Object Guided External Memory Network for Video Object Detection

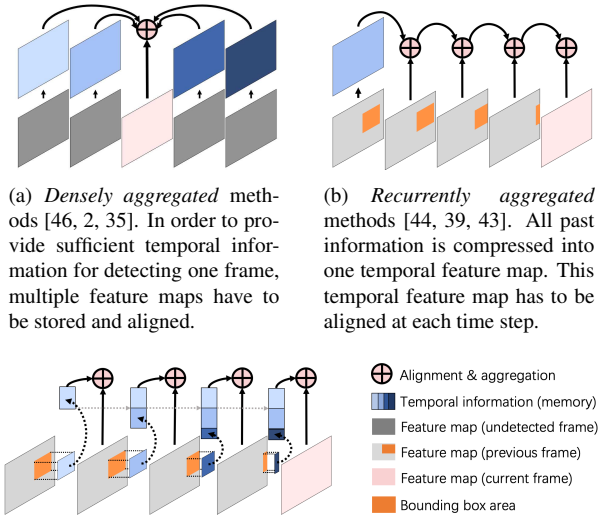
Object Guided External Memory Network for Video Object Detection

Hanming Deng [1], Yang Hua[2], Tao Song[1], Zongpu Zhang[1], Zhengui Xue[1], Ruhui Ma*[1], Neil Robertson[2], and Haibing Guan[1†]

[1]Shanghai Jiao Tong University
{denghanmig,songt333,zhang-z-p,zhenguixue,ruhuima,hbguan}@sjtu.edu.cn
[2]Queen's University Belfast
{Y.Hua, N.Robertson}@qub.ac.uk

Abstract

Video object detection is more challenging than image object detection because of the deteriorated frame quality. To enhance the feature representation, state-of-the-art methods propagate temporal information into the deteriorated frame by aligning and aggregating entire feature maps from multiple nearby frames. However, restricted by feature map's low storage-efficiency and vulnerable content-address allocation, long-term temporal information is not fully stressed by these methods. In this work, we propose the first object guided external memory network for online video object detection. Storage-efficiency is handled by object guided hard-attention to selectively store valuable features, and long-term information is protected when stored in an addressable external data matrix. A set of read/write operations are designed to accurately propagate/allocate and delete multi-level memory feature under object guidance. We evaluate our method on the ImageNet VID dataset and achieve state-of-the-art performance as well as good speed-accuracy tradeoff. Furthermore, by visualizing the external memory, we show the detailed object-level reasoning process across frames.



(a) *Densely aggregated* methods [46, 2, 35]. In order to provide sufficient temporal information for detecting one frame, multiple feature maps have to be stored and aligned.

(b) *Recurrently aggregated* methods [44, 39, 43]. All past information is compressed into one temporal feature map. This temporal feature map has to be aligned at each time step.

(c) Our method using an object guided external memory. Only features within bounding boxes can be stored for storage-efficiency, and each feature is deleted only when redundant to protect long-term information.

Figure 1: Comparison between our method and others. After alignment & aggregation, the aggregated feature map is used for detection on current frame. *Best viewed in color.*

1. Introduction

State-of-the-art image-based object detectors [13, 9, 27, 5, 23, 26, 22] provide effective detection frameworks for general object detection. However, their performance decays when they are directly applied to videos due to the low quality of video frames, such as motion blur, out-of-focus and occlusion.

In order to improve the detection performance in a principled way, state-of-the-art video object detectors [45, 44, 2, 35, 39, 43] exploit rich temporal information in videos to enhance the feature representation on these deteriorated frames. Feature maps of different frames are first aligned due to frame content displacement and then aggregated with the current frame. These temporal feature maps, referred to as various names like spatial-temporal memory [39] or memory buffer [45], are taken directly as memory to prop-

*Ruhui Ma is the corresponding author.

†This work was supported in part by National NSF of China (NO. 61525204, 61732010, 61872234) and Shanghai Key Laboratory of Scalable Computing and Systems.

agate temporal information. Here, we put these methods under a unified view in terms of how memory is read from and written to.

In *densely aggregated* methods (Figure 1(a)), memory is composed of multiple nearby feature maps. Reading involves aligning and aggregating all memory feature maps into the current frame. The aggregated feature map is used for detection on the current frame. Writing happens after each detection, where nearby feature maps of the next frame are written to replace the current memory. These methods aim at providing sufficient temporal information for the current frame. In *recurrently aggregated* methods (Figure 1(b)), memory consists of only one feature map. Reading and writing happen simultaneously. When the memory feature map is aligned and aggregated into the current frame, the aggregated feature map is used for detection on the current frame and becomes the new memory. These methods are faster in computation and are capable of online causal inference. In both *densely aggregated* and *recurrently aggregated* methods, the memory is composed of complete-sized feature maps within the detection network, whose size and content address are dependent on the behaviour of the detection network and input frame, thus we refer to this memory as "internal".

