Video Object Detection with Locally-Weighted Deformable Neighbors

Zhengkai Jiang,[1,2] **Peng Gao,**[3] **Chaoxu Guo,**[1,2] **Qian Zhang,**[4]
Shiming Xiang,[1,2] **Chunhong Pan**[1,2]
[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]The Chinese University of Hong Kong [4]Horizon Robotics
{zhengkai.jiang,chaoxu.guo,smxiang,chpan}@nlpr.ac.cn penggao@ee.cuhk.edu.hk qian01.zhang@hobot.ai

Abstract

Deep convolutional neural networks have achieved great success on various image recognition tasks. However, it is non-trivial to transfer the existing networks to video due to the fact that most of them are developed for static image. Frame-by-frame processing is suboptimal because temporal information that is vital for video understanding is totally abandoned. Furthermore, frame-by-frame processing is slow and inefficient, which can hinder the practical usage. In this paper, we propose LWDN (Locally-Weighted Deformable Neighbors) for video object detection without utilizing time-consuming optical flow extraction networks. LWDN can latently align the high-level features between keyframes and keyframes or non-keyframes. Inspired by (Zhu et al. 2017a) and (Hetang et al. 2017) who propose to aggregate features between keyframes and keyframes, we adopt brain-inspired memory mechanism to propagate and update the memory feature from keyframes to keyframes. We call this process Memory-Guided Propagation. With such a memory mechanism, the discriminative ability of features in keyframes and non-keyframes are both enhanced, which helps to improve the detection accuracy. Extensive experiments on VID dataset demonstrate that our method achieves superior performance in a speed and accuracy trade-off, i.e., 76.3% on the challenging VID dataset while maintaining 20fps in speed on Titan X GPU.

Introduction

Recently, deep convolutional networks (Gao et al. 2018) have achieved significant success in many computer vision tasks, including image classification (Krizhevsky, Sutskever, and Hinton 2012; Feng et al. 2018), image object detection (He et al. 2017) and image semantic segmentation (Chen et al. 2017; Hu et al. 2018). Unlike static image, video contains redundancy and temporal information cue which could be utilized to boost detection accuracy and speed.

Extending object detection from the static image to video (Gupta et al. 2014) is highly valuable in a vast number of scenarios, such as video surveillance and autonomous driving. Static image detectors are unable to handle motion blur, video defocus, unusual poses or object occasion as shown in Figure 1. Furthermore, frame-by-frame deep feature extraction on a heavy feature network like ResNet-101 (He et al. 2016) is time-consuming even on high-speed GPUs. Despite these challenges, videos provide rich temporal and motion information. Effective usage of temporal information is the key component for video object detection.

Figure 1: Typical deteriorated object appearance in videos

A few works have been proposed to exploit such temporal information in videos by means of various post-processing steps (Kang et al. 2016) on top of frame-level detectors, which aims at making object detections coherent across time. However, since temporal coherence is enforced in a second stage, typically these methods cannot be trained end-to-end. To overcome this limitation, recent work have introduced the flow-based aggregation networks (Zhu et al. 2017b) that is trained end-to-end. It exploits optical flow (Dosovitskiy et al. 2015) to find correspondences across time and aggregate features across temporal correspondences to smooth object detection over adjacent frames. However, one of the shortcomings of this method is that in addition to performing object detection, it also needs time-consuming optical flow extraction. This is disadvantageous due to the following reasons: (1) optical flow extraction is time-consuming which may reduce the detection speed; (2) training such a model requires large amounts of flow data to get a pretrained flow extraction network, etc.

To address these shortcomings, inspired by the flow warp operation and non-local networks (Wang et al. 2018), we use spatially variant weights to combine corresponding neighbors. To use the temporal information, we use the difference of the two frames as inputs to predict the weights. Furthermore, we also predict offsets to adaptivly combine the neighbors. Our methods don't need the time-consuming flow ex-

traction network, which can boost video detection speed.

In this work, we propose an effective LWDN (Locally-Weighted Deformable Neighbors) module that uses feature-weighting way to leverage temporal information for video object detection. Our LWDN learns to predict position-sensitive weights, which are used to propagate the keyframe features to keyframes or non-keyframes. Unlike the flow-based feature propagation which needs time-consuming flow extraction, our LWDN needs less computation. Furthermore, to use temporal consistency information, we also adopt brain-inspired memory mechanism to propagate and update the memory feature between keyframes and keyframes. With such a memory mechanism, the discriminative ability of the features in keyframes and non-keyframes are both enhanced, which help to improve the detection accuracy.

