps.

First step extracts a set of convolutional feature maps F over the whole input image I via a fully convolutional backbone network [31, 33, 14, 32, 34, 16, 2, 15, 35]. The backbone network is usually pre-trained on the ImageNet classification task and fine-tuned later. In this work, it is called *feature network*, $\mathcal{N}_{\text{feat}}(I) = F$. It is usually deep and slow. Computing it on all video frames is unaffordable.

Second step generates detection result y upon the feature maps F, by performing region classification and bounding box regression over either sparse object proposals [10, 13, 9, 29, 4, 24, 12, 5] or dense sliding windows [26, 27, 28, 25], via a multi-branched sub-network. It is called *detection network* in this work, $\mathcal{N}_{\text{det}}(F) = y$. It is randomly initialized and jointly trained with $\mathcal{N}_{\text{feat}}$. It is usually shallow and fast.

2.1. Revisiting Two Baseline Methods on Video

Sparse Feature Propagation [37]. It introduces the concept of *key frame* for video object detection, for the first time. The motivation is that similar appearance among adjacent frames usually results in similar features. It is therefore unnecessary to compute features on all frames.

During inference, the expensive feature network $\mathcal{N}_{\text{feat}}$ is applied only on sparse key frames (*e.g.*, every 10^{th}). The feature maps on any non-key frame i are propagated from its preceding key frame k by per-pixel feature value warping and bilinear interpolation. The between frame pixel-wise motion is recorded in a two dimensional *motion field* $M_{i\to k}$[1]. The propagation from key frame k to frame i is denoted as

$$F_{k\to i} = \mathcal{W}(F_k, M_{i\to k}), \tag{1}$$

where $\mathcal{W}$ represents the feature warping function. Then the detection network $\mathcal{N}_{\text{det}}$ works on $F_{k\to i}$, the approximation to the real feature F_i, instead of computing F_i from $\mathcal{N}_{\text{feat}}$.

The motion field is estimated by a lightweight flow network, $\mathcal{N}_{\text{flow}}(I_k, I_i) = M_{i\to k}$ [7], which takes two frames I_k, I_i as input. End-to-end training of all modules, including $\mathcal{N}_{\text{flow}}$, greatly boosts the detection accuracy and makes up for the inaccuracy caused by feature approximation. Compared to the single frame detector, because the computation of $\mathcal{N}_{\text{flow}}$ and Eq. (1) is much cheaper (dozens, see Table 2 in [37]) than feature extraction in $\mathcal{N}_{\text{feat}}$, method in [37] is much faster (up to $10\times$) with small accuracy drop (up to a few mAP points) (see, Figure 3 in [37]).

Dense Feature Aggregation [36]. It introduces the concept of *temporal feature aggregation* for video object detection, for the first time. The motivation is that the deep features would be impaired by deteriorated appearance (*e.g.*, motion blur, occlusion) on certain frames, but could be improved by aggregation from nearby frames.

During inference, feature network $\mathcal{N}_{\text{feat}}$ is *densely* evaluated on all frames. For any frame i, the feature maps of all the frames within a temporal window $[i-r, i+r]$ ($r = 2 \sim 12$ frames) are firstly warped onto the frame i in the same way to [37] (see Eq. (1)), forming a set of feature maps $\{F_{k\to i} | k \in [i-r, i+r]\}$. Different from *sparse feature propagation* [37], the propagation occurs at every frame instead of key frame only. In other words, every frame is viewed as key frame.

The aggregated feature maps $\bar{F}_i$ at frame i is then obtained as the weighted average of all such feature maps,

$$\bar{F}_i(p) = \sum_{k\in[i-r,i+r]} W_{k\to i}(p) \cdot F_{k\to i}(p), \forall p, \tag{2}$$

where the weight $W_{k\to i}$ is adaptively computed as the similarity between the propagated feature maps $F_{k\to i}$ and the real feature maps F_i. Instead, the feature F is projected into an embedding feature F^e for similarity measure, and the projection can be implemented by a tiny fully convolutional network (see Section 3.4 in [36]).

$$W_{k\to i}(p) = \exp\left(\frac{F^e_{k\to i}(p) \cdot F^e_i(p)}{|F^e_{k\to i}(p)| \cdot |F^e_i(p)|}\right), \forall p. \tag{3}$$

Note that both Eq. (2) and (3) are in a position-wise manner, as indicated by enumerating the location p. The weight is normalized at every location p over nearby frames, $\sum_{k\in[i-r,i+r]} W_{k\to i}(p) = 1$.

Similarly as [37], all modules including the flow network and aggregation weight, etc., are jointly trained. Compared to the single frame detector, the aggregation in Eq. (2) greatly enhances the features and improves the detection accuracy (about 3 mAP points), especially for the fast moving objects (about 6 mAP points) (see Table 1 in [36]). However, runtime is about 3 times slower due to the repeated flow estimation and feature aggregation over dense consecutive frames.

