Flow-Guided Feature Aggregation for Video Object Detection

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

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

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

Abstract

Extending state-of-the-art object detectors from image to video is challenging. The accuracy of detection suffers from degenerated object appearances in videos, e.g.*, motion blur, video defocus, rare poses, etc. Existing work attempts to exploit temporal information on box level, but such methods are not trained end-to-end. We present flow-guided feature aggregation, an accurate and end-to-end learning framework for video object detection. It leverages temporal coherence on feature level instead. It improves the per-frame features by aggregation of nearby features along the motion paths, and thus improves the video recognition accuracy. Our method significantly improves upon strong single-frame baselines in ImageNet VID [33], especially for more challenging fast moving objects. Our framework is principled, and on par with the best engineered systems winning the ImageNet VID challenges 2016, without additional bells-and-whistles. The code would be released.*

1. Introduction

Recent years have witnessed significant progress in object detection [11]. State-of-the-art methods share a similar two-stage structure. Deep Convolutional Neural Networks (CNNs) [22, 36, 40, 14] are firstly applied to generate a set of feature maps over the whole input image. A shallow detection-specific network [13, 10, 30, 26, 5] then generates the detection results from the feature maps.

These methods achieve excellent results in still images. However, directly applying them for video object detection is challenging. The recognition accuracy suffers from deteriorated object appearances in videos that are seldom observed in still images, such as motion blur, video defocus, rare poses, etc (See an example in Figure 1 and more in Figure 2). As quantified in experiments, a state-of-the-art still-image object detector (R-FCN [5] + ResNet-101 [14]) deteriorates remarkably for fast moving objects (Table 1 (a)).

*This work is done when Xizhou Zhu and Yujie Wang are interns at Microsoft Research Asia

Nevertheless, the video has rich information about the same object instance, usually observed in multiple "snapshots" in a short time. Such temporal information is exploited in existing video object detection methods [18, 19, 12, 23] in a simple way. These methods firstly apply object detectors in single frames and then assemble the detected bounding boxes across temporal dimension in a dedicated post processing step. This step relies on off-the-shelf motion estimation such as optical flow, and hand-crafted bounding box association rules such as object tracking. In general, such methods manipulate the single-frame detection boxes with mediocre qualities but do not improve the detection quality. The performance improvement is from heuristic post-processing instead of principled learning. There is no end-to-end training. In this work, these techniques are called *box level methods*.

We attempt to take a deeper look at video object detection. We seek to improve the detection or recognition quality by exploiting temporal information, in a principled way. As motivated by the success in image recognition [11], *feature matters*, and we propose to improve the per-frame feature learning by temporal aggregation. Note that the features of the same object instance are usually not spatially aligned across frames due to video motion. A naive feature aggregation may even deteriorate the performance, as elaborated in Table 1 (b) later. This suggests that it is critical to model the motion during learning.

In this work, we propose *flow-guided feature aggregation* (FGFA). As illustrated in Figure 1, the feature extraction network is applied on individual frames to produce the per-frame feature maps. To enhance the features at a reference frame, an optical flow network [8] estimates the motions between the nearby frames and the reference frame. The feature maps from nearby frames are warped to the reference frame according to the flow motion. The warped features maps, as well as its own feature maps on the reference frame, are aggregated according to an adaptive weighting network. The resulting aggregated feature maps are then fed to the detection network to produce the detection result on the reference frame. All the modules of feature extraction, flow estimation, feature aggregation, and detection are

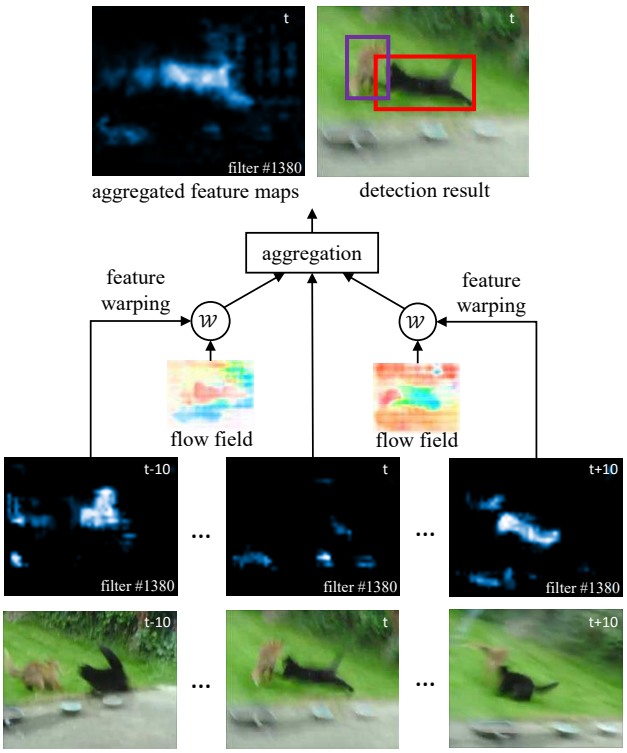

Figure 1. Illustration of FGFA (flow-guided feature aggregation). For each input frame, a feature map sensitive to "cat" is visualized. The feature activations are low at the reference frame t, resulting in detection failure in the reference frame. The nearby frames $t-10$ and $t+10$ have high activations. After FGFA, the feature map at the reference frame is improved and detection on it succeeds.

trained end-to-end.