In short, we propose an effective LWDN module to adaptively learn the feature propagation conditioned on the low-level features between two consecutive frames. Our end-to-end trained models are superior than state-of-the-art methods with faster speed and smaller model size. Without utilizing optical flow for feature propagation, the whole object detection pipeline is easier to train and test in practice. The main contributions of this work are summarized below:

- We propose a flow-free end-to-end framework for fast video object detection which can achieve a better trade-off between speed and accuracy.
- Our light-weight LWDN can effectively align features between consecutive frames which is vital for temporal feature propagation.
- Experiments on VID dataset demonstrate that our method achieves competitive performance with remarkable speed and fewer model parameters compared with other state-of-the-art methods.
- The proposed LWDN together with the end-to-end framework can also benefit other video tasks.

Related Work

Image Object Detection

State-of-the-art methods for general object detection are mainly based on deep convolutional neural networks (Lin et al. 2014), most of which follow two paradigms, two-stage and single-stage. A two-stage pipeline firstly generates region proposals, which are then classified and refined. To speedup R-CNN (Girshick et al. 2014), ROI pooling was introduced to the feature maps shared on the whole image in SPP-Net (He et al. 2014) and Fast R-CNN (Girshick 2015). To replace Selective Search (Uijlings et al. 2013), in Faster R-CNN (Ren et al. 2015), the region proposals are generated by the Region Proposal Networks, and features are shared between RPN and Fast R-CNN. R-FCN (Dai et al. 2016) replaces ROI pooling operation on the intermediate feature maps with position-sensitivity ROI pooling operation on the final score maps. Mask-RCNN (He et al. 2017) is proposed to simultaneously do detection and segmentation. Compared with two-stage pipelines, a single stage method is often more efficient but less accurate. SSD (Liu et al. 2016) was an early attempt of this paradigm. It generates outputs from default boxes on a pyramid of feature maps. YOLO (Redmon et al. 2016) designed specific feature networks for fast object detection. RetinaNet (Lin et al. 2018) was proposed to tackle the imbalance between foreground and background classes.

Video Object Detection

Compared with object detection in images, video (Ballas et al. 2015) object detection was less studied typically, until the new VID challenge was introduced to ImageNet. Kang et al. (Kang et al. 2016) proposed a framework that integrated per-frame proposal generation, bounding box tracking and tubelet re-scoring. It is very time-consuming due to the per-frame feature computation by deep networks. Mnih et al. (Mnih et al. 2014) introduced recurrent models for visual attention. Zhu et al. (Zhu et al. 2017c) proposed an efficient framework which run expensive CNNs on sparse and regularly selected keyframes. Features are propagated to non-keyframes with optical flow. The method achieves 10X speedup than per-frame detection. Zhu et al. (Zhu et al. 2017b) proposed to aggregate nearby features along motion path, improving the feature quality. However, this method runs slowly at around 3 fps due to dense prediction and flow computation. D_T (Feichtenhofer et al. 2017) was proposed to learn object detection and cross-frame tracking with multi-task objective and link frame-level detection (Ba, Mnih, and Kavukcuoglu 2014) to tubes. However, Most of them are either flow-based methods which need time-consuming flow extraction or in slow speed.

Optical Flow

Temporal information in videos requires correspondence in raw pixels or features to build the relationship between consecutive frames. Optical Flow (Dosovitskiy et al. 2015) is widely used in many video applications. Variational approaches have dominated optical flow estimation which mainly address small displacement (Brox and Malik 2011). DFF is the first work to combine flow network with video object detection, achieving perfect performance and speeding up video detection, which shows the information redundancy in video. However, flow extraction is very slow, resulting in difficulty to further speed up video detection.

Exploiting Temporal Information in Video

Applying state-of-the-art still image detectors frame by frame to videos does not provide optimal results. This is mainly due to the low-quality images in videos. Single image detectors can't handle deteriorated images in videos (Han et al. 2016). Temporal feature aggregation provides an effective way to utilize such information (Han et al. 2016). Zhu et al. (Zhu et al. 2017a) adopted sparsely recursive feature propagation and partially update features for non-key frames for better performance. Xiao et al. (Xiao and Lee 2017) proposed Flow-guided GRU to feature aggregation which can get better feature representation for current frame. However, They are all flow-based methods. It's still time-consuming for flow extraction, although flownet is light for boosting detection.

Proposed Method

In this section, we firstly clarify that the LWDN (Locally-Weighted Deformable Neighbors) is the key module of our proposed method. Then, we introduce the inference and training process of the overall pipeline, called memory-guided propagation networks. It is developed with LWDN module, which is embedded in the overall pipeline for feature propagation. Finally, three import modules inside the pipeline, including weight predictor network, feature correlation and aggregation unit, are presented in detail.

Locally-Weighted Deformable Neighbors

Conveniently, we divide the feature of the individual video frame into low-level feature (the lower-part of CNN), high-level feature (the higher-part of CNN) ,task feature which is responsible for the final detection, as well as the memory feature which is used to propagate memory feature between keyframes and keyframes.

