arXiv:1712.05896v1 [cs.CV] 16 Dec 2017

# Impression Network for Video Object Detection

Congrui Hetang* Hongwei Qin Shaohui Liu* Junjie Yan
SenseTime
{hetangcongrui, qinhongwei, liushaohui, yanjunjie}@sensetime.com

## Abstract

*Video object detection is more challenging compared to image object detection. Previous works proved that applying object detector frame by frame is not only slow but also inaccurate. Visual clues get weakened by defocus and motion blur, causing failure on corresponding frames. Multi-frame feature fusion methods proved effective in improving the accuracy, but they dramatically sacrifice the speed. Feature propagation based methods proved effective in improving the speed, but they sacrifice the accuracy. So is it possible to improve speed and performance simultaneously?*

*Inspired by how human utilize impression to recognize objects from blurry frames, we propose Impression Network that embodies a natural and efficient feature aggregation mechanism. In our framework, an impression feature is established by iteratively absorbing sparsely extracted frame features. The impression feature is propagated all the way down the video, helping enhance features of low-quality frames. This impression mechanism makes it possible to perform long-range multi-frame feature fusion among sparse keyframes with minimal overhead. It significantly improves per-frame detection baseline on ImageNet VID while being 3 times faster (20 fps). We hope Impression Network can provide a new perspective on video feature enhancement. Code will be made available.*

## 1. Introduction

Fast and accurate video object detection methods are highly valuable in vast number of scenarios. Single-image object detectors like Faster R-CNN [22] and R-FCN [3] have achieved excellent accuracy on still images, so it is natural to apply them to video tasks. One intuitive way is applying them frame by frame on videos, but this is far from optimal. First, image detectors typically involve a heavy feature network like ResNet-101 [11], which runs rather slow (5fps) even on GPUs. This hampers their potential in real-time applications like autonomous driving and video surveillance. Second, single-image detectors are vulnerable to the common image degeneration problem in videos [33]. As shown in Figure 2, frames may suffer from defocus, motion blur, strange object positions and all sorts of deteriorations, leaving too weak visual clues for successful detections. The two problems make object detection in videos challenging.

Feature-level methods [6, 34, 33, 26] have addressed either one of the two problems. These methods treat single-image recognition pipeline as two stages: 1. the image is passed through a general feature network; 2. the result is then generated by a task-specific sub-network. When transferring image detectors to videos, feature-level methods seek ways to improve the feature stage, while the task network remains unchanged. The task-independence makes feature-level methods versatile and conceptually simple. To improve speed, feature-level methods reuse sparsely sampled deep features in the first stage [34, 26], because nearby video frames provide redundant information. This saves the expensive feature network inference and boosts speed to real-time level, but sacrifices accuracy. On the other hand, accuracy can be improved by multi-frame feature aggregation [33, 21]. This enables successful detection on low-quality frames, but the aggregation cost can be huge thus further slows down the framework. In this work, we combine the advantages of both tracks. We present a new feature-level framework, which runs at real-time speed and outperforms per-frame detection baseline.

Our method, called Impression Network, is inspired by the way how human understand videos. When there comes a new frame, humans do not forget previous frames. Instead, the impression is accumulated along the video, which helps us understand degenerated frames with limited visual clue. This mechanism is embodied in our method to enhance frame feature and improve accuracy. Moreover, we combine it with sparse keyframe feature extraction to obtain real-time inference speed. The pipeline of our method is shown in Figure 1.

To address the redundancy and improve speed, we split a video into segments of equal length. For each segment, only one keyframe is selected for deep feature extraction.