Compared with box level methods, our approach works on *feature level*, performs end-to-end learning and is complementary (*e.g.*, to Seq-NMS [12]). It improves the per-frame features and generates high quality bounding boxes. The boxes can be further refined by box-level methods. Our approach is evaluated on the large-scale ImageNet VID dataset [33]. Rigorous ablation study verifies that it is effective and significantly improves upon strong single-frame baselines. Combination with box-level methods produces further improvement. We report object detection accuracy on par with the best engineered systems winning the ImageNet VID challenges, without additional bells-and-whistles (*e.g.*, model ensembling, multi-scale training/testing, *etc*.).

In addition, we perform an in-depth evaluation according to the object motion magnitude. The results indicate that the fast moving objects are far more challenging than slow ones. This is also where our approach gains the most. Our method can make effective use of the rich appearance information in the varied snapshots of fast moving objects.

2. Related Work

Object detection from image. State-of-the-art methods for general object detection [10, 30, 26, 5] are mainly based on deep CNNs [22, 36, 40, 14]. In [11], a multi-stage pipeline called Regions with Convolutional Neural Networks (R-CNN) is proposed for training deep CNN to classify region proposals for object detection. To speedup, ROI pooling is introduced to the feature maps shared on the whole image in SPP-Net [13] and Fast R-CNN [10]. In Faster R-CNN [30], the region proposals are generated by the Region Proposal Network (RPN), and features are shared between RPN and Fast R-CNN. Most recently, R-FCN [5] replaces ROI pooling operation on the intermediate feature maps with position-sensitivity ROI pooling operation on the final score maps, pushing the feature sharing to an extreme.

In contrast to these methods of still-image object detection, our method focuses on object detection in videos. It incorporates temporal information to improve the quality of convolutional feature maps, and can easily benefit from the improvement of still-image object detectors.

Object detection in video. Recently, ImageNet introduces a new challenge for object detection from videos (VID), which brings object detection into the video domain. In this challenge, nearly all of existing methods incorporate temporal information only on the final stage " bounding-box post-processing". T-CNN [18, 19] propagates predicted bounding boxes to neighboring frames according to pre-computed optical flows, and then generates tubelets by applying tracking algorithms from high-confidence bounding boxes. Boxes along the tubelets are re-scored based on tubelets classification. Seq-NMS [12] constructs sequences along nearby high-confidence bounding boxes from consecutive frames. Boxes of the sequence are re-scored to the average confidence, other boxes close to this sequence are suppressed. MCMOT [23] formulates the post-processing as a multi-object tracking problem. A series of hand-craft rules (*e.g.*, detector confidences, color/motion clues, changing point detection and forward-backward validation) are used to determine whether bounding boxes belong to the tracked objects, and to further refine the tracking results. Unfortunately, all of these methods are multi-stage pipeline, where results in each stage would rely on the results from previous stages. Thus, it is difficult to correct errors produced by previous stages.

By contrast, our method considers temporal information at the feature level instead of the final box level. The entire system is end-to-end trained for the task of video object detection. Besides, our method can further incorporate such bounding-box post-processing techniques to improve the recognition accuracy.

Motion estimation by flow. Temporal information in videos requires correspondences in raw pixels or fea-

tures to build the relationship between consecutive frames. Optical flow is widely used in many video analysis and processing. Traditional methods are dominated by variational approaches [2, 15], which mainly address small displacements [43]. The recent focus is on large displacements [3], and combinatorial matching (*e.g.*, DeepFlow [44], EpicFlow [31]) has been integrated into the variational approach. These approaches are all hand-crafted. Deep learning based methods (*e.g.*, FlowNet [8] and its successors [28, 17]) have been exploited for optical flow recently. The most related work to ours is deep feature flow [49], which shows the information redundancy in video can be exploited to speed up video recognition at minor accuracy drop. It shows the possibility of joint training the flow sub-network and the recognition sub-network.

In this work, we focus on another aspect of associating and assembling the rich appearance information in consecutive frames to improve the feature representation, and then the video recognition accuracy. We follow the design of deep feature flow to enable feature warping across frames.

Feature aggregation. Feature aggregation is widely used in action recognition [34, 20, 24, 47, 38, 1, 21, 41] and video description [7, 46]. On one hand, most of these work [34, 24, 47, 7, 46, 1, 9, 35] use recurrent neural network (RNNs) to aggregate features from consecutive frames. On the other hand, exhaustive spatial-temporal convolution is used to directly extract spatial-temporal features [38, 21, 41, 42]. However, the convolutional kernel size in these methods may limit the modeling of fast-moving objects. To address this issue, a large kernel size should be considered, but it will greatly increase the parameter number, brining issues of overfitting, computational overhead and memory consumption. By contrast, our approach relies on flow-guided aggregation, and can be scalable to different types of object motion.

Visual tracking. Recently, deep CNNs have been used for object tracking [25, 16] and achieved impressive tracking accuracy. When tracking a new target, a new network is created by combining the shared layers in the pre-trained CNN with a new binary classification layer, which is online updated. Tracking is apparently different from the video object detection task, because it assumes the initial localization of an object in the first frame and it does not require predicting class labels.

3. Flow Guided Feature Aggregation

3.1. A Baseline and Motivation

Given the input video frames $\{I_i\}, i = 1, \ldots, \infty$, we aim to output object bounding boxes on all the frames, $\{y_i\}, i = 1, \ldots, \infty$. A baseline approach is to apply an off-the-shelf object detector to each frame individually.