As in Figure 2, motivated by Non-Local Operator (Wang et al. 2018) which computes the response at a position as a weighted sum of features at all positions, we propose a local weighted neighbors operator using convolution to express linear combination of neighbors , with the kernel weights varying across sites. It's used to propagate keyframe task feature to the non-keyframes.

Let the position-sensitive weights be W, then the propagation from the previous keyframe task feature (T_h^k) to the non-keyframe (T_h^t) can be expressed as:

$$T_h^t(c,i,j) = \sum_{u=-h}^{h}\sum_{v=-w}^{w} W_{ij}^{(k,t)}(u,v)\cdot T_h^k(c,i',j') \quad (1)$$

$$i' = i - u \quad (2)$$

$$j' = j - v \quad (3)$$

$\{i,j\}$ is the location in the feature map, $\{2h+1, 2w+1\}$ is the local neighbor kernel weight size in the position $\{i,j\}$, $\{u,v\}$ is the local neighbor index around the location. c is the channel index of feature map.

Furthermore, inspired by deformable convolution (Dai et al. 2017), we also use the corresponding position sensitive offsets to adaptively combine neighbors' feature. It's used to propagate the memory feature from keyframes to keyframes. Then the propagation from the previous keyframe (T_h^k) to the new keyframe ($T_h^{k'}$) is as following:

$$T_h^{k'}(c,i,j) = \sum_{u=-h}^{h}\sum_{v=-w}^{w} W_{ij}^{(k,k')}(u,v)\cdot T_h^k(c,i',j') \quad (4)$$

$$i' = i - u + \Delta p_u \quad (5)$$

$$j' = j - v + \Delta p_v \quad (6)$$

Notice that the regular grid $T_h^k(c, i-u, j-v)$ is augmented with offsets $\{\Delta p_u, \Delta p_v\}$. Now, the sampling is on the irregular and offset location $T_h^k(c,i',j')$. As the offset $\{\Delta p_u, \Delta p_v\}$ is typically fractional, $T_h^k(c,i',j')$ is implemented via bilinear interpolation as:

$$T_h^k(c,i',j') = \sum_{(i,j)} G((i,j),(i',j'))\cdot T_h^k(c,i,j) \quad (7)$$

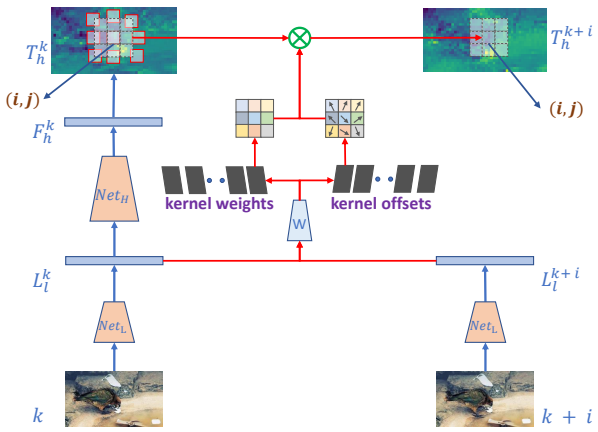

Figure 2: Locally-Weighted Deformable Neighbors Process. We call it adaptively position-sensitive feature propagation. Let k refers to the previous keyframe. T_n^k is the previous keyframe task feature. When the low-level feature L_l^k is computed, the Weight Predictor Network will take both L_l^k and L_l^{k+i} as inputs, yielding a series of position-sensitive kernel weights and corresponding kernel offsets. Then the task feature of previous keyframe T_n^k will be propagated to the current time step via spatially variant deformable convolution using predicted kernel weights and offsets.

Where (i',j') denotes an arbitrary (fractional) location, (i,j) enumerates all integral spatial locations in the feature map T_h^k, and $G(\cdot,\cdot)$ is the bilinear interpolation kernel.

$$G((i,j),(i',j')) = g(i,i')\cdot g(j,j') \quad (8)$$

where $g(a,b) = max(0, 1-|a-b|)$.

Video Object Detection Methods

Inspired by (Hetang et al. 2017) and (Zhu et al. 2017a), who propose to aggregate feature between keyframes and keyframes. We follow them to propagate the memory feature from keyframes to keyframes, then aggregate with the new keyframe feature through a quality-aware network recursively. As mentioned before, we call it Memory-Guided Propagation Networks.

Memory-Guided Propagation Networks Inference

The inference of Memory-Guided Propagation Networks is illustrated in Figure 3 and expressed in Algorithm 1. The proposed method is built on standard still image detector which consists of feature extractor $\mathcal{N}_{feat}$, the region proposal network $\mathcal{N}_{rpn}$ and the region-based detector $\mathcal{N}_{rfcn}$. We call both $\mathcal{N}_{rpn}$ and $\mathcal{N}_{rfcn}$ as $\mathcal{N}_{task}$.

