Progressive Sparse Local Attention for Video Object Detection

Chaoxu Guo[1,2] Bin Fan[1*] Jie Gu[1] Qian Zhang[3] Shiming Xiang[1,2]
Véronique Prinet[1] Chunhong Pan[1]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Horizon Robotics
{*chaoxu.guo,bfan,smxiang,prinet,chpan*}*@nlpr.ia.ac.cn*, {*qian01.zhang*}*@horizon.ai*

Abstract

Transferring image-based object detectors to the domain of videos remains a challenging problem. Previous efforts mostly exploit optical flow to propagate features across frames, aiming to achieve a good trade-off between accuracy and efficiency. However, introducing an extra model to estimate optical flow can significantly increase the overall model size. The gap between optical flow and high-level features can also hinder it from establishing spatial correspondence accurately. Instead of relying on optical flow, this paper proposes a novel module called Progressive Sparse Local Attention (PSLA), which establishes the spatial correspondence between features across frames in a local region with progressively sparser stride and uses the correspondence to propagate features. Based on PSLA, Recursive Feature Updating (RFU) and Dense Feature Transforming (DenseFT) are proposed to model temporal appearance and enrich feature representation respectively in a novel video object detection framework. Experiments on ImageNet VID show that our method achieves the best accuracy compared to existing methods with smaller model size and acceptable runtime speed.

1. Introduction

Object detection is a fundamental problem in computer vision and serves as a core technique in many practical applications, *e.g.* robotics, autonomous driving and human behavior analysis. With the development of convolutional neural networks (CNNs), remarkable successes have been achieved on detecting objects from images [7, 12, 13, 16, 17, 26, 28, 29, 31]. However, applying those techniques on a frame-by-frame basis to a video is often unsatisfactory due to the deteriorated appearance caused by issues such as motion blur, out-of-focus camera, and rare poses frequently encountered in videos. The temporal information encoded inherently in videos has been used to improve the performance of video object detection, as it provides rich cues about the motion in videos that are absent from still images.

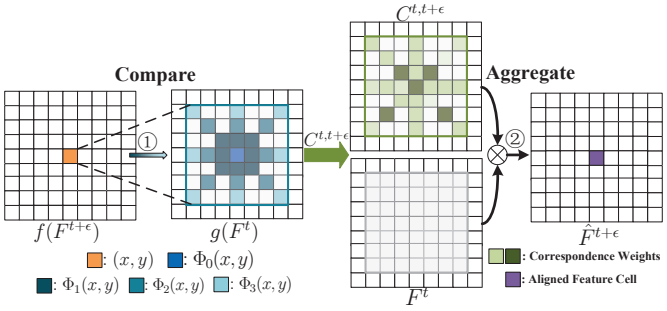

Figure 1. Illustration of Progressive Sparse Local Attention (PSLA). The goal of PSLA is to align feature map F_t with $F^{t+\epsilon}$ in an attention way, which is formulated into two steps: the first step ① is that each feature cell in embedded feature map $f(F^{t+\epsilon})$ compares to the surrounding cells in the embedded feature map $g(F^t)$, in a progressive sparser stride from center to outside. Regions with different colors in $g(F^t)$ represent regions in different strides, which are illustrated in equation 2 and 3. The resulting feature affinities are used to compute correspondence weights $C^{t,t+\epsilon}$, which capture the spatial correspondence between features. The second step ② is that the chosen feature cells in F^t are aggregated with the corresponding weights to generate a feature cell in $\hat{F}^{t+\epsilon}$, which is the aligned feature map from F^t.

Existing methods that leverage temporal information for object detection from videos mainly fall into two categories. The first one relies on dedicated post-processing [15, 20, 21, 23]. These methods firstly run an image-based detector on single frames and then integrate the per-frame results by box-level post-processing, which usually requires an extra object tracker or off-the-shelf optical flow to estimate the motion field and associate the bounding boxes. Modeling temporal coherence in this way is sub-optimal

*Bin Fan is the corresponding author

since detectors do not benefit from the temporal information in the phase of training.

Another category of methods [4, 10, 35, 37, 41, 42, 43] exploits the temporal information in videos when training the detectors. They either pursue a trade-off between accuracy and complexity or seek to improve performance at the expense of runtime. Among these methods, the optical flow is widely used to propagate the high-level features across frames. Extra optical flow models, *e.g.* FlowNet [8], have to be utilized to enable the end-to-end training and achieve better performance. However, adding an optical flow model has several drawbacks. First, the extra model significantly increases the overall model size of detectors (*e.g.*, a typical detector of ResNet101+RFCN has 59.6M parameters and it has to add additional 37M parameters when using FlowNet.), which makes it harder to be deployed on mobile devices. Second, optical flow only establishes local pixel correspondences between two images. Directly transferring the flow field to high-level features may introduce artifacts because it ignores the transformation that happens from layer to layer in the network. Finally, a shift of one pixel in high-level feature maps may correspond to up to tens of pixels in the image. It is very challenging for optical flow to capture such a large displacement.

Our work belongs to the second category. To address the limitations above, we propose a novel module, Progressive Sparse Local Attention (PSLA), to propagate high-level semantic features across frames without relying on optical flow. Specifically, given two features F^t and $F^{t+\epsilon}$ of frames I^t and $I^{t+\epsilon}$ respectively, PSLA first produces correspondence weights based on the feature affinities between F^t and $F^{t+\epsilon}$ and then aligns F^t with $F^{t+\epsilon}$ by aggregating features with corresponding weights. It is similar to attention mechanisms [33] but different in that the attended positions in PSLA are distributed in a local region with progressive sparser strides as illustrated in Fig. 1, which is inspired by the motion distribution in videos as shown in Fig. 3.