*This work is done when Congrui Hetang and Shaohui Liu are interns at SenseTime

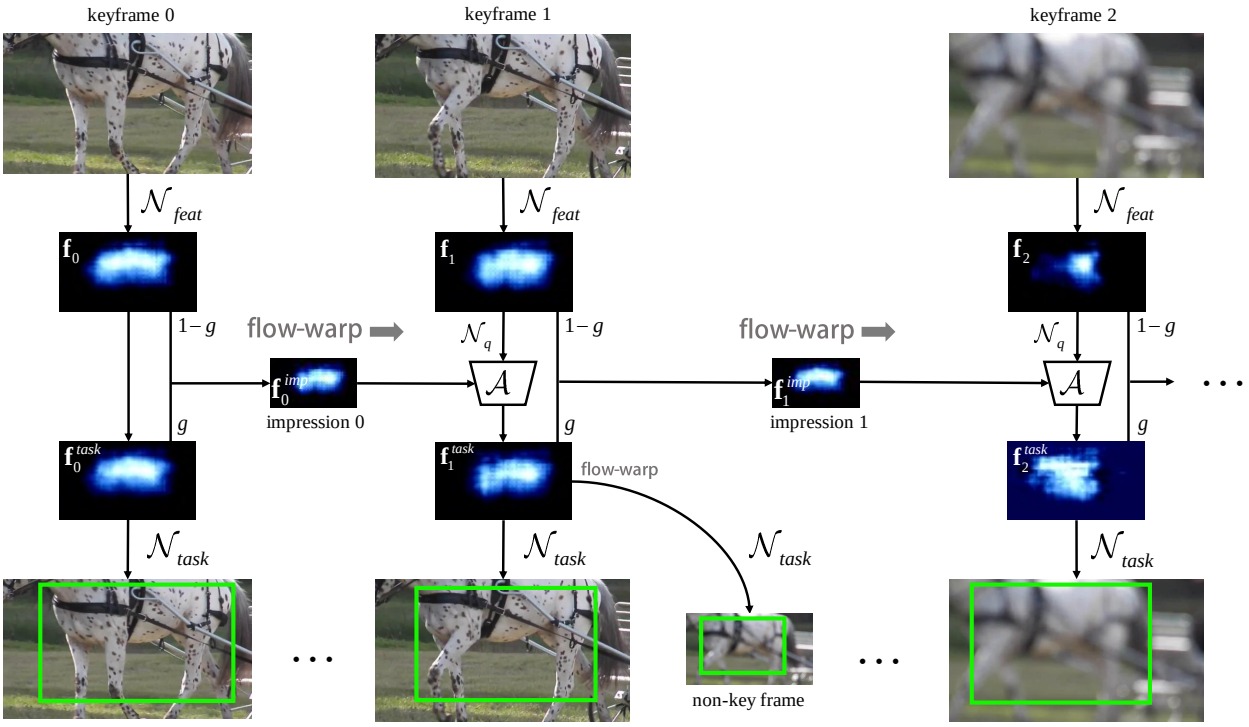


Figure 1: Impression Network inference pipeline. Only keyframes of the first 3 segments are shown. The $581^{st}$ channel (sensitive to horse) of the top feature map is visualized. The detection at keyframe 2 should have failed due to defocus (Figure 5), but the impression feature brought information from previous frames and enhanced $\mathbf{f}_2$, thus $\mathcal{N}_{task}$ was still able to predict correctly.

With flow-guided feature propagation [34, 5], the key feature is reused by non-key frames to generate detection results. Based on this, we adopt our Impression mechanism to perform multi-frame feature fusion. When a key feature is extracted, it not only goes to task network, but is also absorbed by a impression feature. The impression feature is then propagated down to the next keyframe. The task feature for the next keyframe is a weighted combination of its own feature and the impression feature, and the impression feature is updated by absorbing the feature of that frame. This process keeps going on along the whole video. In this framework, the impression feature accumulates high-quality video object information and is propagated all the way down, helping enhance incoming key features if the frames get deteriorated. It improves the overall quality of task features, thus increases detection accuracy.

The Impression mechanism also contributes to the speed. With the iterative aggregation policy, it minimized the cost of feature fusion. Previous work [33] has proved that, video frame features should be spatially aligned with flow-guided warping before aggregation, while flow computation is not negligible. Intuitive way requires one flow estimation for each frame being aggregated, while Impression Network only needs one extra flow estimation for adjacent segments, being much more efficient.

Without bells and whistles, Impression Network surpasses state-of-the-art image detectors on ImageNet VID [24] dataset. It's three times faster (20 fps) and significantly more accurate. We hope Impression Network can provide a new perspective on feature aggregation in video tasks.

Code will be released to facilitate future research.

## 2. Related Work

**Feature Reuse in Video Recognition:** As shown by earlier analysis [30, 35, 15, 32, 27], consecutive video frames are highly similar, as well as their high-level convolutional features. This suggests that video sequences feature an inherent redundancy, which can be exploited to reduce time cost. In single image detectors [8, 10, 22, 7, 3], the heavy feature network (encoder) is much more costly than the task subnetwork (decoder). Hence, when transplanting image detectors to videos, speed can by greatly improved by reusing the deep features of frames. Clockwork Convnets [26] ex-

ploit the different evolve speed of features at different levels. By updating low and high level convolutional features at different frequency, it partially avoids redundant feature computation. It makes the network 1.3 times faster, while sacrifices accuracy by $1\% \sim 4\%$ due to the lack of end to end training. Deep Feature Flow [34] is another successful feature-level acceleration method. It cheaply propagates the top feature of sparse keyframe to other frames, achieving a significant speed-up ratio (from 5 fps to 20 fps). Deep Feature Flow requires motion estimation like optical flow [12, 2, 29, 23, 5, 13] to propagate features, where error is introduced and therefore brings a minor accuracy drop ($\sim 1\%$). Impression Network inherits the idea of Deep Feature Flow, but also utilizes temporal information to enhance the shared features. It's not only faster than per-frame baseline, but also more accurate.

**Exploiting Temporal Information in Video Tasks:** Applying state-of-the-art still image detectors frame by frame on videos does not provide optimal result [33]. This is mainly due to the low-quality images in videos. Single image detectors are vulnerable to deteriorated images because they are restricted to the frame they are looking at, while ignoring the ample temporal information from other frames in the video. Temporal feature aggregation [17, 20, 25, 28, 18, 1, 31] provides a way to utilize such information. Flow-Guided Feature Aggregation(FGFA) [33] aims at enhancing frame features by aggregating all frame features in a consecutive neighborhood. The aggregation weight is learned through end-to-end training. FGFA boosts video detection accuracy to a new level (from $74.0\%$ to $76.3\%$), yet it is three-times slower than per-frame solution (1.3 fps). This is caused by the aggregation cost. For each frame in the fusion range, FGFA requires one optical flow computation to spatially align it with the target frame, which costs even more time than the feature network. Additionally, since neighboring frames are highly similar, the exhaustive dense aggregation leads to extra redundancy. Impression Network fuses features in an iterative manner, where only one flow estimation is needed for every new keyframe. Moreover, the sparse feature sampling reduces the amount of replicated information.