First, we divide the video frames into keyframes such as $\{k^0, k^1\}$ and non-keyframes such as $\{k^1 + i\}$ as in Figure 3. The key idea of the proposed method has the following two points. One is to propagate the keyframe task feature to non-keyframes through a light Weight Predictor Network and Locally-Weighted Deformable Neighbors Operation for higher speed. Another one is to aggregate memory feature

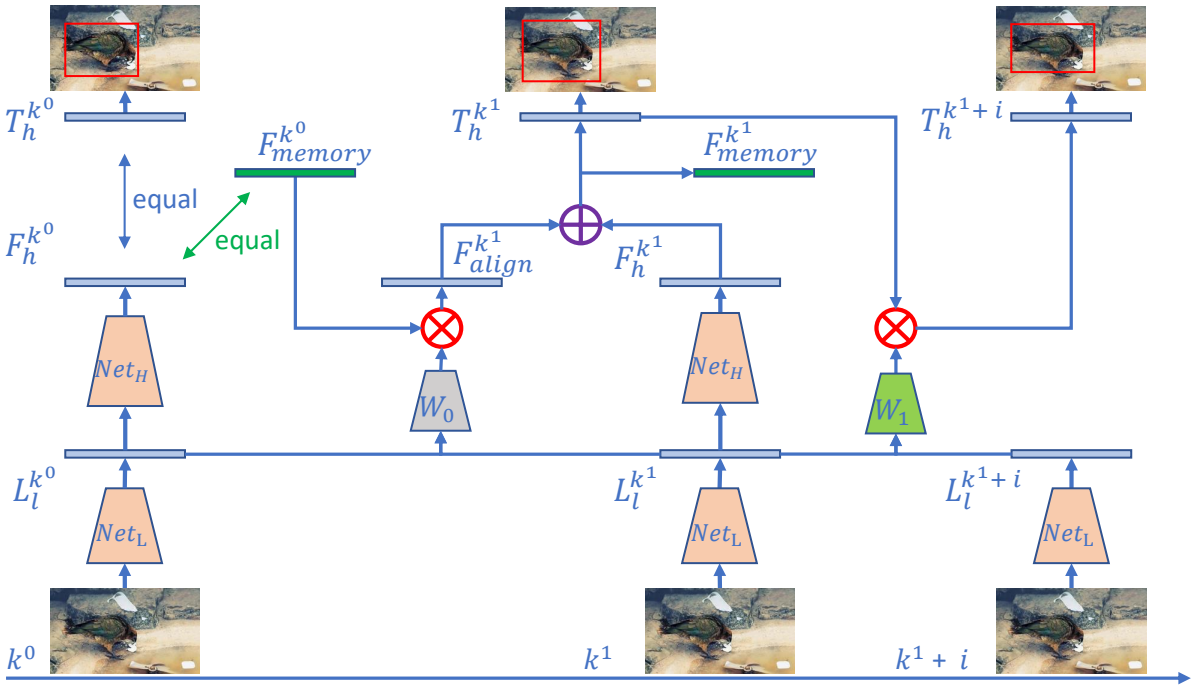

Figure 3: Memory-Guided Propagation Networks inference pipeline. There are main two inference processes, keyframe to keyframe inference for updating memory feature and keyframe to non-keyframe inference for propagating task feature. k^0 and k^1 are keyframes, where, more specifically, k^0 is the first keyframe in a video. $k^1 + i$ is the non-keyframe corresponding k^1. $\bigoplus$ is Aggregation Unit and $\bigotimes$ is LWDN Operation.

between keyframes and keyframes to enhance the feature for more accurate detection.

There are mainly two inference processes, keyframe to keyframe and keyframe to non-keyframe. For keyframe to non-keyframe process, the keyframe task feature is propagated to the non-keyframe through the LWDN (Locally-Weighted Deformable Neighbors) Operation, with weights predicted by the Weight Predictor Network. And the low-level features of corresponding two frames are fed into the Weight Predictor Network to produce the weights. For keyframe to keyframe process, first of all, the keyframe low-level feature pairs are fed into Weight Predictor Network to produce position sensitive weights and offsets. Then the aligned feature is generated by LWDN operation using the old keyframe memory feature together with the weights and the offsets. Lastly, the aligned feature together with the new keyframe high-level feature are aggregated to the task feature and the new memory which will be propagated to next keyframe recursively.