3. High Performance Video Object Detection

The difference between the above two methods is apparent. [37] reduces feature computation by feature approximation, which decreases accuracy. [36] improves feature quality by adaptive aggregation, which increases computation. They are naturally complementary.

On the other hand, they are based on the same two principles: 1) *motion estimation* module is indispensable for effective *feature level* communication between frames; 2) *end-to-end learning over multiple frames* of all modules is crucial for detection accuracy, as repeatedly verified in [37, 36].

[1] Since the warping $\mathcal{W}$ from frame k to i adopts backward warping, we directly estimate and use backward motion field $M_{i\to k}$ for convenience.

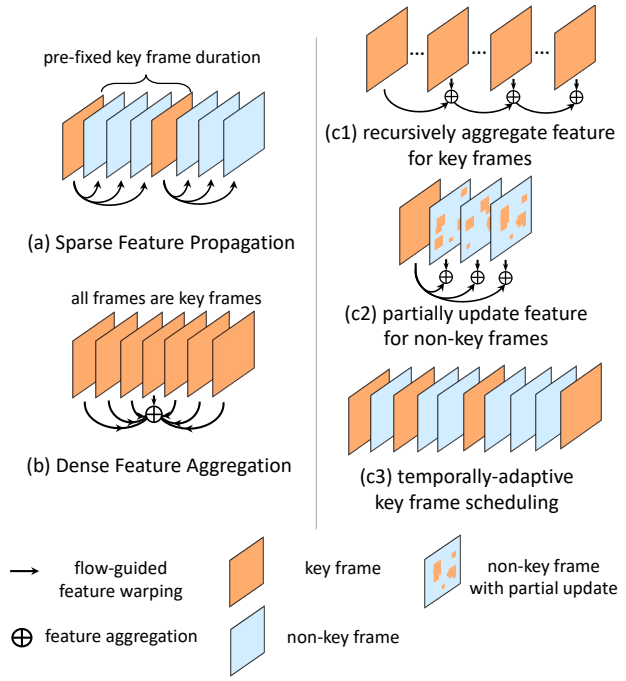

Figure 1. Illustration of the two baseline methods in [37, 36] and three new techniques presented in Section 3.

Based on the same underlying principles, this paper presents a common framework for high performance video object detection, as summarized in Section 3.4. It proposes three novel techniques. The first (Section 3.1) exploits the complementary property and integrates the methods in [37, 36]. It is both accurate and fast. The second (Section 3.2) extends the idea of adaptive feature computation from temporal domain to spatial domain, resulting in spatially-adaptive feature computation that is more effective. The third (Section 3.3) proposes adaptive key frame scheduling that further improves the efficiency of feature computation.

These techniques are simple and intuitive. They naturally extend the previous works. Each one is built upon the previous one(s) and steadily pushes forward the performance (runtime-accuracy trade off) envelope, as verified by extensive experiments in Section 5.

The two baseline methods and the three new techniques are illustrated in Figure 1.

3.1. Sparsely Recursive Feature Aggregation

Although Dense Feature Aggregation [36] achieves significant improvement on detection accuracy, it is quite slow. On one hand, it densely evaluates feature network $\mathcal{N}_{\text{feat}}$ on all frames, however that is unnecessary due to the similar appearance among adjacent frames. One the other hand, feature aggregation is performed on multiple feature maps and thus multiple flow fields are needed to be estimated, which largely slow down the detector.

Here we propose *Sparsely Recursive Feature Aggregation*, which both evaluates feature network $\mathcal{N}_{\text{feat}}$ and applies recursive feature aggregation only on sparse key frames. Given two succeeding key frames k and k', the aggregated feature at frame k' is computed by

$$\bar{F}_{k'} = W_{k\to k'} \odot \bar{F}_{k\to k'} + W_{k'\to k'} \odot F_{k'}, \qquad (4)$$

where $\bar{F}_{k\to k'} = \mathcal{W}(\bar{F}_k, M_{k'\to k})$, and $\odot$ denotes element-wise multiplication. The weight is correspondingly normalized by $W_{k\to k'}(p) + W_{k'\to k'}(p) = 1$ at every location p.

This is a recursive version of Eq. (2), and the aggregation only happens at sparse key frames. In principle, the aggregated key frame feature $\bar{F}_k$ aggregates the rich information from all history key frames, and is then propagated to the next key frame k' for aggregating the original feature $F_{k'}$.

3.2. Spatially-adaptive Partial Feature Updating

Although Sparse Feature Propagation [37] achieves remarkable speedup by approximating the real feature F_i, the propagated feature map $F_{k\to i}$ is error-prone due to some parts with changing appearance among adjacent frames.

For non-key frames, we want to use the idea of feature propagation for efficient computation, however Eq. (1) is subject to the quality of propagation. To quantify whether the propagated feature $F_{k\to i}$ is a good approximation of F_i, a feature temporal consistency $Q_{k\to i}$ is introduced. We add a sibling branch on the flow network $\mathcal{N}_{\text{flow}}$ for predicting $Q_{k\to i}$, together with motion field $M_{i\to k}$, as

$$\{M_{i\to k}, Q_{k\to i}\} = \mathcal{N}_{\text{flow}}(I_k, I_i). \qquad (5)$$

If $Q_{k\to i}(p) \le \tau$, the propagated feature $F_{k\to i}(p)$ is inconsistent with the real feature $F_i(p)$. That is to say, $F_{k\to i}(p)$ is a bad approximation, which suggests updating with real feature $F_i(p)$.