Modern CNN-based object detectors share a similar structure [11, 10, 30, 26, 5]. A deep convolutional sub-network $\mathcal{N}_{\text{feat}}$, is applied on the input image I, to produce feature maps $f = \mathcal{N}_{\text{feat}}(I)$ on the whole image. A shallow detection-specific sub-network, $\mathcal{N}_{\text{det}}$, is applied on the feature maps to generate the output, $y = \mathcal{N}_{\text{det}}(f)$.

Figure 2. Typical deteriorated object appearance in videos.

Video frames contain drastic appearance changes of the same object instance, as exemplified in Figure 2. Detection on single frames generates unstable results and fails when appearance is poor. Figure 1 presents an example. The feature responses for "cat" category are low at the reference frame t due to motion blur. This causes single frame detection failure. Observing that the nearby frames $t - 10$ and $t + 10$ have high responses, their features can be propagated to the reference frame. After the features on the reference frame is enhanced, detection on it succeeds.

Two modules are necessary for such feature propagation and enhancement: 1) motion-guided spatial warping. It estimates the motion between frames and warps the feature maps accordingly. 2) feature aggregation module. It figures out how to properly fuse the features from multiple frames. Together with the feature extraction and detection networks, these are the building blocks of our approach. They are elaborated below.

3.2. Model Design

Flow-guided warping. As motivated by [49], given a reference frame I_i and a neighbor frame I_j, a flow field $\mathbf{M}_{i\to j} = \mathcal{F}(I_i, I_j)$ is estimated by a flow network $\mathcal{F}$ (*e.g.*, FlowNet [8]).

The feature maps on the neighbor frame are warped to the reference frame according to the flow. The warping function is defined as

$$f_{j\to i} = \mathcal{W}(f_j, \mathbf{M}_{i\to j}) = \mathcal{W}(f_j, \mathcal{F}(I_i, I_j)), \qquad (1)$$

where $\mathcal{W}(\cdot)$ is the bilinear warping function applied on all the locations for each channel in the feature maps, and $f_{j\to i}$ denotes the feature maps warped from frame j to frame i.

Feature aggregation. After feature warping, the reference frame accumulates multiple feature maps from nearby frames (including its own). These feature maps provide diverse information of the object instances (*e.g.*, varied illuminations/viewpoints/poses/non-rigid deformations). For aggregation, we employ different weights at different spatial locations and let all feature channels share the same spatial weight. The 2-D weight maps for warped features $f_{j\to i}$ are denoted as $w_{j\to i}$. The aggregated features at the reference frame $\bar{f}_i$ is then obtained as

$$\bar{f}_i = \sum_{j=i-K}^{i+K} w_{j\to i} f_{j\to i}, \tag{2}$$

where K specifies the range of the neighbor frames for aggregation ($K = 10$ by default). Equation (2) is similar to the formula of attention models [32], where varying weights are assigned to the features in the memory buffer.

The aggregated features $\bar{f}_i$ are then fed into the detection sub-network to obtain the results,

$$y_i = \mathcal{N}_{\text{det}}(\bar{f}_i). \tag{3}$$

Compared to the baseline and previous box level methods, our method aggregates information from multiple frames *before* producing the final detection results.

Adaptive weight. The adaptive weight indicates the importance of all buffer frames $[I_{i-K}, \ldots, I_{i+K}]$ to the reference frame I_i at each spatial location. Specifically, at location p, if the warped features $f_{j\to i}(p)$ is close to the features $f_i(p)$, it is assigned to a larger weight. Otherwise, a smaller weight is assigned. Here, we use the cosine similarity metric [27] to measure the similarity between the warped features and the features extracted from the reference frame. Moreover, we do not directly use the convolutional features obtained from $\mathcal{N}_{\text{feat}}(I)$. Instead, we apply a tiny fully convolutional network $\mathcal{E}(\cdot)$ to features f_i and $f_{j\to i}$, which projects the features to a new embedding for similarity measure and is dubbed as the embedding sub-network.

We estimate the weight by

$$w_{j\to i}(p) = exp(\frac{f^e_{j\to i}(p) \cdot f^e_i(p)}{|f^e_{j\to i}(p)||f^e_i(p)|}), \tag{4}$$

where $f^e = \mathcal{E}(f)$ denotes embedding features for similarity measurement, and the weight $w_{j\to i}$ is normalized for every spatial location p over the nearby frames, $\sum_{j=i-K}^{i+K} w_{j\to i}(p) = 1$. The estimation of weight could be viewed as the process that the cosine similarity between embedding features passes through the SoftMax operation.

3.3. Training and Inference

Inference. Algorithm 1 summarizes the inference algorithm. Given an input video of consecutive frames $\{I_i\}$ and the specified aggregation range K, the proposed method sequentially processes each frame with a sliding feature buffer on the nearby frames (of length $2K + 1$ in general, except for the beginning and the ending K frames). At initial, the feature network is applied on the beginning $K + 1$ frames to initialize the feature buffer (L2-L4 in Algorithm 1). Then the algorithm loops over all the video frames to perform video object detection, and to update the feature buffer. For each frame i as the reference, the feature maps of the nearby frames in the feature buffer are warped with respect to it, and their respective aggregation weights are calculated (L6-L10). Then the warped features are aggregated and fed to the detection network for object detection (L11-L12). Before taking the $(i+1)$-th frame as the reference, the feature maps are extracted on the $(i+K+1)$-th frame and are added to the feature buffer (L13).

Algorithm 1 Inference algorithm of flow guided feature aggregation for video object detection.