More specifically, at each time step, first, the low-level feature L_l and high-level feature F_h are computed by the lower-part of CNN Net_L and the higher-part of CNN Net_H, respectively. In the process of keyframe to keyframe propagation, the low-level feature of the two neighboring keyframes are fed into Weight Predictor Network (W_0) to produce position-sensitive weights and offsets. Then the weights and offsets along with the first keyframe memory $F_{memory}^{k^0}$ (both memory feature and task feature of the first keyframe equal the high-level feature $F_h^{k^0}$) are fed into LWDN operation to generate the aligned feature $F_{align}^{k^1}$ for the keyframe k^1. Next the aligned feature as well as the high-level feature $F_h^{k^1}$ aggregate to generate the task feature $T_h^{k^1}$ and the memory $F_{memory}^{k^1}$ which is propagated to the next keyframe recursively. For the keyframe to the non-keyframe propagation process, the low-level features of the non-keyframe and the corresponding keyframe pairs are fed into Weight Predictor Network (W_1) to only produce position-sensitive weights. Next the weights and the corresponding keyframe task feature $T_h^{k^1}$ are as inputs to LWDN operation to produce estimated task feature $T_h^{k^1+i}$ which is used to generate final detection results.

Memory-Guided Propagation Networks Training

The training process of Memory-Guided Propagation Networks is as in Figure 4. With video provided, a standard single-image object detection pipeline can be transferred to video object detection task with little modifications.

During training, each data batch contains three images $\{I_{k+d_0}, I_k, I_{k+d_1}\}$ from a same video sequence. d_0 and d_1 are random offsets whose ranges are controlled by segment length l. Specifically, d_0 lies in $[-l, -0.5l]$, while d_1 lies in $[-0.5l, 0.5l]$. This setting is coherent with inference phase as I_k represents any non-keyframe in current segment, I_{k+d_1}

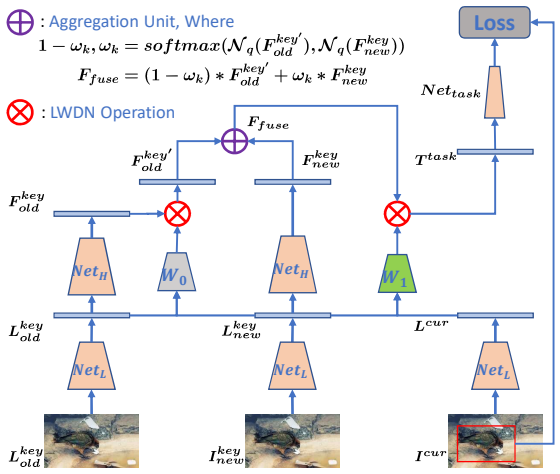

Figure 4: Memory-Guided Propagation Networks Training.

represents keyframe of current segment, and I_{k+d_0} represents old keyframe of last segment. For simplicity, the three images are dubbed as $\{I_{old}^{key}, I^{cur}, I_{new}^{key}\}$. The ground-truth at I^{cur} is provided as label.

In each iteration, first, $\mathcal{N}_{feat}$ is applied on $\{I_{old}^{key}, I_{new}^{key}\}$ to get their high-level features $\{F_{old}^{key}, F_{new}^{key}\}$ and low-level features $\{L_{old}^{key}, L_{new}^{key}\}$. Then low-level feature pairs are fed into Weight Predictor Network (W_0), yielding position sensitive weights $W_{old\to new}$ and position sensitive offsets $O_{old\to new}$, respectively. The low-level feature pairs $\{L_{new}^{key}, L^{cur}\}$ are also fed into Weight Predictor Network (W_1), only yielding position sensitive weights $W_{new\to cur}$. Then $W_{old\to new}$ and $O_{old\to new}$ together with F_{old}^{key} are fed into LWDN operation to generate $F_{old}^{key'}$. The fused feature $\mathcal{F}_{fuse}$ is generated by the aggregation unit using $\{F_{old}^{key'}, F_{new}^{key}\}$. Finally, $W_{new\to cur}$ and $\mathcal{F}_{fuse}$ are also fed into LWDN operation to produce the task feature T_{task}, which is responsible for current non-keyframe detection. We use the current non-keyframe detection loss to train the end-to-end system.

Weight Predictor Network

Weight Predictor Network is used to generate spatially variant weights and offsets. With the two pairs low-level features as inputs, we first concat them along axis 0 to generate the concated feature. After that, we reduce the concated feature to 256 channel using convolution layer with 3×3 kernel. Then we slice the reduced features along axis 0 to get reduced low-level features, respectively. After that, we correlate the reduced low-level features. Last is 3×3 kernel with 256 channels convolution and $k \times k$ channels following softmax operation to generate position sensitive weights. Specifically, for W_1 between keyframes and keyframes, there is also the 1×1 kernel with $2 \times k \times k$ channels to generate position sensitive offsets.