## 3. Impression Network

### 3.1. Impression Network Inference

Given a video, our task is to generate detection results for all its frames $\mathbf{I}_k$, $i = 0, \ldots, N$. To avoid redundant feature computation, we split the frame sequence into segments of equal length $l$. In each segment $\mathbf{S}_k = \{\mathbf{I}_{kl}, \mathbf{I}_{kl+1}, ..., \mathbf{I}_{(k+1)l-1}\}$, only one frame $\mathbf{I}_k^{\text{key}}$ (by default we take the central frame $\mathbf{I}_{kl+\lfloor l/2 \rfloor}$) is selected for feature extraction via the feature network $\mathcal{N}_{feat}$. The key feature


Figure 2: Examples of deteriorated frames in videos.

**Algorithm 1** Inference algorithm of Impression Network for video object detection.

**input**: video frames $\{\mathbf{I}\}$, segment length $l$
**for** $k = 0$ **to** $N$ **do**
  $\mathbf{f}_k = \mathcal{N}_{feat}(\mathbf{I}_k^{\text{key}})$ ▷ extract keyframe feature
  **if** $k = 0$ **then** ▷ first keyframe
    $\mathbf{f}_k^{\text{imp}} = \mathbf{f}_k$ ▷ initialize impression feature
    $\mathbf{f}_k^{\text{task}} = \mathbf{f}_k$
  **else**
    $1 - w_k, w_k = \text{softmax}(\mathcal{N}_q(\mathbf{f}_{k-1}^{\text{imp}\,\prime}), \mathcal{N}_q(\mathbf{f}_k))$
    $\mathbf{f}_k^{\text{task}} = (1 - w_k) \cdot \mathbf{f}_{k-1}^{\text{imp}\,\prime} + w_k \cdot \mathbf{f}_k$ ▷ adaptive weighting
    $\mathbf{f}_k^{\text{imp}} = (1 - g) \cdot \mathbf{f}_k + g \cdot \mathbf{f}_k^{\text{task}}$ ▷ update impression feature
  **end if**
  **for** $j = 0$ **to** $l - 1$ **do** ▷ feature propagation
    $\mathbf{f}_j^k = \mathcal{W}(\mathbf{f}_k^{\text{task}}, \mathcal{F}(\mathbf{I}_j^k, \mathbf{I}_k^{\text{key}}))$ ▷ flow-guided warp
    $y_j^k = \mathcal{N}_{task}(\mathbf{f}_j^k)$ ▷ detection result
  **end for**
**end for**
**output**: detection results $\{y\}$

is propagated to remaining frames with flow-guided warping, where the flow field is computed by a light-weight flow network, following the practice of Deep Feature Flow [34]. Features of all frames are then fed into task network $\mathcal{N}_{task}$ to generate detection results.

In such framework, we use impression mechanism to exploit long-range, cross-segment temporal information. The inference phase of Impression Network is illustrated in Figure 1. Each segment $\mathbf{S}_k$ generates three features: $\mathbf{f}_k$ calculated by passing $\mathbf{I}_k^{\text{key}}$ through $\mathcal{N}_{feat}$, $\mathbf{f}_k^{\text{task}}$ shared by all frames in the segment for detection sub-network and $\mathbf{f}_k^{\text{imp}}$, the impression feature containing long-term temporal information. For the first segment $\mathbf{S}_0$, $\mathbf{f}_0^{\text{imp}}$ and $\mathbf{f}_0^{\text{task}}$ are identical to $\mathbf{f}_0$. For $\mathbf{S}_1$, $\mathbf{f}_1^{\text{task}}$ is a weighted combination of $\mathbf{f}_0^{\text{imp}}$ and $\mathbf{f}_1$. The aggregation unit $\mathcal{A}$ uses a tiny FCN $\mathcal{N}_q$ to generate position-wise weight maps. Generally, larger weights are assigned to the feature with better quality. This is con-

cluded as

$$\mathbf{f}_0^{\text{imp}}, \mathbf{f}_0^{\text{task}} = \mathbf{f}_0, \tag{1}$$

$$1 - w_1, w_1 = \text{softmax}(\mathcal{N}_q(\mathbf{f}_0^{\text{imp}'}), \mathcal{N}_q(\mathbf{f}_1)), \tag{2}$$

$$\mathbf{f}_1^{\text{task}} = (1 - w_1) \cdot \mathbf{f}_0^{\text{imp}'} + w_1 \cdot \mathbf{f}_1. \tag{3}$$

Notice that such quality is not a handcrafted metric, instead it's learned by end-to-end training to minimize task loss. We observe that when $\mathbf{I}_k^{\text{key}}$ is deteriorated by motion blur or defocus, $\mathbf{f}_k$ gets lower quality score, as shown in Figure 4. Also notice that the aggregation of cross-segment features is not simply adding them up. Former practice [33] shows that due to spatial misalignment in video frames, naive weighted mean yields worse results. Here we use flow-guided aggregation. Specifically, we first calculate the flow field of $\mathbf{I}_0^{\text{key}}$ and $\mathbf{I}_1^{\text{key}}$, then perform spatial warping accordingly on $\mathbf{f}_0^{\text{imp}}$ to align it with $\mathbf{I}_1^{\text{key}}$, getting $\mathbf{f}_0^{\text{imp}'}$; the fusion is then done with $\mathbf{f}_0^{\text{imp}'}$ and $\mathbf{f}_1$ to generate $\mathbf{f}_1^{\text{task}}$. $\mathbf{f}_1$ and $\mathbf{f}_1^{\text{task}}$ are then mingled to get $\mathbf{f}_1^{\text{imp}}$:

$$\mathbf{f}_1^{\text{imp}} = (1 - g) \cdot \mathbf{f}_1 + g \cdot \mathbf{f}_1^{\text{task}}. \tag{4}$$

Here a constant factor $g$ controls the contribution of $\mathbf{f}_1^{\text{task}}$. $g$ serves as a gate to control the memory of the framework (detailed in Figure 6). If set $g$ to 0, $\mathbf{f}_k^{\text{imp}}$ will only contain information of $\mathbf{I}_{k-1}^{key}$. The procedure keeps going on until all frames in a video get processed.