1: **input**: video frames $\{I_i\}$, aggregation range K
2: **for** $k = 1$ **to** $K + 1$ **do** ▷ initialize feature buffer
3: $f_k = \mathcal{N}_{\text{feat}}(I_k)$
4: **end for**
5: **for** $i = 1$ **to** ∞ **do** ▷ reference frame
6: **for** $j = \max(1, i - K)$ **to** $i + K$ **do** ▷ nearby frames
7: $f_{j\to i} = \mathcal{W}(f_j, \mathcal{F}(I_i, I_j))$ ▷ flow-guided warp
8: $f^e_{j\to i}, f^e_i = \mathcal{E}(f_{j\to i}, f_i)$ ▷ compute embedding features
9: $w_{j\to i} = \exp(\frac{f^e_{j\to i} \cdot f^e_i}{|f^e_{j\to i}||f^e_i|})$ ▷ compute aggregation weight
10: **end for**
11: $\bar{f}_i = \sum_{j=i-K}^{i+K} w_{j\to i} f_{j\to i}$ ▷ aggregate features
12: $y_i = \mathcal{N}_{\text{det}}(\bar{f}_i)$ ▷ detect on the reference frame
13: $f_{i+K+1} = \mathcal{N}_{\text{feat}}(I_{i+K+1})$ ▷ update feature buffer
14: **end for**
15: **output**: detection results $\{y_i\}$

As for runtime complexity, the ratio of the proposed method versus the single-frame baseline is as

$$r = 1 + \frac{(2K+1) \cdot (\mathcal{O}(\mathcal{F}) + \mathcal{O}(\mathcal{E}) + \mathcal{O}(\mathcal{W}))}{\mathcal{O}(\mathcal{N}_{\text{feat}}) + \mathcal{O}(\mathcal{N}_{\text{det}})}, \tag{5}$$

where $\mathcal{O}(\cdot)$ measures the function complexity. Typically, the complexity of $\mathcal{N}_{\text{det}}$, $\mathcal{E}$ and $\mathcal{W}$ can be ignored when they are compared with $\mathcal{N}_{\text{feat}}$. The ratio is approximated as: $r \approx 1 + \frac{(2K+1)\cdot\mathcal{O}(\mathcal{F})}{\mathcal{O}(\mathcal{N}_{\text{feat}})}$. The increased computation mostly comes from $\mathcal{F}$. This is affordable, because the complexity of $\mathcal{F}$ is also much lower than that of $\mathcal{N}_{\text{feat}}$ in general.

Training. The entire FGFA architecture is fully differentiable and can be trained end-to-end. The only thing to note is that the feature warping module is implemented by bilinear interpolation and also fully differentiable w.r.t. both of the feature maps and the flow field.

Temporal dropout. In SGD training, the aggregation range number K is limited by memory. We use a large K in

inference but a small $K(=2$ by default) in training. This is no problem as the adaptive weights are properly normalized during training and inference, respectively. Note that during training, the neighbor frames are randomly sampled from a large range that is equal to the one during inference. As an analogy to dropout [37] technique, this can be considered as a *temporal dropout*, by discarding random temporal frames. As evidenced in Table 3, this training strategy works well.

3.4. Network Architecture

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

Flow network. We use FlowNet [8] ("simple" version). It is pre-trained on the Flying Chairs dataset [8]. It is applied on images of half resolution and has an output stride of 4. As the feature network has an output stride of 16 (see below), the flow field is downscaled by half to match the resolution of the feature maps.

Feature network. We adopt the state-of-the-art ResNet (-50 and -101) [14] and Inception-Resnet [39] as the feature network. The original Inception-ResNet is designed for image recognition. To resolve feature misalignment issue and make it proper for object detection, We utilize a modified version dubbed as "Aligned-Inception-ResNet", which is described in [6]. The ResNet-50, ResNet-101, and the Aligned-Inception-ResNet models are all pre-trained on ImageNet classification.

The pretrained models are crafted into feature networks in our FGFA model. We slightly modify the nature of three models for object detection. We remove the ending average pooling and the fc layer, and retain the convolution layers. To increase the feature resolution, following the practice in [4, 5], the effective stride of the last block is changed from 32 to 16. Specially, at the beginning of the last block ("conv5" for both ResNet and Aligned-Inception-ResNet), the stride is changed from 2 to 1. To retain the receptive field size, the dilation of the convolutional layers (with kernel size > 1) in the last block is set as 2. Finally, a randomly initialized 3×3 convolution is applied on top to reduce the feature dimension to 1024.

Embedding network. It has three layers: a $1 \times 1 \times 512$ convolution, a $3 \times 3 \times 512$ convolution, and a $1 \times 1 \times 2048$ convolution. It is randomly initialized.

Detection network. We use state-of-the-art R-FCN [5] and follow the design in [49]. On top of the 1024-d feature maps, the RPN sub-network and the R-FCN sub-network are applied, which connect to the first 512-d and the last 512-d features respectively. 9 anchors (3 scales and 3 aspect ratios) are utilized in RPN, and 300 proposals are produced on each image. The position-sensitive score maps in R-FCN are of 7×7 groups.

4. Experiments

4.1. Experiment Setup

ImageNet VID dataset [33]. It is a prevalent large-scale benchmark for video object detection. Following the protocols in [18, 23], model training and evaluation are performed on the 3,862 video snippets from the training set and the 555 snippets from the validation set, respectively. The snippets are fully annotated, and are at frame rates of 25 or 30 fps in general. There are 30 object categories. They are a subset of the categories in the ImageNet DET dataset.