Algorithm 1: Inference algorithm of Memory-Guided Propagation Networks

Input: video frames $\{I\}$, segment length l
Output: detection results of the whole video
for $k = 0; k \leq N$ **do**
  $F_k^{key} = \mathcal{N}_{feat}(I_k^{key})$
  if *k=0* **then**
    $F_k^{memory} = F_k^{key}$
    $F_k^{task} = F_k^{key}$
  else
    $1-\omega_k, \omega_k =$
     $softmax(\mathcal{N}_q(F_{k-1}^{memory}), \mathcal{N}_q(F_k^{key}))$
    $F_k^{task} = (1-\omega_k) * F_{k-1}^{memory} + \omega_k * F_k^{key}$
    $F_k^{memory} = (1-g) * F_{k-1}^{memory} + g * F_k^{task}$
  end
  for $j = 0; j \leq l-1$ **do**
    $F_j^{task} = \mathcal{W}(F_k^{task}, W_{I_k^j, I_k^{key}})$
    $Det_j^k = \mathcal{N}_{task}(F_j^{task})$
  end

Feature Correlation

We use correlation which performs multiplicative path comparisons between two neighbor low-level feature maps as the input of Weight Predictor Network. Given two level feature map f_1, f_2, with w,h and c being their width, height and number of channels. Our correlation layer lets the network compare each patch from f_1 with each path from p_2 which can help to get better difference descriptor about two frames. Specially now we consider only a single comparison of two patches. The correlation of two patches centered as x_1 on the first map and x_2 in the second map is then defined as

$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle \quad (9)$$

For a square patch of size $K := 2k + 1$, obviously, it only convolves data with another data. Therefor, it has no trainable weights. We use correlation to capture the difference between two consecutive frames.

Aggregation Unit

The aggregation weights of the features are generated by a quality estimation network $\mathcal{N}_q$. It has three randomly initialized layers: a $3\times3\times256$ convolution, a $1\times1\times16$ convolution and a $1\times1\times1$ convolutions. The output is position-wise raw score map which will be applied on each channel of corresponding features. Raw score maps of different features are normalized by the score map with features and sum them up to obtain the fused features as in Algorithm 1.

Experiments

Datasets and Evaluation Metrics

We evaluate the proposed method on the ImageNet VID dataset which has been treated as a benchmark for video

Table 1: Performance comparison with the state-of-the-art systems on ImageNet VID validation set. The mean average precision (mAP %) over all classes are provided. Our LWDN outperforms the most of them considering both accuracy and speed. Besides, our model has fewer parameters than the existing flow-based models. With the ResNet-101 backbone, we can achieve 76.3% mAP while maintaining 20 fps in TITAN X GPU. Furthermore, our LWDN outperforms the corresponding static-image detector by a large margin 2.7%.

Methods	**Base Network**	**Base Detector**	**mAP (%)**	**FPS**	**Params (M)**
TPN+LSTM (Kang et al. 2016)	-	-	68.4	< 7	-
Winner ILSVRC'15	VID Winner	-	73.8	-	-
Winner ILSVRC'16	VID Winner	-	76.2	-	-
D (&T loss) (Feichtenhofer et al. 2017)	Resnet-101	R-FCN	75.8	< 7	-
R-FCN (Dai et al. 2016)	Resnet-101	R-FCN	73.6	7	-
DFF (Zhu et al. 2017c)	Resnet-101	R-FCN	73.0	**29**	97.8
FGFA (Zhu et al. 2017b)	Resnet-101	R-FCN	76.3	< 2	100.5
LWDN(ours)	Resnet-101	R-FCN	**76.3**	20	**77.5**

object detection (Russakovsky et al. 2015). VID dataset is split into 3862 training videos and 555 validation videos. All snippets are fully annotated with object bounding box and tracking IDs. Totally, there are 30 object categories in VID dataset which are a subset of ImageNet DET dataset. We report results on the validation set and use mean average precision (mAP) as the evaluation metric following the previous methods (Zhu et al. 2017c).

Although there are more than 112,000 frames in VID training set, the redundancy among video frames makes the training procedure less efficient. Moreover, the quality of frames in video is much poorer than the still images in DET dataset. Thus we follow previous approaches and train our model on an intersection of ImageNet VID and DET dataset.

Implementation Details

Our training set consists of the full ImageNet (Deng et al. 2009) VID training set together with images from the ImageNet DET training set. Only the same 30 classes categories are used. Usually, images from ImageNet DET have better quality, which is vital for video object detection training. As mentioned before, each training batch contains three images. If sampled from DET, all images within the same mini-batch will be the same. In our experiment, one GPU will hold only one mini-batch.

In both training and testing stage, images are resized to have the shorter side of 600 pixels (Ren et al. 2015). We choose `conv4_3` as the split point between the lower and higher parts of the network. We use the R-FCN detector (Dai et al. 2016) with ResNet-101 (He et al. 2016) as the backbone network which is the same with most previous video detection methods. We also adopt the OHEM (Shrivastava, Gupta, and Girshick 2016) during the training stage following DFF (Zhu et al. 2017c). For training, 4 epochs with SGD optimization method are performed on 8 GPUs with each GPU holding one mini-batch. Learning rate begins with 2.5e-4 and divides by 10 after 2.5 epochs. We also employ standard left-right flipping augmentation.