By iteratively absorbing every keyframe feature, the impression feature contains visual information in a large time span. The weighted aggregation of $\mathbf{f}_k$ and $\mathbf{f}_{k-1}^{\text{imp}}$ can be seen as a balancing between memory and new information, depending on the quality of the new incoming keyframe. When the new keyframe gets deteriorated, the impression feature compensate for the subsequent weak feature, helping infer bounding box and class information through low-level visual clue such as color distribution. On the other hand, the impression feature also keeps getting updated. Since sharp and clear frames get higher scores, they contribute more to an effective impression. Compared to exhaustively aggregating all nearby features in a fixed range for every frame, our framework is more natural and elegant. The whole process is summarized in Algorithm 1.

### 3.2. Impression Network Training

The training procedure of Impression Network is rather simple. With video data provided, a standard single-image object detection pipeline can be transfered to video tasks with slight modifications. The end-to-end training framework is illustrated in Figure 3.

During training, each data batch contains three images $\mathbf{I}_{k+d_0}, \mathbf{I}_k, \mathbf{I}_{k+d_1}$ from a same video sequence. $d_0$ and $d_1$

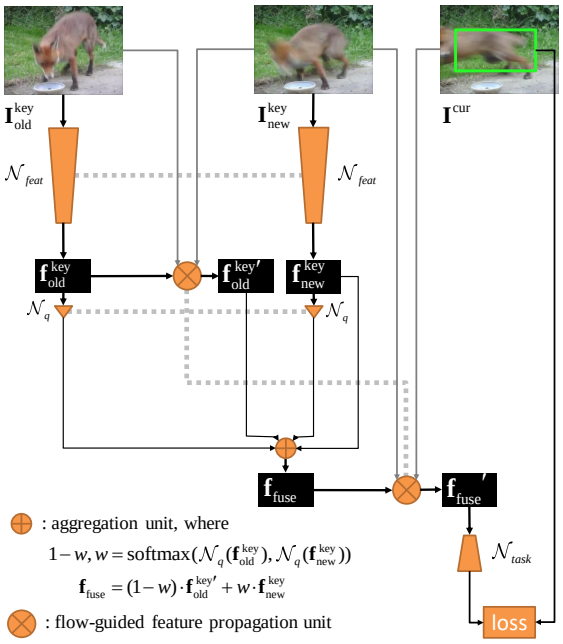


Figure 3: Training framework of Impression Network. The data flow is marked by solid lines. Components linked with dashed lines share weights. The working condition of inference stage is simulated with video frame triplets. All components are optimized end-to-end.

are random offsets whose ranges are controlled by segment length $l$. Typically, $d_0$ lies in $[-l, -0.5l]$, while $d_1$ falls into $[-0.5l, 0.5l]$. This setting is coherent with the inference phase, as $\mathbf{I}_k$ represents an arbitrary frame from segment $\mathbf{S}_n$, $\mathbf{I}_{k+d_1}$ for keyframe of current segment $\mathbf{I}_n^{\text{key}}$, while $\mathbf{I}_{k+d_0}$ stands for the previous keyframe. For simplicity, the three images are dubbed as $\{\mathbf{I}_{\text{old}}^{\text{key}}, \mathbf{I}^{\text{cur}}, \mathbf{I}_{\text{new}}^{\text{key}}\}$. The ground-truth at $\mathbf{I}^{\text{cur}}$ is provided as label.

In each iteration, first, $\mathcal{N}_{feat}$ is applied on $\{\mathbf{I}_{\text{old}}^{\text{key}}, \mathbf{I}_{\text{new}}^{\text{key}}\}$ to get their deep features $\{\mathbf{f}_{\text{old}}^{\text{key}}, \mathbf{f}_{\text{new}}^{\text{key}}\}$. Then, image pairs $\{\mathbf{I}_{\text{new}}^{\text{key}}, \mathbf{I}_{\text{old}}^{\text{key}}\}$ and $\{\mathbf{I}^{\text{cur}}, \mathbf{I}_{\text{new}}^{\text{key}}\}$ are fed into the flow network, yielding optical flow fields $\mathbf{M}_{\text{old}\to\text{new}}$ and $\mathbf{M}_{\text{new}\to\text{cur}}$, respectively. Flow-guided warping unit then use $\mathbf{M}_{\text{old}\to\text{new}}$ to propagate $\mathbf{f}_{\text{old}}^{\text{key}}$ to align with $\mathbf{f}_{\text{new}}^{\text{key}}$. We denote the warped old keyframe feature as $\mathbf{f}_{\text{old}}^{\text{key}'}$. The aggregation unit weights and fuses $\{\mathbf{f}_{\text{old}}^{\text{key}'}, \mathbf{f}_{\text{new}}^{\text{key}}\}$, generating $\mathbf{f}_{\text{fuse}}$. $\mathbf{f}_{\text{old}}^{\text{key}}$ in training corresponds to the impression feature in inference. This is an approximation since it only contains information of one previous keyframe. Finally, $\mathbf{f}_{\text{fuse}}$ is warped to $\mathbf{I}^{\text{cur}}$ according to $\mathbf{M}_{\text{new}\to\text{cur}}$ to get $\mathbf{f}_{\text{fuse}}'$, the task feature for a standard detection sub-network. Since all the components are differentiable, the detection loss propagates all the way back to jointly fine-tune $\mathcal{N}_{task}$, $\mathcal{N}_{feat}$, flow network and feature ag-

gregation unit, optimizing task performance. Notice that single-image datasets can be fully exploited in this framework, in which case the three images are all the same.