Slow, medium, and fast motion. For better analysis, the ground truth objects are categorized according to their motion speed. An object's speed is measured by its averaged intersection-over-union (IoU) scores with its corresponding instances in the nearby frames (± 10 frames). The indicator is dubbed as "motion IoU". The lower the motion IoU is, the faster the object moves. Figure 3 presents the histogram of all motion IoU scores. According to the score, the objects are divided into slow (score > 0.9), medium (score $\in [0.7, 0.9]$), and fast (score < 0.7) groups, respectively. Examples from various groups are shown in Figure 4.

In evaluation, besides the standard mean average-precision (mAP) scores, we also report the mAP scores over the slow, medium, and fast groups, respectively, denoted as mAP(slow), mAP(medium), and mAP(fast). This provides us a more detailed analysis and in-depth understanding.

Implementation details. During training, following [18, 23], both the ImageNet DET training and the ImageNet VID training sets are utilized. Two-phase training is performed. In the first phase, the feature and the detection networks are trained on ImageNet DET, using the annotations of the 30 categories as in ImageNet VID. SGD training is performed, with one image at each mini-batch. 120K iterations are performed on 4 GPUs, with each GPU holding one mini-batch. The learning rates are 10^{-3} and 10^{-4} in the first 80K and in the last 40K iterations, respectively. In the second phase, the whole FGFA model is trained on ImageNet VID, where the feature and the detection networks are initialized from the weights learnt in the first phase. 60K iterations are performed on 4 GPUs, with learning rates of 10^{-3} and 10^{-4} in the first 40K and in the last 20K iterations, respectively. In both training and inference, the images are resized to a shorter side of 600 pixels for the feature network, and a shorter side of 300 pixels for the flow network. Experiments are performed on a workstation with Intel E5-2670 v2 CPU 2.5GHz and Nvidia K40 GPU.

4.2. Ablation Study

FGFA Architecture Design Table 1 compares our FGFA with the single-frame baseline and its variants.

Method (a) is the single-frame baseline. It has a mAP 73.4% using ResNet-101. It is close to the 73.9% mAP

$\mathcal{N}_{\text{feat}}$	ResNet-50					ResNet-101				
methods	(a)	(b)	(c)	(d)	(e)	(a)	(b)	(c)	(d)	(e)
multi-frame feature aggregation?		✓	✓	✓	✓		✓	✓	✓	✓
adaptive weights?			✓	✓	✓			✓	✓	✓
flow-guided?				✓	✓				✓	✓
end-to-end training?		✓	✓	✓			✓	✓	✓	
mAP (%)	70.6	$69.6_{\downarrow 1.0}$	$71.8_{\uparrow 1.2}$	$\mathbf{74.0}_{\uparrow \mathbf{3.4}}$	$72.1_{\uparrow 1.5}$	73.4	$72.0_{\downarrow 1.4}$	$74.3_{\uparrow 0.9}$	$\mathbf{76.3}_{\uparrow \mathbf{2.9}}$	$74.5_{\uparrow 1.1}$
mAP (%) (slow)	79.3	$81.4_{\uparrow 2.1}$	$81.5_{\uparrow 2.2}$	$\mathbf{82.4}_{\uparrow \mathbf{3.1}}$	$81.3_{\uparrow 2.0}$	82.4	$82.3_{\downarrow 0.1}$	$82.2_{\downarrow 0.2}$	$\mathbf{83.5}_{\uparrow \mathbf{1.2}}$	$82.5_{\uparrow 0.1}$
mAP (%) (medium)	68.6	$71.4_{\uparrow 2.8}$	$71.4_{\uparrow 2.8}$	$\mathbf{72.6}_{\uparrow \mathbf{4.0}}$	$71.5_{\uparrow 2.9}$	71.6	$74.5_{\uparrow 2.9}$	$74.6_{\uparrow 3.0}$	$\mathbf{75.8}_{\uparrow \mathbf{4.2}}$	$74.6_{\uparrow 3.0}$
mAP (%) (fast)	50.1	$42.5_{\downarrow 7.6}$	$50.4_{\uparrow 0.3}$	$\mathbf{55.0}_{\uparrow \mathbf{4.9}}$	$51.2_{\uparrow 1.1}$	51.4	$44.6_{\downarrow 6.8}$	$52.3_{\uparrow 0.9}$	$\mathbf{57.6}_{\uparrow \mathbf{6.2}}$	$53.2_{\uparrow 1.8}$
runtime (ms)	203	204	220	647	647	288	288	305	733	733

Table 1. Accuracy and runtime of different methods on ImageNet VID validation, using ResNet-50 and ResNet-101 feature extraction networks. The relative gains compared to the single-frame baseline (a) are listed in the subscript.

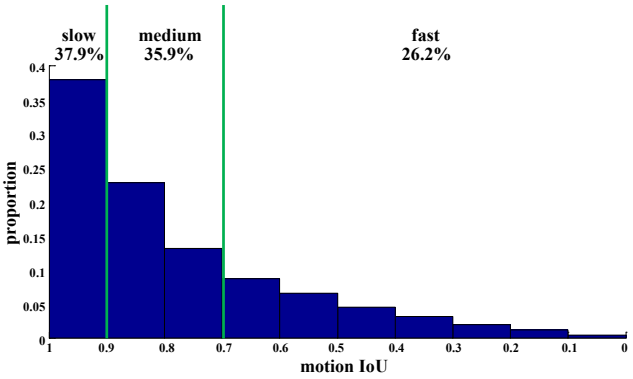

Figure 3. Histogram of the motion IoUs of all ground truth object instances, and the division of slow, medium and fast groups.