We develop an economic way to partially update the features on non-key frames layer-by-layer. Supposing $\mathcal{N}_{\text{feat}}$ has N layers, features at frame i are updated by

$$\hat{F}_i^{(n)} = U_{k\to i} \odot \mathcal{N}_{\text{feat}}^{(n)}(\hat{F}_i^{(n-1)}) + (1 - U_{k\to i}) \odot F_{k\to i}^{(n)}, \quad (6)$$

where $\hat{F}_i^{(0)} = I_i$, $\hat{F}_i^{(n)}$ and $F_{k\to i}^{(n)}$ are the partially updated features and propagated features at layer n, respectively, and $\mathcal{N}_{\text{feat}}^{(n)}$ is the network operation (e.g. convolution) on layer n. Because the resolution of feature maps in different layers is different, we use nearest neighbor interpolation to update the mask $U_{k\to i}$ to have the same spatial resolution as $F_{k\to i}^{(n)}$. Thus, $\hat{F}_i = \hat{F}_i^{(N)}$ is the final partially updated feature.

Following [3], we use a straight-through estimator for the gradient $\frac{\partial U_{k\to i}(p)}{\partial Q_{k\to i}(p)} = -1$, if $|Q_{k\to i}(p) - \tau| \le 1$, $\frac{\partial U_{k\to i}(p)}{\partial Q_{k\to i}(p)} = 0$, otherwise. Thus it is fully differentiable.

We can regard $Q_{k\to i}(p) - \tau$ as a new valuable for the estimation of $Q_{k\to i}(p)$, since τ can be viewed as the bias of $Q_{k\to i}(p)$, which takes no effect to the estimate $Q_{k\to i}(p)$. For simplicity, we directly set $\tau = 0$ in this paper.

To further improve the feature quality for non-key frames, feature aggregation is also utilized as similar as Eq. 4:

$$\bar{F}_i = W_{k\to i} \odot \bar{F}_{k\to i} + W_{i\to i} \odot \hat{F}_i, \tag{7}$$

where the weight is normalized by $W_{k\to i}(p) + W_{i\to i}(p) = 1$ at every location p.

3.3. Temporally-adaptive Key Frame Scheduling

Evaluating feature network $\mathcal{N}_{\text{feat}}$ only on sparse key frames is crucial for high speed. A naive key frame scheduling policy picks a key frame at a pre-fixed rate, *e.g.*, every l frames[37]. A better key frame scheduling policy should be adaptive to the varying dynamics in the temporal domain. It can be designed based on the feature consistency indicator $Q_{k\to i}$:

$$key = is\_key(Q_{k\to i}). \tag{8}$$

Here we designed a simple heuristic is_key function:

$$is\_key(Q_{k\to i}) = [\frac{1}{N_p}\sum_p \mathbf{1}(Q_{k\to i}(p) \le \tau)] > \gamma \tag{9}$$

where $\mathbf{1}(\cdot)$ is the indicator function, N_p is the number of all locations p. For any location p, $Q_{k\to i}(p) \le \tau$ indicates changing appearance or large motion which will lead to bad feature propagation quality, if the area to recompute $(Q_{k\to i}(p) \le \tau)$ is larger than a portion γ of all the pixels, the frame is marked as key. Figure. 2 shows an example of the area satisfied $Q_{k\to i}(p) \le \tau$ varying through time. Three orange points are examples of key frame selected by our is_key function, their appearance are clearly different. Two blue points are examples of non-key frame, their appearance indeed changed slightly compared with the preceding key frame.

To explore the potential and upper bound of key frame scheduling, we designed an oracle scheduling policy that exploits the ground-truth information. The experiment is performed with our proposed method, except for key frame scheduling policy. Given any frame i, both the detection results of picking frame i as a key frame or non-key frame are computed, and the two mAP scores are also computed using ground truth. If picking it as a key frame results a higher mAP score, frame i is marked as key.

This oracle scheduling achieves a significantly better result, *i.e.*, 80.9% mAP score at 22.8 fps runtime speed. This indicates the importance of key frame scheduling and suggests that it is an important future working direction.

3.4. A Unified Viewpoint

All methods are summarized under a unified viewpoint.

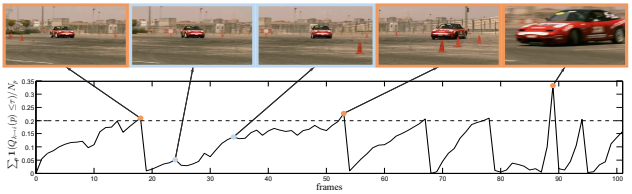

Figure 2. The area satisfying $Q_{k\to i}(p) \le \tau$ on video frames, where the key frame scheduling in Eq. (9) is applied ($\gamma = 0.2$).