Our proposed LWDN will propagate information from keyframes to keyframes by predicting position-sensitive offsets and weights. For keyframes to non-keyframes, locally weighted attention (Li, Shi, and Lin 2018) is enough for capturing information propagation. For test stage, our method will first detect the keyframe then propagate feature maps according Weight Predictor Network and LWDN operation.

Comparison to the state-of-the-art

We compare our proposed LWDN with the existing state-of-the-art image and video object detectors. Without using optical flow information, our method outperforms previous methods heavily relying on optical flow network for feature propagation. The results consolidate that LWDN can latently learn the feature correspondence between consecutive video frames. As can be seen from table1, most methods are unable to balance between speed and performance. Our method achieves better performance with fast speed and reasonable model parameters, which can boost the superiority of our methods. Furthermore, without flow extraction network, the training process is easier compared with other methods. Simultaneously, the LWDN we proposed can also be used in other video task such as video semantic segmentation.

To sum up, first, our detector outperforms static image-based R-FCN detector with large margin (+2.7%) while maintaining high speed (20fps). Second, our model param size is smallest compared with other video object detectors using optical flow network, e.g. , 77.5M VS around 100M.

Ablation Experiments

In this section, we conduct extensive ablation studies to validate the effectiveness of the proposed LWDN. We follow the previous evaluation protocols. R-FCN and ResNet-101 are used as the base detector and the backbone network, respectively. All experiments adopt left-right flip augmentation during training stage for the ablation study.

We study the influence of sparse feature, memory propagation, quality-aware memory aggregation and end-to-end training on the final performance. We use DFF (Zhu et al. 2017c) as our baseline. DFF achieves 73.0% mAP and runs at 34 ms. Sparse feature will extract exact deep features at the keyframe while propagate information from keyframes to non-keyframes by LWDN. When applying memory propagation, features between keyframes will be latently aligned

Table 2: Accuracy and runtime of different approaches

Methods	(a)	(b)	(c)
sparse feature	√	√	√
memory		√	√
quality-aware			√
end-to-end	√	√	√
mAP(%)	73.0%	75.5%	76.3%
runtime(ms)	38	48	48

Table 3: The influence of different kernel size of LWDN

kernel size	3	5	7	9	11
mAP(%)	74.7	75.5	75.8	76.3	75.7

and naively combined (feature add) to capture the long-range video information. By naive memory propagation, performance can be boosted to 75.5% mAP. Motivated by the gating mechanism, we train a quality networks which can adaptively combine current keyframe feature with previous memory feature. Sharing with the same aim with GRU (Nilsson and Sminchisescu 2016). Quality network will strengthen and suppress memory features accordingly. By adding quality aware memory aggregation, we further boost the performance to 76.3% mAP while running 48 ms. All networks are trained end-to-end.

LWDN Kernel Size

We conduct an ablation experiment to study the influence of the kernel size on the final performance. The keyframe interval in training stage is 10 frames and the keyframe interval is also 10 frames during testing stage in default. Our proposed LWDN (Locally-Weighted Deformable Neighbors) module will align feature from keyframes to keyframes or keyframes to non-keyframes. The kernel size represents the local neighbors for feature propagation among different frames within one video snippet. When the kernel size equals 9, we achieve best results 76.3% mAP for video object detection as in table3. Apparently, the smaller kernel size is, the less runtime will be. For larger kernel size, it will be is difficult to learn the robust feature correspondence.

Train Keyframe Interval

We conduct an ablation experiment to study the influence of the training keyframe interval. Table4 shows the results of training in different keyframe intervals. We achieve the best results of LWDN networks when the keyframe interval is set to 10 frames. And when the interval is too long or too short, the mAP will both be poor due to the difficulty in capturing motion information.

Table 4: The influence of different training keyframe interval

keyframe interval	4	6	8	10	12
mAP(%)	74.7	75.5	75.3	76.3	75.4

Table 5: The influence of different testing keyframe interval

keyframe interval	6	8	10	12	14
mAP(%)	76.7	76.5	76.3	75.6	75.2
runtime(ms)	60	53	48	45	44

Table 6: The influence of different memory gate

memory gate g	0.0	0.2	0.4	0.8	1.0
mAP(%)	75.2	75.3	75.6	76.0	76.3

Test Keyframe Interval

In this section, we conduct an ablation experiment to study the influence of the testing keyframe interval. The results are in table5. Apparently, larger interval benefits runtime. When the testing keyframe interval is 10 frames, we achieve 76.3% mAP, while maintaining 48 ms runtime. Specially, when the testing keyframe interval is 14 frames, the speed achieves 23fps while attaining 75.2% mAP.