Based on PSLA, a video object detection framework is proposed, in which expensive extraction of high-level features is performed on sparse key frames while low-cost extraction of low-level features is applied on dense non-key frames. Based on the extracted features, PSLA is used in two distinct and complementary situations: (1) to propagate high-level features (at a given layer of the network) from key frames to non-key frames. This allows us to assign most of the computation cost to key frames and improves efficiency when testing without sacrificing accuracy. Moreover, a small network named *Quality Net* is devised to complement the propagated high-level features with low-level information from features of non-key frames, in order to reduce the aliasing effect of feature propagation. We name this procedure Dense Feature Transforming (DenseFT). (2) to maintain a temporal feature F_t that models temporal appearance of the video, by propagating high-level features across key frames. Meanwhile, an *Update Net* is proposed to recursively update F_t with high-level features of key frames. Our ablation study shows that exploiting temporal context contributes to a substantial gain in performance. We name this procedure Recursive Feature Updating (RFU).

We conducted extensive experiments on ImageNet VID [32] for video object detection. Our results are on par with or outperform state-of-the-art methods in both speed and accuracy with reduced model size. In addition, we show that our model can generalize to other tasks such as video semantic segmentation on the CityScapes dataset [6].

In summary, the contributions of this paper include:

- We propose a novel module Progressive Sparse Local Attention (PSLA) to establish the spatial correspondence between feature maps without relying on extra optical flow models, which reduces model parameters significantly while achieves better results.
- Based on PSLA, two techniques, Recursive Feature Updating (RFU) and Dense Feature Transforming (DenseFT), are developed to model temporal appearance and enhance the feature representation of non-key frames, respectively.
- We introduce a novel framework for video object detection which achieves state-of-the-art performance on ImageNet VID [32].

2. Related Work

Image Object Detection. Existing state-of-the-art methods for image object detection mostly follow two paradigms, two-stage and single-stage. A two-stage pipeline consists of generation of region proposals, region classification, and location refinement. R-CNN [13] is a seminal work of two-stage methods. Fast R-CNN [12] improves the speed and accuracy by sharing computation of feature extraction while Faster R-CNN[31] learns to generate region proposals. Some following variants, *e.g.* R-FCN [7] and FPN [26], further improve the performance. Comparing to two-stage detectors, single-stage methods are more efficient but less accurate. SSD [28] produces detection results from default anchor boxes from multiple feature maps. YOLO [29, 30] formulate detection as a regression problem. Lin *et al.* [27] propose focal loss to address the problem of data imbalance. In this paper, we use R-FCN as our base detector.

Video Object Detection. Different from image object detection, methods for video detection should take temporal information into account. T-cnn [20] utilizes off-the-shelf optical flow to propagate bounding boxes. Then the boxes are re-scored and removed by considering the temporal context of videos. Tpn [19] proposes a tubelet proposal network and employs an LSTM to incorporate tem-

poral information from tubelet proposals. To boost the performance, MANet [35] and FGFA [42] use the optical flow estimated by FlowNet [8] to aggregate the feature of multiple nearby frames. Instead of relying on optical flow, D&T [10] predicts the bounding boxes of the next frame by performing correlation between the features of current and next frame. To reduce the computation cost, Zhu *et al*. [43, 41] use optical flow to propagate the high-level features of keyframes to other frames and avoid extracting expensive feature frequently. Chen *et al*. [4] improve both speed and accuracy by designing a time-scale lattice. But an extra classifier is required to re-score the bounding boxes. It increases the model parameters greatly. The closest work to ours is STMN [37], which utilizes a module similar to correlation to align feature maps in a local region. Different from [37], our approach focuses on a sparse neighborhood and softmax normalization is utilized to better establish spatial correspondence. We improves both speed and accuracy while STMN improves accuracy at the expense of runtime.

Self-attention. Self-attention is a mechanism first introduced in [33] for machine translation. To integrate enough context and long-range information in a sequence, it computes the response at a position in a sequence by taking the weighted mean values of all positions, where the weights are learned by backpropagation without explicit supervision. Bahdanau *et al*. [1] apply soft attention to machine translation aiming to capture soft alignments between the source and target words. Unlike those previous works [1, 33], our proposed PSLA is a more general form of self-attention. In this paper, it is applied in the temporal-spatial domain towards aligning two feature maps.

Nonlocal Operators. Nonlocal is a traditional filter algorithm [2] that is widely used in image denoising [3, 24], super-resolution [14] and texture synthesis [9]. Those approaches compute response as a weighted mean of all the pixels in an image, where the weights are obtained based on the patch appearance similarity. More recently, based on the same principle, Wang *et al*. [36] develop a nonlocal operator utilized for video classification and object detection. It aims to capture long-range dependency within feature maps and augments the receptive field. This operator is further extended to image generation [40] and semantic segmentation [11, 18, 39]. Different from those methods, PSLA focuses on a local region with progressive sparser strides.