To efficiently compute feature maps, *Spatially-adaptive Partial Feature Updating* (see Section 3.2) is utilized. Although Eq. (6) is only defined for non-key frames, it can be generalized to all frames. Given a frame i and its preceding key frame k, Eq. (6) is utilized, and summarized as

$$\hat{F}_i = PartialUpdate(I_i, F_k, M_{i\to k}, Q_{k\to i}). \tag{10}$$

For key frames, $Q_{k\to i} = -\infty$, propagated features $F_{k\to i}$ are always bad approximation of real features F_i, we should recompute feature $\hat{F}_i = \mathcal{N}_{\text{feat}}(I_i)$. For non-key frames, when $Q_{k\to i} = +\infty$, propagated features $F_{k\to i}$ are always good approximation of true features F_i, we directly use the propagated feature from the preceding key frame $\hat{F}_i = F_{k\to i}$.

To enhance the partially updated feature maps $\hat{F}_i$, feature aggregation is utilized. Although Eq. (4) only defined *Sparsely Recursive Feature Aggregation* for key frames, and Eq. (7) only defined feature aggregation for partially updated non-key frames. Eq. (4) can be regarded as a degenerated version of Eq. (7), supposing $i = k', \hat{F}_i = F_{k'}$. Thus feature aggregation is always performed as Eq. (7), and summarized as

$$\bar{F}_i = \mathcal{G}(\bar{F}_k, \hat{F}_i, M_{i\to k}), \tag{11}$$

To further improves the efficiency of feature computation, *Temporally-adaptive Key Frame Scheduling* (see Section 3.3) is also utilized.

Inference Algorithm 1 summarizes the unified inference algorithm. Different settings result in different degenerated versions, and Table 1 presents all methods from the unified viewpoint. Our method (c3) integrates all the techniques and works best.

If *Temporally-adaptive Key Frame Scheduling* is adopted, and both options do_aggr and $do\_spatial$ are set as $true$, then it is the online version of our proposed method. Utilizing a naive key frame scheduling, *i.e.*, pick a key every l frame, and both options do_aggr and $do\_spatial$ set as $false$, the algorithm degenerates to Sparse Feature Propagation [37] when $l > 1$, and the per-frame baseline when $l = 1$. The algorithm would degenerate to Dense Feature Aggregation [36] under condition that $do\_aggr = true, key = true$, and $do\_spatial = false$

method	$is\_key(\cdot,\cdot)$	key frame usage	do_aggr	$do\_spatial$	accuracy↔speed
per-frame baseline (*)	all frames	N.A	$false$	$false$	none
Sparse Feature Propagation [37]	every l frames	sparse, 1	$false$	$false$	l
Dense Feature Aggregation [36]	all frames	dense, ≥ 1	$true$	$false$	#key frames
our method (c1)	every l frames	sparse, recursive	$true$	$false$	l
our method (c2)	every l frames	sparse, recursive	$true$	$true$	l, λ
our method (c3)	temporally-adaptive	sparse, recursive	$true$	$true$	λ, γ

Table 1. All methods under a unified viewpoint.

for all the frames (*i.e.*, $l = 1$), and the unified feature aggregation on Line 20 is replaced by the dense aggregation in Eq. (2). Among all options in Table 1, a sparse key frame scheduling is crucial of fast inference, $do\_aggr = true$ and $do\_spatial = true$ is crucial for high accuracy.

Training All the modules in the entire architecture, including $\mathcal{N}_{\text{flow}}$, $\mathcal{N}_{\text{feat}}$ and $\mathcal{N}_{\text{det}}$, can be jointly trained. Due to memory limitation, in SGD, two nearby frames are randomly sampled in each mini-batch. The preceding frame is set as key, and the succeeding one is set as non-key, which are denoted as I_k and I_i, respectively.

In the forward pass, feature network $\mathcal{N}_{\text{feat}}$ is applied on I_k to obtain the feature maps F_k. Next, a flow network $\mathcal{N}_{\text{flow}}$ runs on the frames I_i, I_k to estimate the 2D flow field $M_{i\to k}$ and the feature consistency indicator $Q_{k\to i}$. Partially updated feature maps $\hat{F}_i$ is computed through Eq. (6), and then the aggregated current feature maps $\bar{F}_i$ is calculated through Eq. (7). Finally, the detection sub-network $\mathcal{N}_{\text{det}}$ is applied on $\bar{F}_i$ to produce the result y_i. Loss function is defined as,

$$L = L_{det}(y_i) + \lambda \sum_p U_{k\to i}(p), \tag{12}$$

where the updating mask $U_{k\to i}$ is defined in Eq. (6). The first term is the loss function for object detection, following the multi-task loss in Faster R-CNN [29], which consists of classification loss and bounding box regression loss together. Typically, features on the current frame have better detection accuracy. Thus, the first term encourages $U_{k\to i}$ to be $\mathbf{1}$. The second term encourages $U_{k\to i}$ to be $\mathbf{0}$ and thus enforces a constraint on the size of areas to be recomputed. The parameter λ controls this speed-accuracy trade off. Using a large λ achieves high speed. By default, $\lambda = 2$ and only less than 5% locations need to be updated.