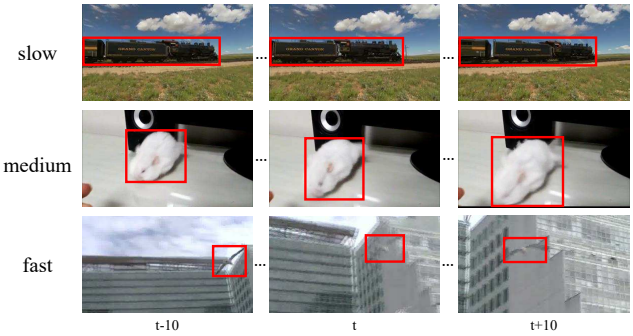

Figure 4. Example video snippets of object instances with slow, medium and fast motions. The motion IoUs are 0.98, 0.77 and 0.26, respectively.

instance size	small	middle	large
mAP (%)	24.2	49.5	83.2
mAP (%) (slow)	36.7	56.4	86.9
mAP (%) (medium)	32.4	51.4	80.9
mAP (%) (fast)	24.9	43.7	67.5

Table 2. Detection accuracy of small (area$< 50^2$ pixels), medium ($50^2 \leq$area$\leq 150^2$pixels), and large (area$> 150^2$pixels) object instances of the single-frame baseline (entry (a)) in Table 1.

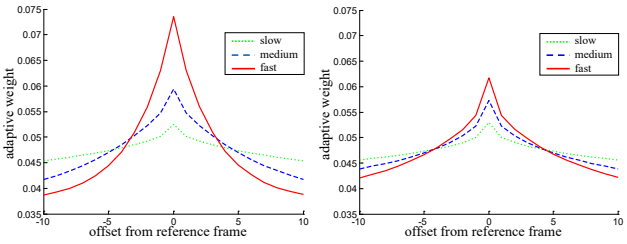

Figure 5. Adaptive weight distribution over frames. Left: entry without flow-guided feature warping (Table 1 (c)); Right: entry with flow-guided feature warping (Table 1 (d)). The histogram is performed within the boxes of instances with varying motions.

in [49], which is also based on R-FCN and ResNet-101. This indicates that our baseline is competitive and serves as a valid reference for evaluation. Note that we do not add bells and whistles like multi-scale training/testing, exploiting context information, model ensemble, *etc*., in order to facilitate comparison and draw clear conclusions.

Evaluation on motion groups shows that detecting fast moving objects is very challenging: mAP is 82.4% for slow motion, and it drops to 51.4% for fast motion. As objects of different sizes may have different motion speed, we further analyze the influence of the object size. Table 2 presents the mAP scores of small, middle, and large objects of different motion speeds. It shows that "fast motion" is an intrinsic challenge, irrespective to how large the object is.

Method (b) is a naive feature aggregation approach and a degenerated variant of FGFA. No flow motion is used. The flow map $M_{i\to j}$ is set to all zeros in Eq. (1). No adaptive weighting is used. The weight $w_{i\to j}$ is set to $\frac{1}{2K+1}$ in Eq. (2). The variant is also trained end-to-end in the same way as FGFA. The mAP decreases to 72.0% using ResNet-101, 1.4% shy of baseline (a). The decrease for fast motion (51.4% $\rightarrow$ 44.6%) is much more significant than that for slow motion (82.4% $\rightarrow$ 82.3%). It indicates that it is critical to consider motion in video object detection.

# training frames	2*							5						
# testing frames	1	5	9	13	17	21*	25	1	5	9	13	17	21	25
mAP (%)	70.6	72.3	72.8	73.4	73.7	74.0	74.1	70.6	72.4	72.9	73.3	73.6	74.1	74.1
runtime (ms)	203	330	406	488	571	647	726	203	330	406	488	571	647	726

Table 3. Results of using different number of frames in training and inference, using ResNet-50. Default parameters are indicated by *.

method	feature network	mAP (%)	runtime (ms)
single-frame baseline	ResNet-101	73.4	288
+ MGP		74.1	574*
+ Tubelet rescoring		75.1	1662
+ Seq-NMS		76.8	433*
FGFA	ResNet-101	76.3	733
+ MGP		75.5	1019*
+ Tubelet rescoring		76.6	1891
+ Seq-NMS		78.4	873*
FGFA	Aligned-	77.8	819
+ Seq-NMS	Inception-ResNet	**80.1**	954*

Table 4. Results of baseline method and FGFA before and after combination with box level techniques. As for runtime, entry marked with * utilizes CPU implementation of box-level techniques.

Method (c) adds the adaptive weighting module into (b). It obtains a mAP 74.3%, 2.3% higher than that of (b). Note that adding the adaptive weighting scheme is of little help for mAP (slow) and mAP (medium), but is important for mAP (fast) ($44.6\% \rightarrow 52.3\%$). Figure 5 (Left) shows that the adaptive weights for the fast moving instances concentrate on the frames close to the reference, which have relatively small displacement w.r.t. the reference in general.