3. The Proposed Method

3.1. Overview

The pipeline of our framework is illustrated in Fig. 2. Given a video, each frame is first processed by a CNN to extract features; it is followed by a task network N_t for a specific task, such as object detection in this paper. To save the computational cost, frames are divided into key frames and non-key frames in our framework, for which the feature extraction networks are different, denoted as N_f and N_l respectively. The feature extraction network for non-key frames N_l is a more lightweight one than N_f. In addition, to make use of the long term temporal information embedded in the video, a temporal feature F_t is maintained across the whole video, which is gradually updated at key frames by the proposed *Recursive Feature Updating* (RFU) module. Aided by the temporal feature, the semantic features of key frames will also be enhanced by RFU to benefit the final task. Meanwhile, due to the lightweight network used for non-key frames, their features are less powerful for the final task. For this reason, the *Dense Feature Transforming* (DenseFT) module is proposed to enrich their features by propagating from the temporal feature F_t. The key assumption for such a design is that the contents of non-key frames are similar to that of nearby key frames. The core of RFU and DenseFT is to align and propagate the temporal feature to that of the currently processed frame, which is addressed by the *Progressive Sparse Local Attention* (PSLA) module. In the following, we will describe in detail the proposed PSLA, RFU, and DenseFT.

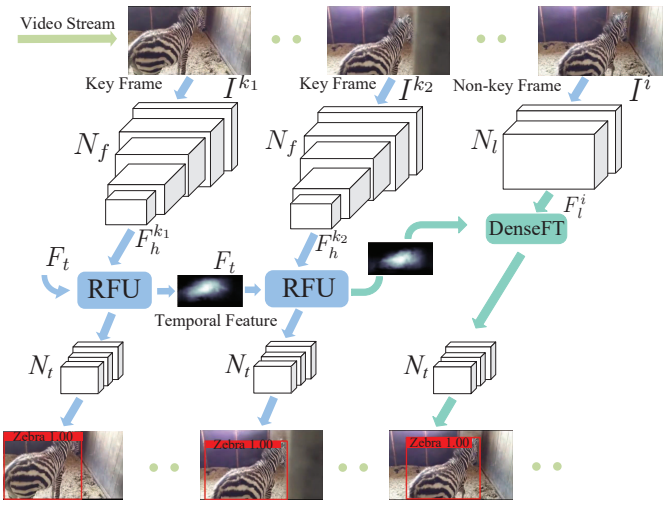

Figure 2. Pipeline of the proposed video detection framework. Only two key frames I^{k1}, I^{k2} and one non-key frame I^i are shown for simplicity. Key frames are firstly fed into N_f to produce high-level features F_h^{k1} and F_h^{k2} while non-key frames are fed into a low-cost network N_l to extract low-level features F_l^i. Based on the high-level features, a temporal feature F_t is maintained by *Recursive Feature Updating* (RFU) to model the temporal appearance of videos, where F_t is updated recursively. Meanwhile, *Dense Feature Transforming* (DenseFT) is utilized to propagate semantic features from updated F_t from the nearest key frame to non-key frames. This whole procedure is applied along the entire sequence. PSLA is embedded in RFU and DenseFT for feature alignment and propagation. The outputs of RFU or DenseFT are fed into a task network N_t to produce the detection results.

3.2. Progressive Sparse Local Attention

The core of our framework is to align and propagate feature maps across frames. To this end, we introduce Progressive Sparse Local Attention (PSLA), a novel module that

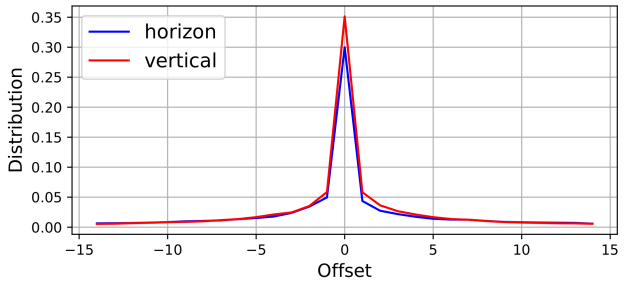

Figure 3. Optical flow field of sampled 100 ImageNet VID videos computed by FlowNet in horizon and vertical dimension. Best viewed in color.

aims to establish the spatial correspondence between two feature maps in order to propagate features among them.

PSLA proceeds by first computing correspondence weights based on the feature affinities between pairs of feature cells, originating from two distinct feature maps and distributed in progressively *sparser* strides (see Fig. 1). The motivation for this strategy is originated from Fig. 3, where the marginal distributions of optical flow field along the vertical and horizon axis[1] are largely concentrated around zero. This suggests that the feature cells used to compute correspondence weights can be limited to a neighborhood with progressively sparser strides. This setting enables PSLA to focus more on nearby positions (associated with small motions) and less on the positions far away (associated with larger motions) and also in accordance with the characteristic of visual perceptional organization of retina [34].

Formally, let F^t and $F^{t+\epsilon}$ be feature maps of frame I^t and $I^{t+\epsilon}$ respectively, and their corresponding embedded features are denoted as $f(F^t)$ and $g(F^{t+\epsilon}) \in \mathbb{R}^{c\times h\times w}$, where c, h, w is the number of channels, height and width of embedded feature maps respectively. The embedding functions $f(\cdot)$ and $g(\cdot)$ here are used to reduce the channel dimension of F^t and $F^{t+\epsilon}$ for saving computation. PSLA compares each feature cell from $g(F^{t+\epsilon})$ to surrounding cells from $f(F^t)$ at local sparse locations. The resulting feature affinities are normalized to produce the weights used to align F^t. The feature cells with higher affinities, indicating higher correspondence, will get higher weights and a larger proportion of their information is propagated to a new feature cell. Finally, the aligned features are propagated to frame $I^{t+\epsilon}$. At this stage, we aim to explain the general operations of PSLA thus do not specify how F^t and $F^{t+\epsilon}$ come from, which will be clarified in sect. 3.3 and 3.4.