During training, by default $U_{k\to i}$ is predicted by $\mathcal{N}_{\text{flow}}$. However, to encourage good performance for both cases of propagating feature and recomputing feature from scratch, we also randomly enforce $U_{k\to i} = \mathbf{0}$ and $U_{k\to i} = \mathbf{1}$ respectively, i.e. given a pair of frames, $U_{k\to i}$ is randomly set as $\mathcal{N}_{\text{flow}}$ prediction, $\mathbf{0}$ and $\mathbf{1}$ with equal probability $1/3$. For methods without using partial feature updating, training does not change and $U_{k\to i}$ is simply ignored during inference. Thus, a unified single training strategy is used.

3.5. Network Architecture

We introduce the incarnation of different sub-networks in our proposed model.

Flow network. We use FlowNet [7] ("simple" version). It is pre-trained on the Flying Chairs dataset [7]. It is applied on images of half resolution and has an output stride of 4. As the feature network has an output stride of 16 (see below), the flow field is downscaled by half to match the resolution of the feature maps. An additional randomly initialized 3x3 convolution is added to predict the feature propagability indicator, which shares feature with the last convolution of the FlowNet.

Feature network. We adopt the state-of-the-art ResNet-101 [14] as the feature network. The ResNet-101 model is pre-trained on ImageNet classification. We slightly modify the nature of ResNet-101 for object detection. We remove the ending average pooling and the fc layer, and retain the convolution layers. To increase the feature resolution, following the practice in [1, 4], the effective stride of the last block is changed from 32 to 16. Specially, at the beginning of the last block ("conv5" for both ResNet-101), the stride is changed from 2 to 1. To retain the receptive field size, the dilation of the convolutional layers (with kernel size > 1) in the last block is set as 2. Finally, a randomly initialized 3×3 convolution is applied on top to reduce the feature dimension to 1024.

Detection network. We use state-of-the-art R-FCN [4] and follow the design in [37]. On top of the 1024-d feature maps, the RPN sub-network and the R-FCN sub-network are applied, which connect to the first 512-d and the last 512-d features respectively. 9 anchors (3 scales and 3 aspect ratios) are utilized in RPN, and 300 proposals are produced on each image. The position-sensitive score maps in R-FCN are of 7×7 groups.

4. Related Work

Speed/accuracy trade-offs in object detection. As summarized in [17], speed/accuracy trade-off of modern detection systems can be achieved by using different feature networks [31, 33, 14, 32, 34, 16, 2, 15, 35] and detection networks [10, 13, 9, 29, 4, 24, 12, 5, 26, 27, 28, 25], or varying some critical parameters such as image resolution,

Algorithm 1 The unified flow-based inference algorithm for video object detection.

input: video frames $\{I_i\}$
$k = 0$ ▷ initialize key frame
$F_0 = \mathcal{N}_{\text{feat}}(I_0)$
$y_0 = \mathcal{N}_{\text{det}}(F_0)$
if do_aggr **then**
  $\bar{F}_0 = F_0$
end if
for $i = 1$ **to** ∞ **do**
  $\{M_{i\to k}, Q_{k\to i}\} = \mathcal{N}_{\text{flow}}(I_k, I_i)$ ▷ evaluate flow network
  $key = is\_key(Q_{k\to i})$ ▷ key frame scheduling
  if key **then**
    $Q_{k\to i} = -\infty$ ▷ need computing feature from scratch
  else if $do\_spatial$ **then**
    $Q_{k\to i}$ unchanged ▷ need partially updating
  else
    $Q_{k\to i} = +\infty$ ▷ suppose always good quality, propagate
  end if
  $\hat{F}_i = PartialUpdate(I_i, F_k, M_{i\to k}, Q_{k\to i})$ ▷ partially update
  if do_aggr **then**
    $\bar{F}_i = \mathcal{G}(\bar{F}_k, \hat{F}_i, M_{i\to k})$ ▷ recursively aggregate
    $y_i = \mathcal{N}_{\text{det}}(\bar{F}_i)$
  else
    $y_i = \mathcal{N}_{\text{det}}(\hat{F}_i)$
  end if
  if key **then** ▷ update the most recent key frame
    $k = i$
  end if
end for
output: detection results $\{y_i\}$

box proposal number. PVANET [22] and YOLO [27] even design specific feature networks for fast object detection. By applying several techniques (*e.g.* batch normalization, high resolution classifier, fine-grained features and multi-scale training), YOLO9000 [28] achieves higher accuracy meanwhile keep the high speed.

Since our proposed method only considers how to compute higher quality feature faster by using temporal information, and is not designed for any specific feature networks and detection networks, such techniques are also suitable for our proposed method.