Method (d) is the proposed FGFA method, which adds the flow-guided feature aggregation module to (c). It increases the mAP score by 2% to 76.3%. The improvement for fast motion is more significant ($52.3\% \rightarrow 57.6\%$). Figure 5 shows that the adaptive weights in (d) distribute more evenly over neighbor frames than (c), and it is most noticable for fast motion. It suggests that the flow-guided feature aggregation effectively promotes the information from nearby frames in feature aggregation. The proposed FGFA method improves the overall mAP score by 2.9%, and mAP (fast) by 6.2% compared to the single-frame baseline (a). Some example results are shown in Figure 6.

Method (e) is a degenerated version of (d) without using end-to-end training. It takes the feature and the detection sub-networks from the single-frame baseline (a), and the pre-trained off-the-shelf FlowNet. During training, these modules are fixed and only the embedding sub-network is learnt. It is clearly worse than (d). This indicates the importance of end-to-end training in FGFA.

As to runtime, the proposed FGFA method takes 733ms to process one frame, using ResNet-101 and FlowNet. It is slower than the single-frame baseline (288ms) because the flow network is evaluated $2K + 1(K = 10)$ times for each frame. To reduce the number of evaluation, we also experimented with another version of FGFA, in which the flow network is only applied on adjacent frame pairs. The flow field between non-adjacent frames is obtained by compositing the intermediate flow fields. In this way, the flow field computation on each adjacent frame pair can be re-used for different reference frames. The per-frame computation time of FGFA is reduced to 356ms, much faster than 733ms. The accuracy is slightly decreased ($\sim 1\%$) due to error accumulation in flow field composition.

 # frames in training and inference Due to memory issues, we use the lightweight ResNet-50 in this experiment. We tried 2 and 5 frames in each mini-batch during SGD training (5 frame reaches the memory cap), and 1, 5, 9, 13, 17, 21, and 25 frames in inference. Results in Table 3 show that training with 2 and 5 frames achieves very close accuracy. This verifies the effectiveness of our *temporal dropout* training strategy. In inference, as expected, the accuracy improves as more frames are used. The improvement saturates at 21 frames. By default, we sample 2 frames in training and aggregate over 21 frames in inference.

4.3. Combination with Box-level Techniques

Our approach focuses on improving feature quality and recognition accuracy in video frames. The output object boxes can be further improved by previous box-level techniques as post-processing. In particular, we tested three prevalent techniques, namely, motion guided propagation (MGP) [18], Tubelet rescoring [18], and Seq-NMS [12]. Note that MGP and Tubelet rescoring are used in the winning entry of ImageNet VID challenge 2015 [18]. We utilized the official public code for MGP and Tubelet rescoring, and re-implemented Seq-NMS.

Table 4 presents the results. The three techniques are firstly combined with our single-frame baseline using ResNet-101 model. They all improve the baseline. This indicates that such post-processing techniques are effective. Between them, Seq-NMS obtains the largest gain. When they are combined with FGFA using ResNet-101 model, no improvement is observed for MGP and Tubelet rescoring. However, Seq-NMS is still effective (mAP increased to 78.4%). By using Aligned-Inception-ResNet as the feature network, the mAP of FGFA+Seq-NMS is further improved