Memory Gate

The memory gate g controls the component of memory feature as in Algorithm 1. The keyframe interval in training stage is 10 frames while 10 frames during testing stage in default. From the algorithm 1, it can be seen that g controls the available range of temporal information. When the g is set to 0.0, the memory feature consists solely of the previous keyframe feature; while g to 1.0 leads in more temporal information. Table 6 shows the mAP of different g setting. Apparently, larger g benefits performance, The involvement of long-range (Donahue et al. 2015) feature aggregation can help detection in longer series of low-quality frames as the conclusion in (Hetang et al. 2017).

Conclusion

We propose LWDN (Locally-Weighted Deformable Neighbors) for video object detection without utilizing time-consuming optical flow extraction. LWDN is conceptually motivated by Deformable CNN (Dai et al. 2017) and Non-local networks (Wang et al. 2018). And LWDN can latently align the high-level features from keyframes to non-keyframe or keyframes. Our end-to-end pipeline contains recursively updated memory feature for keyframes to keyframes message passing and a LWDN module for keyframes to non-keyframes or keyframes message passing. Extensive ablation experiments on VID dataset demonstrate the effective of our method over previous methods while achieving a better speed and accuracy trade-off.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grants 91646207, 61773377, and 61573352, and the Beijing Natural Science Foundation under Grants 4162064 and L172053.

References

Ba, J.; Mnih, V.; and Kavukcuoglu, K. 2014. Multiple object recognition with visual attention. *arXiv:1412.7755*.

Ballas, N.; Yao, L.; Pal, C.; and Courville, A. 2015. Delving deeper into convolutional networks for learning video representations. *arXiv:1511.06432*.

Brox, T., and Malik, J. 2011. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*.

Chen, L.-C.; Papandreou, G.; Schroff, F.; and Adam, H. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*.

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

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. *ICCV*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; and Brox, T. 2015. Flownet: Learning optical flow with convolutional networks. In *ICCV*.

Feichtenhofer, C.; Pinz, A.; Zisserman, A.; and Zisserman, A. 2017. Detect to track and track to detect. In *CVPR*.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2018. Hypergraph neural networks. *arXiv preprint arXiv:1809.09401*.

Gao, P.; Lu, P.; Li, H.; Li, S.; Li, Y.; Hoi, S.; and Wang, X. 2018. Question-guided hybrid convolution for visual question answering. *arXiv preprint arXiv:1808.02632*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Girshick, R. 2015. Fast r-cnn. In *ICCV*.

Gupta, S.; Girshick, R.; Arbeláez, P.; and Malik, J. 2014. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*.

Han, W.; Khorrami, P.; Paine, T. L.; Ramachandran, P.; Babaeizadeh, M.; Shi, H.; Li, J.; Yan, S.; and Huang, T. S. 2016. Seq-nms for video object detection. *arXiv:1602.08465*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*.

Hetang, C.; Qin, H.; Liu, S.; and Yan, J. 2017. Impression network for video object detection. *arXiv:1712.05896*.

Hu, P.; Wang, G.; Kong, X.; Kuen, J.; and Tan, Y.-P. 2018. Motion-guided cascaded refinement network for video object segmentation. In *CVPR*.

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

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Li, Y.; Shi, J.; and Lin, D. 2018. Low-latency video semantic segmentation. In *CVPR*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2018. Focal loss for dense object detection. *TPAMI*.

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

Mnih, V.; Heess, N.; Graves, A.; et al. 2014. Recurrent models of visual attention. In *NIPS*.

Nilsson, D., and Sminchisescu, C. 2016. Semantic video segmentation by gated recurrent flow propagation. *arXiv:1612.08871*.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*.

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

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*.

Shrivastava, A.; Gupta, A.; and Girshick, R. 2016. Training region-based object detectors with online hard example mining. In *CVPR*.

Uijlings, J. R.; Van De Sande, K. E.; Gevers, T.; and Smeulders, A. W. 2013. Selective search for object recognition. *IJCV*.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *CVPR*.

Xiao, F., and Lee, Y. J. 2017. Spatial-temporal memory networks for video object detection. *arXiv:1712.06317*.

Zhu, X.; Dai, J.; Yuan, L.; and Wei, Y. 2017a. Towards high performance video object detection. *arXiv:1711.11577*.

Zhu, X.; Wang, Y.; Dai, J.; Yuan, L.; and Wei, Y. 2017b. Flow-guided feature aggregation for video object detection. In *ICCV*.

Zhu, X.; Xiong, Y.; Dai, J.; Yuan, L.; and Wei, Y. 2017c. Deep feature flow for video recognition. In *CVPR*.