Video object detection. Existing object detection methods incorporating temporal information in video can be separated into box-level methods [21, 20, 11, 23, 19, 8] and feature-level methods [37, 36] (both are flow-based methods and introduced in Section 2.1).

Box-level methods usually focus on how to improve detection accuracy considering temporary consistency within a tracklet. T-CNN [20, 21] first propagates predicted bounding boxes to neighboring frames according to pre-computed optical flows, and then generates tubelets by applying tracking algorithms. Boxes along each tubelet will be re-scored based on the tubelet classification result. Seq-NMS [11] constructs sequences along nearby high-confidence bounding boxes from consecutive frames. Boxes of the sequence are re-scored to the average confidence, other boxes close to this sequence are suppressed. MCMOT [23] formulates the post-processing as a multi-object tracking problem, and finally tracking confidence are used to re-score detection confidence. TPN [19] first generates tubelet proposals across multiple frames (≤ 20 frames) instead of bounding box proposals in a single frame, and then each tubelet proposal is classified into different classes by a LSTM based classifier. D&T [8] simultaneously outputs detection boxes and regression based tracking boxes with a single convolutional neural networks, and detection boxes are linked and re-scored based on tracking boxes.

Feature-level methods usually use optical flow to get pixel-to-pixel correspondence among nearby frames. Although feature-level methods are more principle and can further incorporate with box-level methods, they suffer from inaccurate optical flow. Still ImageNet VID 2017 winner is powered by feature-level methods DFF [37] and FGFA [36]. Our proposed method is also a feature-level method, which introduces *Spatially-adaptive Partial Feature Updating* to fix the inaccurate feature propagation caused by inaccurate optical flow.

5. Experiments

ImageNet VID dataset [30] is a prevalent large-scale benchmark for video object detection. Following the protocols in [20, 23], model training and evaluation are performed on the 3,862 video snippets from the training set and the 555 snippets from the validation set, respectively. The snippets are fully annotated, and are at frame rates of 25 or 30 fps in general. There are 30 object categories, which are a subset of the categories in the ImageNet DET dataset.

During training, following [20, 23], both the ImageNet VID training set and the ImageNet DET training set (only the same 30 categories as in ImageNet VID) are utilized. SGD training is performed. Each mini-batch samples one image from either ImageNet VID or ImageNet DET datasets, at $1:1$ ratio. 120K iterations are performed on 4 GPUs, with each GPU holding one mini-batch. The learning rates are 10^{-3} and 10^{-4} in the first 80K and in the last 40K iterations, respectively. In both training and inference, the images are resized to a shorter side of 600 pixels for the image recognition network, and a shorter side of 300 pixels for the flow network. Experiments are performed on a workstation with Intel E5-2670 v2 CPU 2.5GHz and Nvidia K40 GPU.

5.1. Evaluation under a Unified Viewpoint

Overall comparison results are shown in Figure 3.

Sparse Feature Propagation [37] is a degenerated version in Algorithm 1 (see Table 1). By varying key frame duration l from 1 to 10, it can achieve $5\times$ speedup with moderate accuracy loss (within 1%).

Similarly, for Dense Feature Aggregation [36], by varying the temporal window to be aggregated from ± 1 to ± 10 frames, it improves mAP score by 2.9% but is $3\times$ slower than per-frame baseline.

For our method (c1), key frames are picked once every l frames ($l = 1 \sim 10$ frames). Compared with Sparse Feature Propagation [37], the only difference is do_aggr set as $true$ instead of $false$, which leads to almost 1% improvement in mAP score with the same speedup. It recursively aggregates feature maps on sparse key frames, and the aggregated feature maps are propagated to non-key frames (see Eq. (4)). Compared with Dense Feature Aggregation [36], recursive aggregation is performed only on sparse key frames instead of dense feature aggregation performed on every frames, which leads to $10\times$ speedup with 2% accuracy loss. Compared with per-frame baseline, it achieves 1% higher accuracy and $3\times$ faster speed.

Our method (c2) extends our method (c1) by setting $do\_spatial$ as $true$ instead of $false$. It can further utilize rich appearance information from nearby frames with negligible computation burden. Compared with Sparse Feature Propagation [37], it improves mAP score with almost 2% and keeps the same high speed. Compared with Dense Feature Aggregation [36], it can speed up $9\times$ with 1% accuracy loss. Compared with per-frame baseline, this version results 1.8% higher accuracy with $3\times$ speedup and 1.4% higher accuracy with $4\times$ speedup.

Our method (c3) further extends our method (c2) by utilizing a temporally-adaptive key frame scheduling instead of a pre-fixed key frame duration. γ in Eq. (9) is fixed as 0.2. Compared with our method (c2), it further improves detection accuracy with $0.5\% \sim 1\%$ when high runtime speed is demanded. Compared with Sparse Feature Propagation [37], it improves mAP score with nearly 2% at all runtime speed. Compared with per-frame baseline, this version results 1% higher accuracy with $4.75\times$ speedup.