Specifically, the operation of PSLA can be formulated to two steps as follows: The first step is to produce sparse correspondence weights based on the feature affinities. Given two feature maps F^t and $F^{t+\epsilon}$, embedded via two functions $f(\cdot)$ and $g(\cdot)$, the procedure to compute affinity between two feature cells at positions p_1 and p_2 is defined as

$$c_{(p_1,p_2)} = \left\langle g(F^{t+\epsilon}_{(x_1,y_1)}), f(F^t_{(x_2,y_2)}) \right\rangle, \tag{1}$$

where (x_1, y_1) and (x_2, y_2) are position coordinates of p_1, p_2 respectively and $g(F^{t+\epsilon}_{(x_1,y_1)}), f(F^t_{(x_2,y_2)}) \in R^{c\times 1\times 1}$. $\langle\cdot\rangle$ represents the inner product. For each location (x, y) in $g(F^{t+\epsilon})$, only the positions in $\Phi(x, y)$ of $f(F^t)$ are considered; $\Phi(x, y)$ is a neighborhood defined by progressively sparser strides and a max displacement d. For clarity, we divide $\Phi(x, y)$ into a series of sub-regions as

$$\Phi(x, y) = \{\Phi_0(x, y), \Phi_1(x, y), ..., \Phi_d(x, y)\}, \tag{2}$$

where

$$\begin{aligned} &\Phi_0(x, y) = \{(x, y)\}, \\ &\Phi_s(x, y) = \{(x + a, x + b), \forall a, b \in \{s, 0, -s\}\}\backslash\{(x, y)\}, \end{aligned} \tag{3}$$

s is set to satisfy $1 \le s \le d$ in our implementation. $\Phi_s(x, y)$ stands for the positions in sub-region with stride s. The spatial arrangement of $\Phi(x, y)$ in $g(F^t)$ is shown in Fig. 1, where regions of different colors correspond to different sub-regions $\Phi_s(x, y)$. As stated in the beginning of this section, it is designed as a progressively sparser grid from center to outside. Then we can compute the normalized correspondence weights:

$$\hat{c}_{(p_1,p_2)} = \frac{exp(c_{(p_1,p_2)})}{\sum_{p_2\in\Phi(x_1,y_1)} exp(c_{(p_1,p_2)})}. \tag{4}$$

By introducing a *softmax* as normalization, we force the weights to compete with each other. As a result, PSLA can capture the most similar and critical feature in the region, similar to an attention mechanism [1] and can implicitly establish spatial correspondence between two feature maps.

Then, in the second step, F^t can be aligned with $F^{t+\epsilon}$ by aggregating the corresponding feature cells with correspondence weights:

$$\hat{F}^{t+\epsilon}_{(x_1,y_1)} = \sum_{p_2:(x_2,y_2)\in\Phi(x_1,y_1)} \hat{c}_{(p_1,p_2)} F^t_{(x_2,y_2)}. \tag{5}$$

The procedure of aligning feature using PSLA can be formulated as $\hat{F}^{t+\epsilon} = PLSA(F^{t+\epsilon}, F^t)$, which is the core module embedded in RFU (sect. 3.3) and DenseFT (sect. 3.4).

3.3. Recursive Feature Updating

Videos provide rich information that are beneficial for object recognition, *e.g.* visual cues and temporal context from nearby frames. However, image object detectors ignore the appearance and context information from previous frames in a video sequence. This inspires us to propose *Recursive Feature Updating* (RFU). RFU is a procedure that aggregates and integrates semantic features of sparse key frames along time, aiming to increase detection accuracy by exploiting temporal context.

[1]Precisely, the optical flow field is computed with FlowNet[8], on 100 videos randomly sampled from ImageNet VID training split[32]

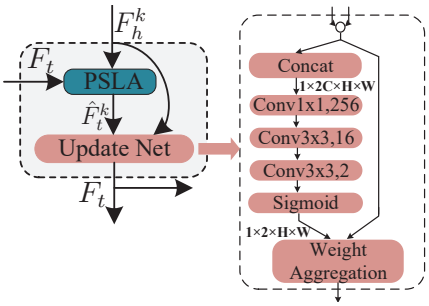

Figure 4. Recursive Feature Updating (RFU). (Conv$k \times k, n$) is a convolution layer with kernel size of k and n output channels.

Specifically, RFU maintains and updates a temporal feature F_t with semantic features of sparse key frames recursively throughout the whole video. In this procedure, directly updating F_t with the feature of a new key frame can be problematic because the movement of objects in videos would generate misaligned spatial features. Therefore PSLA is exploited to enforce the spatial consistency between F_t and high-level features of the new key frame. Given a high-level feature F_h^k of a new key frame I^k, where k is the index of key frame in the image sequence, the operation of PSLA can be formulated as $\hat{F}_t^k = PSLA(F_h^k, F_t)$.