### 3.3. Module Design

**Feature Network:** We use ResNet-101 pretrained for ImageNet classification. The fully connected layers are removed. For denser feature map, feature stride is reduced from 32 to 16. Specifically, the stride of the last block is modified from 2 to 1. To maintain receptive field, A dilation of 2 is applied to convolution layers with kernel size greater than 1. A 1024-channel $3 \times 3$ convolution layer (randomly initialized) is appended to reduce feature dimension.

**Flow-Guided Feature Propagation:** Before aggregation, we spatially align frame features by flow-guided warping. Optical flow field is calculated first to obtain pixel-level motion path, then reference feature is warped to target frame with bilinear sampling. The procedure is defined as

$$\mathbf{f}^{\text{ref}'} = \mathcal{W}(\mathbf{f}^{\text{ref}}, \mathcal{F}(\mathbf{I}^{\text{cur}}, \mathbf{I}^{\text{ref}})) \cdot \mathbf{S}$$

where $\mathbf{I}^{\text{cur}}$ and $\mathbf{I}^{\text{ref}}$ denotes target frame and reference frame respectively, $\mathbf{f}^{\text{ref}}$ is the deep feature of reference frame, $\mathbf{f}^{\text{ref}'}$ denotes reference feature warped to target frame, $\mathcal{F}$ stands for flow estimation function, W denotes the bilinear sampler, and $\mathbf{S}$ is a predicted position-wise scale map to refine warped feature. We adopt the state-of-the-art CNN-based FlowNet [5, 13] for optical flow computation. Specifically, we use FlowNet-S [5]. The flow network is pretrained on FlyingChairs dataset. The scale map has equal channel dimension with task features, and is predicted with flow field in parallel through an additional $1 \times 1$ convolution layer attached to the top of FlowNet-S. The new layer is initialized with weights of all zeros and fixed biases of all ones. The implementation of bilinear sampling unit has been well described in [14, 4, 34]. It is fully differentiable.

**Aggregation Unit:** The aggregation weights of features are generated by a quality estimation network $\mathcal{N}_q$. It has three randomly initialized layers: a $3 \times 3 \times 256$ convolution, a $1 \times 1 \times 16$ convolution and a $1 \times 1 \times 1$ convolution. The output is a position-wise raw score map which will be applied on each channel of task feature. Raw score maps of different features are normalized by softmax function to sum up to one. We then multiply the score maps with features and sum them up to obtain the fused feature as Eq. 3.

**Detection Network:** We use the state-of-the-art R-FCN as detection sub-network. RPN and R-FCN are attached to the 1024-channel convolution of the feature network, using the first and second 512 channels respectively. RPN uses 9 anchors and generates 300 proposals for each image. We use $7 \times 7$ groups position-sensitive score maps for R-FCN.


Figure 4: Examples of frames assigned with different aggregation weights. The white number is the spatially averaged pixel-wise weight $w_k$ in algorithm 1. Consistent with intuition, the scoring FCN $\mathcal{N}_q$ assigns larger weights to sharp and clear frames.

### 3.4. Runtime Complexity Analysis

The ratio of inference time of our method to that of per-frame evaluation is:

$$r = \frac{O(\mathcal{A}) + l \times (O(\mathcal{W}) + O(\mathcal{F}) + O(\mathcal{N}_{task})) + O(\mathcal{N}_{feat})}{l \times (O(\mathcal{N}_{feat}) + O(\mathcal{N}_{task}))}$$

In each segment of length $l$ , Impression Network requires: 1. $l$ flow warping $(O(\mathcal{W}) + O(\mathcal{F}))$ in total, one for impression feature propagation and $l - 1$ for non-key frame detection; 2. One feature fusion operation $(O(\mathcal{A}))$; 3. One feature network inference for keyframe feature; 4. $l$ detection subnetwork inference. In comparison, per-frame solution takes $l$ $\mathcal{N}_{feat}$ and $l$ $\mathcal{N}_{task}$ inference. Notice that compared to $\mathcal{N}_{feat}$ (Resnet-101 in our practice) and FlowNet, the complexity of $\mathcal{A}$, $\mathcal{W}$ and $\mathcal{N}_{task}$ are negligible. So the ratio can be approximated as:

$$r \approx \frac{O(\mathcal{F})}{O(\mathcal{N}_{feat})} + \frac{1}{l}$$

In practice, the flow network is times smaller than Resnet-101, while $l$ is large ($\geq$ 10) to reduce redundancy. This suggests that unlike existing feature aggregation method like FGFA, Impression Network can perform multi-frame feature fusion while maintaining a noticeable speedup over per-frame solution.