5.2. Ablation Study

We conduct ablation study for three different options of our method. The detailed setting is shown in Table 1. All of three options use sparsely recursive feature aggregation for key frame, and then propagate the aggregated features to non-key frames, *i.e.*, $do\_aggr = true$. The difference among them is key-frame scheduling and whether partial feature updating is used or not.

Our method (c1) We evaluate the effect of recursive feature aggregation compared with non-recursive aggregation (*i.e.*, dense aggregation) on sparse key frames. Here, we use several variant numbers of key frames for non-recursive aggregation. Results are shown in Figure 4. For non-recursive aggregation methods, aggregating more key frames is better when runtime speed is slow. Moreover, when aggregating more than 2 key frames, accuracy descends quickly. It is caused by feature inconsistency from propagated key frames with large key frame duration l, which is on the demand for high runtime speed. Recursive aggregation can solve this problem well by only considering two key frames in aggregation. More important, the aggregated feature theoretically contains all historical information of previous key frames. So the aggregation no longer needs more key frames (larger than 2 frames). As we can see, recursive aggregation surpasses the non-recursive aggregation at almost all runtime speed.

Our method (c2) We evaluate the effect of partially updating coefficient λ and key frame duration l, which actually controls the speed-accuracy trade-off. Figure 5 shows the results with varying λ and fixed l. Key frame duration $l = 10$ achieves the best speed-accuracy trade-off. Small l leads to redundancy between two consecutive key frames, which is not useful for recursive aggregation, thus results in a little accuracy loss. Large l leads to highly diverse feature response between two consecutive key frames, which is also not helpful. Figure 6 shows the results with varying l and fixed λ. Partially updating coefficient $\lambda = 2.0$ achieves the best speed-accuracy trade-off. Small λ implies very large recomputed area, and always gives low runtime speed regardless of key frame duration. High λ implies very small recomputed area, which does not fully exploit the strength of partially updating.

Our method (c3) We compare our *Temporally-adaptive Key Frame Scheduling* with different γ (see Eq. (9)), the results are showed in Figure 7. Different γs result almost the same performance when runtime speed is slow. $\gamma = 0.2$ results best speed-accuracy trade off when high runtime speed is demanded. The oracle key frame scheduling policy (described in Section. 3.3) achieves an incredibly better results.

Different flow networks We also evaluated different flow networks (including FlowNetS, FlowNetC and FlowNet2 [18]) for our proposed method. Results are showed in Figure. 8. FlowNetS results best speed-accuracy trade-off, this is because fast inference of flow network is the key to speedup in our proposed method. With joint training, FlowNetS can achieve significantly better results, which is consistent with [37, 36].

Deformable R-FCN [5] We further replace the detection system with Deformable R-FCN, which is slightly slower than the original R-FCN but much more accurate. Results are showed in Figure. 9. Our proposed method works well, and achieves 77.8% mAP score at 15.2 fps runtime speed, better than ImageNet VID 2017 winner (76.8% mAP score at 15.4 fps runtime speed [6]).

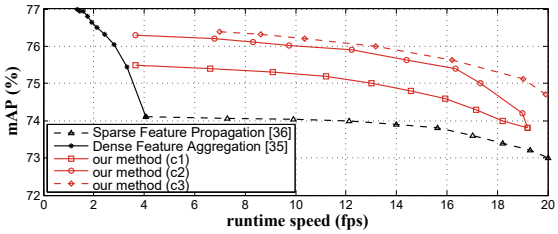

Figure 3. Speed-accuracy trade-off curves for methods in Table 1.

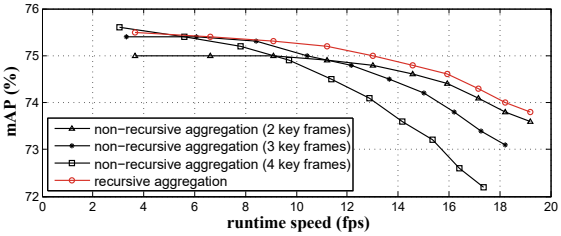

Figure 4. Speed-accuracy trade-off curves for our method (c1) and its non-recursive aggregation variants.

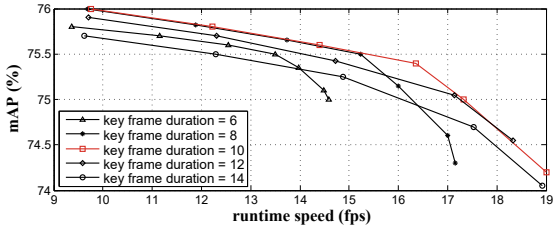

Figure 5. Speed-accuracy trade-off curves for our method (c2), and each curve shares a fixed key frame duration l.

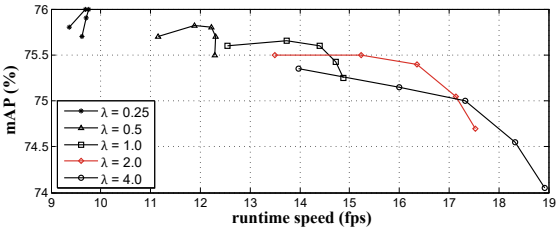

Figure 6. Speed-accuracy trade-off curves for our method (c2), and each curve shares a fixed partially updating coefficient λ.