After aligning the temporal feature, a tiny neural network, named *Update Net*, is devised to fuse $\hat{F}_t^k$ with F_h^k adaptively, with the goal of incorporating temporal context of videos into $\hat{F}_t^k$. As shown in Fig. 4, *Update Net* takes the concatenation of $\hat{F}_t^k$ and F_h^k as inputs. Then it produces the adaptive weights $\hat{W}^k$ and W^k through multiple layers of convolution, where $\hat{W}^k$ and $W^k \in \mathbb{R}^{1\times h\times w}$ indicate the importance of feature cells at each spatial location of two different feature maps. The weights are normalized over two feature maps for every spatial location so that $\hat{W}_{ij}^k + W_{ij}^k = 1$. Finally, F_t is updated based on the weights:

$$F_t = \hat{W}^k \cdot \hat{F}_t^k + W^k \cdot F_h^k, \tag{6}$$

where $\cdot$ is the Hadamard product (*i.e.* element-wise multiplication) after broadcasting the weight maps. Finally the updated F_t is used in place of F_h^k to produce results of key frame I^k and taken as the updated temporal feature.

3.4. Dense Feature Transforming

Since the features extracted by N_l for non-key frames are less powerful, we introduce Dense Feature Transforming (DenseFT) to generate semantic features of non-key frames by feature transformation and propagating from the maintained temporal feature F_t.

Specifically, the extracted low-level features F_l are used by PSLA to propagate semantic features from the temporal feature F_t at the nearest key frame. However, these low-level features do not contain sufficient semantic information to find spatial correspondence. The aligned feature may fail to preserve critical information. To address this issue, a light-weight network *Transform Net* is employed

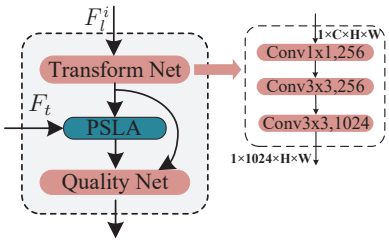

Figure 5. Dense Feature Transforming (DenseFT).

to further encode the extracted low-level features, aiming to approximate the high-level semantic features. This is a pivotal step because it not only enriches the semantic information of low-level features but also avoids the gradients generated by feature propagation flowing into N_l directly, hence improving the robustness of training as well. The encoded features are fed into PSLA to align F_t with non-key frames.

After propagating F_t to non-key frames, we fuse it with the low-level features F_l. The reason for this is the aliasing effect, caused by weighted aggregation in feature alignment, may make the propagated feature lose some details of object appearance that are important for recognition. To this end, a network *Quality Net* is embedded in DenseFT to complement detail information. Finally, the output of *Quality Net* is fed into N_t to produce results of non-key frames.

3.5. Implementation Details

We use ResNet-101 pretrained on ImageNet as N_f for feature extraction, whose layers lower than *res4b3* (including *res4b3*) are selected to construct N_l. Following [43], an RPN is used to generate region proposals and R-FCN is used as the task-specific network N_t for object detection. The embedding function $f(\cdot)$ and $g(\cdot)$ in Equ. (1) are implemented with 1×1 convolution layers with 256 filters. For the hyper-parameter of PSLA, max displacement d is set as 4 by default. The whole network including RFU and DenseFT is trained end-to-end on 8 GPUs for 120K iterations using SGD. Learning rate is 2.5×10^{-4} in the first 80K iterations and 2.5×10^{-5} in the last 40K iterations. During testing, we employ a fixed key frame schedule similar to [43], *i.e.*, a video is split into segments containing an equal number of frames and the middle frames are chosen as key frames. Key frame interval l is set as 10 by default.

The details of *Update Net* is illustrated in Fig. 4. A 1×1 convolution layer is first used to reduce the features to 256 channels, which is followed by two 3×3 convolution layers with 16 and 2 filters respectively, to produce the corresponding spatial weights for each feature. The structure of *Quality Net* is the same as *Update Net*.

As shown in Fig. 5, *Transform Net* is implemented with a bottleneck block. Firstly, a convolution layer with 1×1 kernel is used to reduce the feature channels. Then, two successive 3×3 convolution layers with 256 and 1024 filters respectively are appended to further encode the feature.

4. Experiments

4.1. Dataset and Setup

We evaluate our framework on ImageNet VID [32] dataset that contains objects of 30 classes with fully annotated bounding boxes. Following the protocols in [43], the model is trained on the intersection of training split of VID and a subset of ImageNet DET [32] with the same categories as VID. The trained model is tested on the validation split of VID.

During training, we train the network with a batch of three images on each GPU. Each batch is sampled from either ImageNet VID or ImageNet DET at $1:1$ ratio. When sampling from VID, we first sample an image as a non-key frame I^i. Then we sample another two images I^{k1} and I^{k2} near the non-key frame in a random offset as key frames. Specifically, if I^i is the n^{th} frame of a video then I^{k1} lies in $[-l+n, -0.5l+n]$ and I^{k2} lies in $[-0.5l+n, 0.5l+n]$, where l is the key frame interval. Three images are the same when sampling from ImageNet DET. Only non-key frames are provided with labels in the training phase.

4.2. Results

We compare our framework with several state-of-the-art methods for video object detection w.r.t accuracy and complexity.