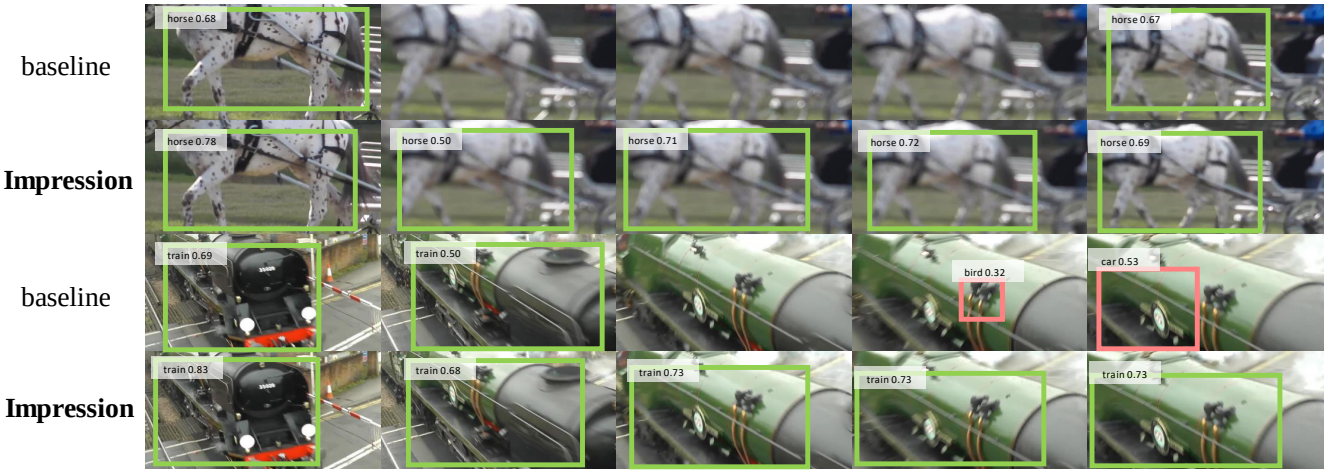


Figure 5: Examples where Impression Network outperforms per-frame baseline (standard ResNet-101 R-FCN). Green boxes are true positives while red ones are false positives.

# 4. Experiments

## 4.1. Experiment Setup

**ImageNet VID dataset [24]:** It is a large-scale video object detection dataset. There are 3862, 555 and 937 snippets with frame rates of 25 and 30 in training, validation and test sets, respectively. All the video snippets are fully-annotated. Imagenet VID dataset has 30 object categories, which is a subset of the Imagenet DET dataset. In our experiments, following the practice in [16, 19, 34, 33], model are trained on the training set, while evaluations are done on the validation set with the standard mean average precision (mAP) metric.

**Implementation Details:** Our training set consists of the full ImageNet VID train set, together with images from ImageNet DET train set. Only the same 30 categories are used. As mentioned before, each training batch contains three images. If sampled from DET, all images are same. In both training and testing, images are resized to have the shorter side of 600 and 300 pixels for the feature network and the flow network, respectively. The whole framework is trained end to end with SGD, where 120K iterations are performed on 8 GPUs. The learning rate is $10^{-3}$ for the first 70K iterations, then reduced to $10^{-4}$ for the remaining 50K iterations. For clear comparison, no bells-and-whistles like multi-scale training and box-level post-processing are used. Inference time is measured on a Nvidia GTX 1060 GPU.

## 4.2. Ablation Study

**Architecture Design:** Table 1 summarizes main experiment results. It shows a comparison of single-frame baseline, Impression Network and its variants.

| methods | (a) | (b) | (c) | **(d)** | (e) |
|---|---|---|---|---|---|
| sparse feature? | | ✓ | ✓ | ✓ | ✓ |
| impression? | | | ✓ | ✓ | ✓ |
| quality-aware? | | | | ✓ | ✓ |
| end-to-end? | ✓ | ✓ | ✓ | ✓ | |
| mAP (%) | 74.2 | 73.6 | 75.2 | **75.5** | 70.3 |
| runtime (ms) | 156 | 48 | 50 | **50** | 50 |

Table 1: Accuracy and runtime of different approaches.

*Method (a)* is the standard ResNet-101 R-FCN applied frame by frame to videos. The accuracy is close to the 73.9% mAP reported in [34], which shows its validity as a strong baseline for our evaluations. The runtime is a little bit faster, probably due to differences in implementation environment. The ≈6fps inference speed is insufficient for real-time applications, where typically a speed of ≥15fps is required.

*Method (b)* is a variant of *Method (a)* with sparse feature extraction. In this approach, videos are divided into segments of $l$ frames. Only one keyframe in each segment will be passed through the feature network for feature extraction. That feature is then propagated to other frames with optical flow. Finally, the detection sub-network generates results for every frame. The structure is identical to a Deep Feature Flow framework for video object detection [34]. Specifically, $l$ is set to 10 for all experiments in this table. We select the 5th frame as keyframe, because this minimizes average feature propagation distance, thus reduces the error introduced and improves accuracy (explained later). Compared to per-frame evaluation, there's a minor accuracy drop of 0.6%, mainly because of lessened information, as well

as errors in flow-guided feature warping. However, the inference speed remarkably increases to 21fps, proving that sparse feature extraction is an efficient way to trade accuracy for speed.