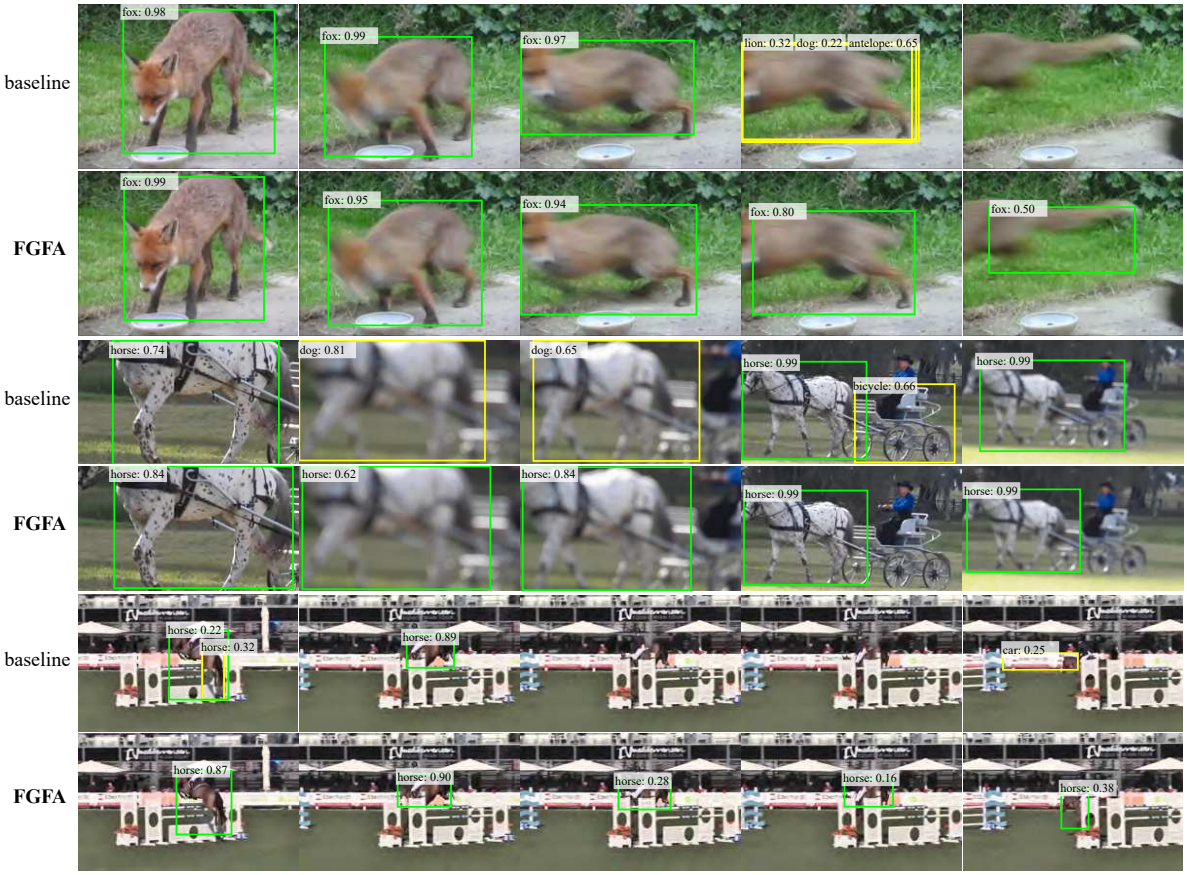

Figure 6. Example video clips where the proposed FGFA method improves over the single-frame baseline (using ResNet-101). The green and yellow boxes denote correct and incorrect detections, respectively. More examples are available at https://youtu.be/R2h3DbTPvVg.

to 80.1%, showing that Seq-NMS is highly complementary to FGFA.

Comparison with state-of-the-art systems Unlike image object detection, the area of video object detection lacks principled metrics [48] and guidelines for evaluation and comparison. Existing leading entries in ImageNet VID challenge 2015 and 2016 show impressive results, but they are complex and highly engineered systems with various bells and whistles. This makes direct and fair comparison between different works difficult.

This work aims at a principled learning framework for video object detection instead of the best system. The solid improvement of FGFA over a strong single frame baseline verifies the effectiveness of our approach. As a reference, the winning entry of ImageNet VID challenge 2016 (NUIST Team) [45] obtains 81.2% mAP on ImageNet VID validation. It uses various techniques like model ensembling, cascaded detection, context information, and multi-scale inference. In contrast, our approach does not use these techniques (only Seq-NMS is used) and achieves best mAP at 80.1%. Thus, we conclude that our approach is highly competitive with even the currently best engineered system.

5. Conclusion and Future Work

This work presents an accurate, end-to-end and principled learning framework for video object detection. Because our approach focuses on improving feature quality, it would be complementary to existing box-level framework for better accuracy in video frames. Several important aspects are left for further exploration. Our method slows down a bit, and it would be possibly sped up by more lightweight flow networks. There is still large room to be improved in fast object motion. More annotation data (*e.g.*, YouTube-BoundingBoxes [29]) and precise flow estimation may be benefit to improvements. Our method can further leverage better adaptive memory scheme in the aggregation instead of the attention model used. We believe these open questions will inspire more future work.

References

[1] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016. 3

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 3

[3] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 2011. 3

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

[5] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1, 2, 3, 5

[6] 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

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 3

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 1, 3, 5

[9] M. Fayyaz, M. Hajizadeh Saffar, M. Sabokrou, M. Fathy, R. Klette, and F. Huang. Stfcn: Spatio-temporal fcn for semantic video segmentation. In *ACCV Workshop*, 2016. 3

[10] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 1, 2, 3

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 3

[12] W. Han, P. Khorrami, T. Le Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016. 1, 2, 7

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 5

[15] B. K. Horn and B. G. Schunck. Determining optical flow. In *Artificial intelligence*, 1981. 3

[16] N. Hyeonseob and H. Bohyung. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 3

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 3

[18] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arxiv:1604.02532*, 2016. 1, 2, 5, 7

[19] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 1, 2

[20] A. Kar, N. Rai, K. Sikka, and G. Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *CVPR*, 2017. 3

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 3

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[23] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee. Multi-class multi-object tracking using changing point detection. In *ECCV*, 2016. 1, 2, 5

[24] Z. Li, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *arXiv preprint arXiv:1607.01794*, 2016. 3

[25] W. Lijun, O. Wanli, W. Xiaogang, and L. Huchuan. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 3

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 2, 3

[27] C. Luo, J. Zhan, L. Wang, and Q. Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *arXiv preprint arXiv:1702.05870*, 2017. 4

[28] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016. 3

[29] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017. 8

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3

[31] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 3

[32] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015. 4

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. In *IJCV*, 2015. 1, 2, 5

[34] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *ICLR Workshop*, 2016. 3

[35] M. Siam, S. Valipour, M. Jagersand, and N. Ray. Convolutional gated recurrent networks for video segmentation. *arXiv preprint arXiv:1611.05435*, 2016. 3

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *JMLR*, 2014. 5

[38] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 3

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

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[41] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 3

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep end2end voxel2voxel prediction. In *CVPR Workshop*, 2016. 3

[43] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg. A survey on variational optic flow methods for small displacements. In *Mathematical models for registration and applications to medical imaging*. 2006. 3

[44] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 3

[45] J. Yang, H. Shuai, Z. Yu, R. Fan, Q. Ma, Q. Liu, and J. Deng. Efficient object detection from videos. http://image-net.org/challenges/talks/2016/Imagenet2016VID.pptx, 2016. 8

[46] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015. 3

[47] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 3

[48] H. Zhang and N. Wang. On the stability of video detection and tracking. *arXiv preprint arXiv:1611.06467*, 2016. 8

[49] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *CVPR*, 2017. 3, 5, 6