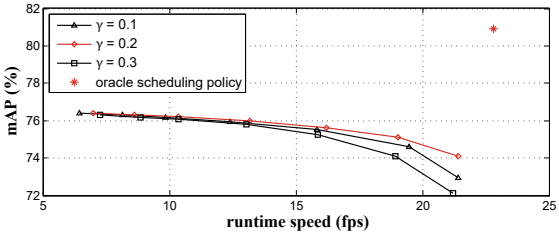

Figure 7. Speed-accuracy trade-off curves for our method (c3) with different γ.

5.3. Comparison with State-of-the-art Methods

We further compared with several state-of-the-art methods & systems for object detection from video, with reported results on ImageNet VID validation. It is worth mentioning that different recognition networks, object detectors, and post processing techniques are utilized in different approaches. Thus it is hard to draw a fair comparison.

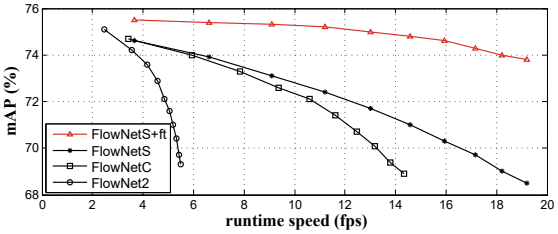

Figure 8. Speed-accuracy trade-off curves for our method (c1) with different flow networks. 'FlowNetS+ft' stands for FlowNetS jointly trained within our proposed method. Other flow networks are used without joint training.

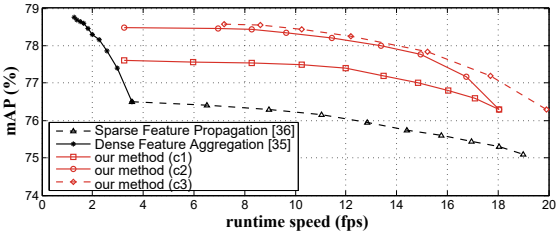

Figure 9. Speed-accuracy trade-off curves for all methods in Table 1 combined with Deformable R-FCN.

method	feature network	mAP (%)	runtime (fps) (TitanX/K40)
Ours	ResNet-101+DCN	**78.6**	13.0 / 8.6
		77.8	22.9 / 15.2
TPN [19]	GoogLeNet	68.4	2.1 / -
D&T [8]	ResNet-101	75.8	7.8 / -
ImageNet VID 2017 winner [6]	ResNet-101	76.8	- / 15.4

Table 2. Comparison with state-of-the-art methods.

Table 2 presents the results. For our method, we reported results by picking two operational points on curve "our method (c3)" from Figure 9. The mAP score is 78.6% at a runtime of 13.0 / 8.6 fps on Titan X / K40. The mAP score slightly decrease to 77.8% at a faster runtime of 22.9 / 15.2 fps on Titan X / K40. As a comparison, TPN [19] gets an mAP score of 68.4% at a runtime of 2.1 fps on Titan X. In the latest paper of D&T [8], an mAP score of 75.8% is obtained at a runtime of 7.8 fps on Titan X. Sequence NMS [11] can be applied to D&T to further improve the performance, which can also be applied in our approach. We also compared with the winning entry [6] of ImageNet VID challenge 2017, which is also based on sparse feature propagation [37] and dense feature aggregation [36]. It gets an mAP score of 76.8% at a runtime of 15.4 fps on Titan X. It is heavily-engineered and the implementation details are unreported. Our method is more principled, and achieves better performance in terms of both accuracy and speed.

References

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 5

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2, 5

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016. 3

[4] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 2, 5

[5] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 2, 5, 7

[6] J. Deng, Y. Zhou, B. Yu, Z. Chen, S. Zafeiriou, and D. Tao. Speed/accuracytradeoffs forobjectdetection-fromvideo. http://image-net.org/challenges/talks_2017/Imagenet2017VID.pdf, 2017. 1, 7, 8

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2, 5

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *ICCV*, 2017. 6, 8

[9] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2, 5

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2, 5

[11] W. Han, P. Khorrami, T. Le Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016. 6, 8

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 2, 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2, 5

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 5

[16] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 2, 5

[17] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 1, 5

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 7

[19] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017. 6, 8

[20] 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 arxiv:1604.02532*, 2016. 6

[21] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 6

[22] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, and M. Park. Pvanet: Deep but lightweight neural networks for real-time object detection. *arXiv preprint arXiv:1608.08021*, 2016. 6

[23] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee. Multi-class multi-object tracking using changing point detection. In *ECCV*, 2016. 6

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 5

[25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017. 2, 5

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2, 5

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2, 5, 6

[28] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 2, 5, 6

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 5

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. In *IJCV*, 2015. 6

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 5

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 2, 5

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2, 5

[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 5

[35] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 2, 5

[36] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[37] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *CVPR*, 2017. 1, 2, 3, 4, 5, 6, 7, 8