*Method (c)* is a degenerated version of Impression Network. Keyframe features are iteratively fused to generate the impression feature, but without quality-aware weighting unit. The weights in Eq. 3 are naively fixed to 0.5. For all experiments here, the memory gate $g$ in Eq. 4 is set to 1.0. With information of previous frames fused into current task feature, mAP increases for 1.0% over per-frame baseline. Notice that sparse feature extraction is still enabled here, which proves that 1.the computational redundancy of per-frame evaluation is huge; 2.such redundancy is not necessary for higher accuracy. Due to the one additional flow estimation for each segment, the framework slows down a little bit, yet still runs at a real-time-level 20fps.

*Method (d)* is the proposed Impression Network. Here the aggregation unit uses the tiny FCN to generate position-wise weights. Through end-to-end training, the sub-network learns to assign smaller weights to features of deteriorated frames, as shown in Figure 4. Experiment on ImageNet VID validation set shows the $w_k$ in algorithm 1 obeys a normal distribution of $\mathcal{N}(0.5, 0.016^2)$. Quality-aware weighting brings another 0.3% mAP improvement, mainly because of the increment of valid information. Overall, Impression Network increases mAP by 1.3% to 75.5%, comparable to exhaustive feature aggregation method [33], while significantly faster, running at 20fps. Impression Network shows that, if redundancy and temporal information are properly handled, the speed and accuracy of video object detection can actually be simultaneously improved. Examples are shown in Figure 5.

*Method (e)* is Impression Network without end-to-end training. The feature network is trained in single-image detection framework, same to that in *Method (a)*. The flow network is the off-the-shelf FlyingChairs pretrained FlowNet-S [5]. Only the weighting and detection sub-networks learn during training. This clearly worsen the mAP, showing the importance of end-to-end optimization.

**The Influence of Memory Gate:** As shown in Eq. 4, the memory gate $g$ controls the component of impression features. Here we study its influence on mAP. Experiment settings are the same as *Method (d)* in Table 1, except that $g$ varies from 0.0 to 1.0. Figure 6 shows the average contribution of previous keyframes to current detection at different $g$ values. It can be seen that $g$ controls the available range of temporal information. When set to 0.0, the impression feature consists solely of the previous key feature, just like how the framework is trained; while setting $g$ to 1.0 leads in more temporal information. Figure 7 shows the mAP of different $g$ setting. Apparently, larger $g$ benefits accuracy. The

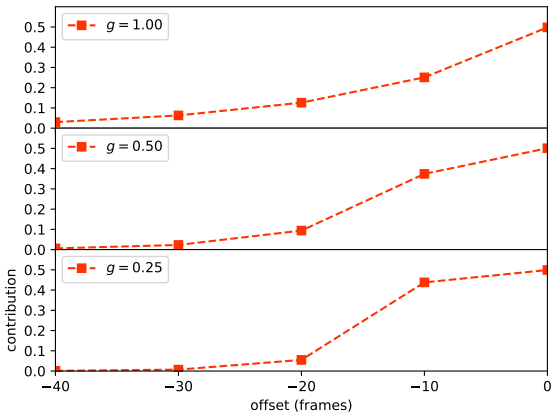


Figure 6: Averaged contribution of previous keyframes to current detection at different memory gate $g$. When $g$ is 1.0, the contribution smoothly decreases as offset grows. As $g$ decreases, the impression gets increasingly occupied by the nearest keyframe, while the contribution of earlier ones rapidly shrinks to 0.

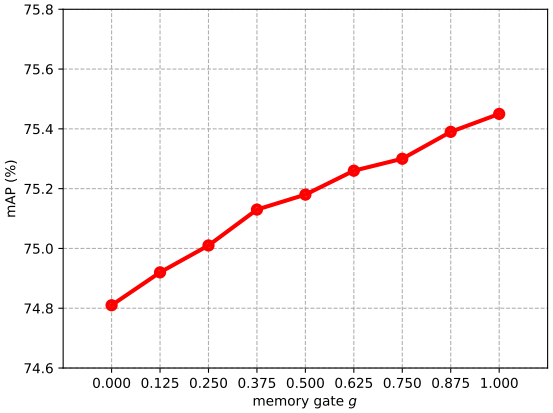


Figure 7: mAP at different $g$ values. Although it's not exactly how the network is trained, enabling long-range aggregation do brings noticeable improvement.

involvement of long-range feature aggregation may help detection in longer series of low-quality frames.

**Different Keyframe Selection:** In aforementioned experiments, to reduce error, we select the central frame of each segment as keyframe. Here we explain this and compare different keyframe scheduling. Flow-guided feature warping introduces error, and as shown in [34], the error has positive correlation with propagation distance. This is because that larger displacement increases the difficulty of pixel-level matching. Hence, we take average feature prop-

| keyframe id | $\bar{d}$ (frames) | mAP (%) |
|---|---|---|
| 0 | 5.5 | 73.9 |
| 1 | 4.7 | 74.4 |
| 2 | 4.1 | 74.9 |
| 3 | 3.7 | 75.2 |
| 4 | 3.5 | 75.5 |
| 5 | 3.5 | 75.5 |

Table 2: Average propagation distance and mAP at different keyframe selections. Other settings are same as *Method (d)* in Table 1. Because of the symmetry, only id 0-5 is shown.

| method | mAP (%) | runtime (ms) |
|---|---|---|
| FGFA | 76.3 | 733 |
| FGFA-fast | 75.3 | 356 |
| Impression Network | 75.5 | 50 |

Table 3: Comparison with aggregation-based method FGFA and its faster variant. Settings are same as *Method (d)* in Table 1.