The results are shown in Table 1. Our method achieves 77.1% mAP at runtime of 30.8\18.7 fps on TITAN V\X when using ResNet101 as a backbone. It surpasses the frame baseline (*i.e*. R-FCN [7]) both in terms of accuracy and runtime, showing the potential of exploiting temporal information in videos to improve the object detection performance. Compared to the optical flow based methods such as DFF [43] and FGFA [42], our method achieves much higher mAP and is only slightly slower than DFF. However, it is worth to note that our method significantly reduces the model parameters by nearly 34%($96.6M \rightarrow 63.7M$). Although MANet [35] achieves 1% higher mAP than our method, it is much slower. Towards [41] is inferior to our method when using the same backbone.

Note that, due to the high efficiency of the proposed framework, it can be used with more powerful backbones to further improve the accuracy while still maintaining fast runtime. As can be seen, using ResNet101+DCN as the backbone, our method achieves 80.0% mAP at runtime of 26.0\13.34 fps on TITAN V\X, which is better than recent advances in terms of both accuracy and speed. The most competing method to ours is ST-lattice [4], which obtains higher fps with lower mAP. Nevertheless, ST-lattice requires an extra ResNet-101 based classifier to re-score the bounding boxes and two ResNet18 models to propagate and refine bounding boxes. For these reasons, it takes at least 100M parameters (required by all models). For comparison, our best model is much smaller, requiring about 72M parameters.

Methods	mAP (%)	runtime (fps)	model size (params)	Backbone
TCN [22]	47.5	-	-	GoogLeNet
TPN [19]	68.4	2.1(X)	-	GoogLeNet
R-FCN [7]	73.9	4.05(K)	59.6M	ResNet101
TCNN [20]	73.8	-	-	GoogLeNet
DFF [43]	73.1	20.25(K)	96.6M	ResNet101
D(&T loss) [10]	75.8	-	-	ResNet101
FGFA [42]	76.3	1.36(K)	100.4M	ResNet101
D&T(online) [10]	78.7	5.3(X)	-	ResNet101
D&T($\delta = 1$) [10]	79.8	-	-	ResNet101
MANET [35]	78.1	5(XP)	-	ResNet101
ST-lattice [4]	79.6	20(X)	$>$ 100M	ResNet101
Towards [41]	78.6	13.0(X)	-	ResNet101+DCN
Ours	77.1	30.8(V)\18.73(X)	63.7M	ResNet101
Ours	**80.0**	**26.0(V)\13.34(X)**	**72.2M**	ResNet101+DCN
FGFA [42] + [15]	78.4	1.14(K)	100.4M	ResNet101
MANET [35] + [15]	80.3	-	-	ResNet101
STMN [37] + [15]	80.5	1.2(X)	-	ResNet101
Ours + [15]	78.6	5.7(X)	-	ResNet101
Ours + [15]	**81.4**	**6.31(V)\5.13(X)**	**72.2M**	ResNet101+DCN

Table 1. Performance of our method and state-of-the-art methods on ImageNet VID. Results of other methods are obtained from their papers, where different GPUs were used. X means TITAN X, XP means TITAN XP, K means K40, Ti means 1080 Ti and V means TITAN V.

After combining with Seq-NMS [15], the mAP of our method finally reaches 81.4%, outperforming all the state-of-the-art methods to the best of our knowledge. MANet [35] and STMN [37] also achieve very high mAPs when combined with Seq-NMS, however, they suffer from high computation complexity since they use more than 10 nearby frames to augment the feature of the reference frame. Conversely, our method only requires few key frames to propagate the features and in the meantime reduces the feature extraction time for non-key frames with a lightweight network, thus significantly reducing its runtime. To sum up, the overall performance of our method is better than previous works, achieving a very good tradeoff among accuracy, speed and model size.

4.3. Ablation Study

We conduct ablation study on ImageNet VID to validate the effectiveness of PSLA and the proposed framework. After introducing different configurations used for ablation study, we firstly compare PSLA to existing non-optical flow alternatives for feature propagation. Then we compare PSLA to optical flow. Finally, we conduct ablation study on different modules of the proposed framework. We also show that the proposed framework is general enough to benefit other kinds of feature propagation methods.

Besides relying on the widely used optical flow to propagate feature maps, there are two typical alternatives in literature, MatchTrans [37] and Nonlocal [36]. Basically, MatchTrans computes the propagation weights by accumulating all similarity scores in the local region while Nonlocal considers all positions. By contrast, PSLA uses a progressively sparse local region. It also applies softmax when computing the propagation weights so that spatial correspondence

Methods	Feature Propagation	Transform Net	RFU	Quality Net
Nonlocal_S	Nonlocal [36]	✓	✗	✗
Nonlocal_F	Nonlocal [36]	✓	✓	✓
MatchTrans_S	MatchTrans [37]	✓	✗	✗
MatchTrans_F	MatchTrans [37]	✓	✓	✓
DensePSLA_S	Dense PSLA	✓	✗	✗
DensePSLA_F	Dense PSLA	✓	✓	✓
our method (a)	PSLA	✓	✗	✗
our method (b)	PSLA	✓	✓	✗
our method (c)	PSLA	✓	✓	✓

Table 2. The configuration of different methods for ablation study.