agation distance $\bar{d}$ as a metric for flow error, and seek the way to minimize it. $\bar{d}$ is calculated as:

$$\bar{d} = \begin{cases} \frac{\sum_{d=1}^{l-1} d + l}{l}, & k = 0, l-1 \\ \frac{\sum_{d=1}^{k} d + \sum_{d=1}^{l-1-k} d + l}{l}, & 0 < k < l-1 \end{cases}$$

where $d$ is propagation distance, $k$ is the id of keyframe, and $l$ is segment length. Key feature needs to be propagated to non-key frames, and there's also an impression feature propagation of distance $l$. Apparently there's an optimal $k$ to minimize $\bar{d}$:

$$\arg\min_k(\bar{d}) = \frac{l-1}{2} \tag{5}$$

which shows that the central frame is the best. Table 2 shows mAPs at different keyframe selections, coherent with our assumption. Notice that selecting the first frame enables strict real-time inference, while selecting the central frame brings a slight latency of $l/2$ frames. This can be traded-off according to application needs.

### 4.3. Compare with Other Feature-Level Methods

We compare Impression Network with other feature-level video object detection methods. In Figure 8, we compare the speed-accuracy curve of Impression Network and Deep Feature Flow [34]. Per-frame baseline is also marked. Segment length $l$ varies from 1 to 20. Apparently, Impression Network is more accurate than per-frame solution even in high-speed zone. Similar to Deep Feature Flow, it also offers a smooth accuracy-speed trade-off as $l$ varies. The accuracy drops a little when $l$ gets close to 1, which is reasonable because Impression Network is trained for aggregating sparse frame features. Dense sampling limits aggregation range and result in a less useful impression.

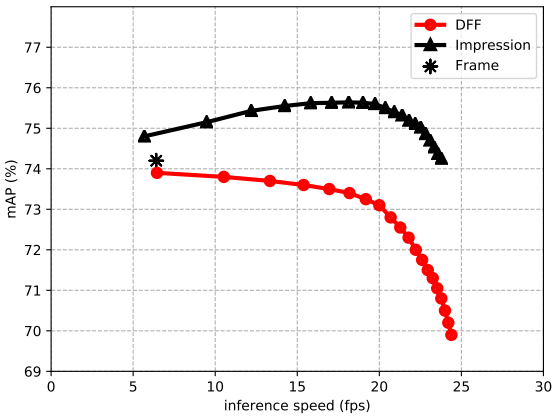


Figure 8: Comparing speed-accuracy curves of Deep Feature Flow (DFF) and Impression Network (Impression). Both using ResNet-101 as feature network and FlowNet-S as flow network.

Table 3 compares Impression Network with Flow-Guided Feature Aggregation and its faster variant. Both are described in [33]. FGFA is the standard version with a fusion radius of 10, and FGFA-fast is the accelerated version. It only calculates flow fields for adjacent frames, and composite them for non-adjacent pairs. This comparison shows that the accuracy of Impression Network is on par with the best aggregation-based method, yet being much more efficient.

## 5. Conclusion and Future Work

This work presents a fast and accurate feature-level method for video object detection. The proposed Impression mechanism explores a novel scheme for feature aggregation in videos. Since Impression Network works at feature stage, it's complementary to existing box-level post-processing methods like Seq-NMS [9]. For now we use FlowNet-S [5] to guide feature propagation for clear comparison, while more efficient flow algorithms [13] exist and can surely benefit our method. We use fixed segment length for simplicity, while a adaptively varying length may schedule computation more reasonably. Moreover, as a feature-level method, Impression Network inherits the task-independence, and has the potential to tackle image degeneration problem in other video tasks.

## References


[1] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 3

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004. 3

[3] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1, 2

[4] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017. 5

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 1, 3, 5, 7, 8

[6] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. *arXiv preprint arXiv:1708.03088*, 2017. 1

[7] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[9] W. Han, P. Khorrami, T. L. 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. 8

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[12] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 3

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016. 3, 5, 8

[14] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 5

[15] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3852–3861, 2016. 2

[16] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 6

[17] A. Kar, N. Rai, K. Sikka, and G. Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. *arXiv preprint arXiv:1611.08240*, 2016. 3

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 3

[19] 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 *European Conference on Computer Vision*, pages 68–83. Springer, 2016. 6

[20] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 2017. 3

[21] Y. Liu, J. Yan, and W. Ouyang. Quality aware network for set to set recognition. *arXiv preprint arXiv:1704.03373*, 2017. 1

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

[23] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015. 3

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 6

[25] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015. 3

[26] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *Computer Vision–ECCV 2016 Workshops*, pages 852–868. Springer, 2016. 1, 2

[27] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan. Dl-sfa: deeply-learned slow feature analysis for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2632, 2014. 2

[28] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4597–4605, 2015. 3

[29] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 3

[30] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 2

[31] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 3

[32] Z. Zhang and D. Tao. Slow feature analysis for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):436–450, 2012. 2

[33] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. *arXiv preprint arXiv:1703.10025*, 2017. 1, 2, 3, 4, 6, 7, 8

[34] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. *arXiv preprint arXiv:1611.07715*, 2016. 1, 3, 5, 6, 7, 8

[35] W. Zou, S. Zhu, K. Yu, and A. Y. Ng. Deep learning of invariant features via simulated fixations in video. In *Advances in neural information processing systems*, pages 3203–3211, 2012. 2