Methods	max displacement	mAP (%)	runtime (fps)	parameters (M)
Nonlocal_S	-	72.1	40(V)	62.7
MatchTrans_S	2	71.4	41.2(V)	62.7
DensePSLA_S	2	72.9	41.2(V)	62.7
our method (a)	2	**73.6**	**42.7(V)**	62.7
MatchTrans_S	3	71.5	40.8(V)	62.7
DensePSLA_S	3	73.7	40.8(V)	62.7
our method (a)	3	**74.3**	**42.5(V)**	62.7
MatchTrans_S	4	72.5	40.6(V)	62.7
DensePSLA_S	4	73.6	40.6(V)	62.7
our method (a)	4	**74.4**	**42.0(V)**	62.7
MatchTrans_S	5	72.4	40.2(V)	62.7
DensePSLA_S	5	73.0	40.2(V)	62.7
our method (a)	5	**73.8**	**41.4(V)**	62.7

Table 3. Comparison of different feature propagation methods.

can be implicitly established. To better analyse the effectiveness of using progressively sparse local region, we implement a dense version of PSLA (denoted as DensePSLA) which uses all positions in the local region as MatchTrans does but with the same way to compute propagation weights as PSLA does (*i.e.* with softmax, Equ. (4)). Furthermore, to show the performance of different feature propagation methods, a simple object detection framework is implemented by only propagating the feature of preceding key frame to non-key frames. These methods are denoted by adding _S to the propagation methods, such as Nonlocal_S, MatchTrans_S. By contrast, _F means using the RFU and DenseFT modules in our video detection framework. All these evaluated methods in ablation study are summarized in Table 2. Note that the *Transform Net* is used on all these methods as it enables stable training according to our experiments.

Performance of different feature propagation methods The results of using different feature propagation methods are listed in Table. 3. By attending to the local region instead of all positions, our method(a) outperforms Nonlocal_S by a large margin. Moreover, when comparing to MatchTrans_S and DensePSLA_S, our method(a) achieves better results at all max displacement settings and consumes less runtime as well, demonstrating the effectiveness and importance of introducing progressive sparsity in PSLA.

Fig. 6 shows the trade-off between speed and accuracy of different methods in our video detection framework with different max displacements[2]. Essentially, larger key frame interval means a larger number of non-key frames whose feature extraction is significantly reduced, thus faster running time will it be. Therefore, by setting different key frame intervals for different methods, we can obtain different mAPs versus different speeds. It is obvious that the proposed method consistently outperforms other competitive methods at all evaluated key frame intervals. We can also observe from Fig. 6 that, mAP increases along with the speed at the beginning (*i.e.*, small key frame interval) but decreases when the key frame interval reaches a large number. On one hand, small key frame interval only causes small motion of objects between the key and non-key frames which is hard to be captured in high-level feature maps whose receptive field is 16×16. Thus, feature propagation may aggregate harmful information and hurt the performance. On the other hand, too large key frame interval leads to a very large motion of objects, in which case establishing spatial correspondence is quite challenging. As a result, the accuracy decreases when the key frame interval is set either too small or too large.

Methods	max displacement	key frame interval	mAP(%)
DFF*	-	15	72.2
DFF*	-	25	69.7
DFF*	-	35	67.5
our method (a)	4	15	72.9 (+0.7)
our method (a)	4	25	70.5 (+0.8)
our method (a)	4	35	68.5 (+1.0)

Table 4. Comparison between PSLA and DFF at different key frame intervals. * means our re-implementation.

PSLA VS. optical flow In order to validate the advantage of PSLA on capturing spatial correspondence on feature maps, we compare PSLA with DFF [43], a pioneer work on video object detection with optical flow. The results are illustrated in Table 4, where the results of DFF are obtained by our own implementation. Obviously, our method(a) performs much better than DFF. The larger the key frame interval is, the more significant the relative improvement is. It verifies that by directly establishing the spatial correspondence in feature maps, PSLA aligns two feature maps better than aligning them based on pixel-level correspondence from optical flow.

Effectiveness of the proposed framework Table 5 gives the results of our method when gradually adding RFU and DenseFT modules. Firstly, only using PSLA to propagate features from the nearby key frame achieves 74.4% mAP.

[2]Nonlocal is a global method, so it does not have a parameter of max displacement. Thus, the four curves of Nonlocal_F is identical in Fig. 4

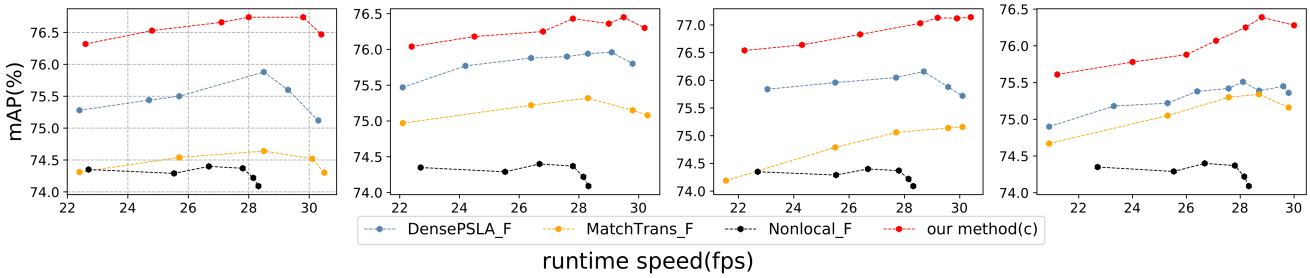

Figure 6. mAP vs. runtime for different methods. The results from left to right correspond to max displacement d, from 2 to 5.

Methods	mAP(%)	runtime (fps)
our method(a)	74.4	42.0
our method(b)	75.8	31.2
our method(c)	**77.1**	30.8
Nonlocal_S	72.1	40.0
Nonlocal_F	74.1	28.3
MatchTrans_S	72.5	40.6
MatchTrans_F	75.2	30.1
DensePSLA_S	73.6	40.6
DensePSLA_F	75.7	30.1

Table 5. The proposed video detection framework benefits various feature propagation methods. The max displacement is set as 4. All results are tested on TITAN V.

Then, by exploiting RFU to model the temporal appearance used for feature propagation (*i.e.*, method(b)), the performance of our method(a) is improved by 1.4%. Finally, the result is further improved to 77.1% by adding DenseFT to enhance the feature representations of non-key frames. RFU and DenseFT have also been used in other feature propagation methods, and we can observe consistent performance improvement from Table 5.

4.4. Extension to other tasks

The proposed framework shown in Fig. 2 could be actually used for other vision tasks beyond object detection studied in this paper. Here we conduct a simple experiment on video object segmentation to demonstrate the possible extension of our method. Specifically, we replace the R-FCN used in this paper with deeplab [5] for semantic segmentation in videos. In this case, our method is similar to Low-latency [25]. The difference is that Low-latency predicts location-adaptive kernel weights to generate features of non-key frames while we utilize the parameter-free PSLA to perform feature alignment. The experiment is conducted on CityScapes [6], which contains snippets of street scenes from 50 different cities. We train our framework on the training set and evaluate the pixel-level mean intersection-over-union (mIoU) on the validation set. More training details are given in supplementary materials.

The results are summarized in Table 6. For a fair comparison, we re-implemented the frame baseline and DFF with the same setting as our method. Our method achieves very close performance to the frame baseline with higher fps, verifying the effectiveness of the proposed PSLA. It also achieves the best mIoU comparing to DFF and DVS at a reasonable speed. For comparison, DFF achieves faster runtime than the baseline at the cost of accuracy by a large margin. Comparing to DVS [38], a better mIoU is achieved by ours. Although faster, DVS relies on the FlowNet to propagate features, thus having much more parameters and being less preferable to practical scenarios. As for Low-latency, it is hard to compare directly since its segmentation head is not specified. Our good result on video semantic segmentation demonstrates the universality of the proposed method for video recognition.

Methods	mIoU(%)	runtime (fps)
DVS [38]	70.4	19.8(Ti)
DFF [43]	69.2	5.6(K)
Frame baseline [43]	71.1	1.52(K)
DFF (*)	69.8	15.4(V)
Frame baseline (*)	72.1	6.2(V)
Ours	71.9	11.6(V)

Table 6. Comparison of different methods on Cityscape. * means our re-implementation.

5. Conclusion

In this paper, we propose a novel framework for video object detection. At its core, a novel module PSLA is proposed to propagate features effectively. In addition, two techniques RFU and DenseFT are designed to model temporal appearance and enhance feature representations. We conduct ablation studies on ImageNet VID to prove the effectiveness of our framework on video object detection. The proposed framework achieves 81.4% mAP on ImageNet VID and outperforms state-of-the-art methods. Additional experiment of video semantic segmentation on CityScapes demonstrates the generalization ability of the framework.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grants 61573352, 61876180, 91646207, 61773377, the Young Elite Scientists Sponsorship Program by CAST (2018QNRC001), and the Beijing Natural Science Foundation under Grant L172053.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. 3, 4

[2] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, 2005. 3

[3] Harold Christopher Burger, Christian J Schuler, and Stefan Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms. *CoRR*, abs/1211.1552, 2012. 3

[4] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing video object detection via a scale-time lattice. In *CVPR*, 2018. 2, 3, 6

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *T-PAMI*, 40(4):834–848, 2018. 8

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 8

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1, 2, 6

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2, 3, 4

[9] AA Efros and TK Leung. Texture synthesis by nonparametric sampling. inproceedings of the international conference on computer vision. *ICCV*, 1999. 3

[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *CVPR*, 2017. 2, 3, 6

[11] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, pages 3146–3154, 2019. 3

[12] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 1, 2

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[14] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, 2009. 3

[15] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *CoRR*, abs/1602.08465, 2016. 1, 6

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1

[18] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. *CoRR*, abs/1811.11721, 2018. 3

[19] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017. 2, 6

[20] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *T-CSVT*, 28(10):2896–2907, 2017. 1, 2, 6

[21] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 1

[22] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 6

[23] Byungjae Lee, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung, and Phill Kyu Rhee. Multi-class multi-object tracking using changing point detection. In *European Conference on Computer Vision*, pages 68–83. Springer, 2016. 1

[24] Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *CVPR*, 2017. 3

[25] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *CVPR*, 2018. 8

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 2

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 2

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1, 2

[30] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2, 4, 6

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2, 3

[34] Brian A Wandell and Jonathan Winawer. Computational neuroimaging and population receptive fields. *Trends in cognitive sciences*, 2015. 4

[35] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. Fully motion-aware network for video object detection. In *ECCV*, 2018. 2, 3, 6

[36] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. 2018. 3, 6, 7

[37] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *ECCV*, 2018. 2, 3, 6, 7

[38] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *CVPR*, 2018. 8

[39] Yuhui Yuan and Jingdong Wang. Ocnet: Object context network for scene parsing. *CoRR*, abs/1809.00916, 2018. 3

[40] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018. 3

[41] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *CVPR*, 2018. 2, 3, 6

[42] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, 2017. 2, 3, 6

[43] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, 2017. 2, 3, 5, 6, 7, 8