The internal memory has drawbacks in temporal propagation. In dense methods, usually more than 20 nearby feature maps [45, 2] are stored to provide sufficient information. Because of the complete-sized feature maps in internal memory, the redundant information irrelevant to the detecting objects is also stored and propagated, leading to low storage-efficiency. In recurrent methods, all past information is compressed into one feature map, in which the spatial location of information is solely dependent on the location of the current frame content. As a result, long-term information is easily interrupted when the current frame content is deteriorated, drastically changing in appearance or out-of-sight, since the current aggregated feature map will become new internal memory.

To better exploit temporal information, in this work, we propose an object guided external memory network for online video object detection, as shown in Figure 1(c). By external memory [11], hereinafter, we mean the kind of memory whose size and content address are independent of the detection network and the input frame. Our method targets at the drawbacks of internal memory. First, object information produced in detection offers a natural hard attention [25] to improve storage-efficiency. By selectively storing features within detected bounding boxes instead of entire feature maps, more temporal information across longer time steps can be stored into the same amount of storage. Second, as motivated by Neural Turing Machine (NTM) [10, 11] and Memory Network [37, 31], long term dependency and reasoning power can be tackled by storing previous pieces of temporal information into an addressable data matrix for essential period of time and deleting only redundant ones. Important information from the past can be accurately recalled without the risk of being accidentally forgotten. The size of our external memory varies with the number of features stored. Novel *read* and *write* operations are designed to access this dynamic external memory, where attention-based [34, 17, 10] *read* focuses on propagating variable sized memory accurately to the current frame while balancing present/memory feature signal strength in the aggregated feature map, and *write* selectively stores new features and deletes redundant memory under object guided hard attention.

To our best knowledge, this is the first work on exploring external memory in video object detection. Meanwhile, we test with different levels of visual features in external memory, including *pixel* level features from convolutional feature maps with finer details and *instance* level features produced by roi-pooling with higher semantic information, and achieve complementary detection performance. To show the reasoning power and importance of long-term dependency, we further visualize our external memory and *read* operation, in which the reasoning process for hard examples reveals how high quality objects in memory lead to a correct prediction on the current deteriorated frame across large temporal distance. In addition, based on the property of external memory, we can propagate memory to feature maps of different sizes, which inspires us to explore speed-accuracy tradeoff by reducing computation on sparsely down-scaled frames.

To sum up, our contributions are listed as follows:

- We propose to use object guided hard attention to improve storage-efficiency and propagate temporal information through external memory to address long-term dependency in video object detection.
- We design an object guided external memory network and novel *write* and *read* operations to efficiently store and accurately propagate multi-level features.
- Extensive experiment studies on the ImageNet VID dataset show our network can achieve state-of-the-art performance as well as good speed-accuracy tradeoff, meanwhile we visualize the external memory to show the reasoning details and importance of long-term dependency.

2. Related Work

Single Image Object Detection. The state-of-the-art image-based object detectors [13, 9, 27, 5, 23, 26, 22] are built upon deep convolutional neural networks (CNNs). A feature network, consisting of deep CNNs [14, 33, 30], takes an image as input, and the output feature map is sent to a detection-specific sub-network to produce detection results in the form of bounding box coordinates and object

class. Roi-pooling [13, 27] performs max pooling within proposal regions on the feature map and resizes it to a fixed size. Region Proposal Network is introduced in Faster-RCNN [27] to generate region proposals and can be end-to-end trained. In R-FCN [5], position-sensitive score maps and position-sensitive roi-pooling (psroi-pooling) are proposed to address translation-variance for accurate and efficient detection. We use R-FCN as our single frame baseline.

Video Object Detection. The task of video object detection aims to detect every frame of a video. Box level methods [19, 20, 12, 8, 3, 24] optimize the bounding box linkage across multiple frames. T-CNN [19, 20] leverages optical flow to propagate the bounding box across frames and links the bounding box into tubelets with tracking algorithms. Seq-NMS [12] considers all bounding boxes within a video and re-scores them for optimal bounding box linkage. DTTD [8] simultaneously achieves detections and trackings. The frame level detections are linked by across-frame tracklets to provide the final prediction. STL [3] detects on sparse key frames, and propagates the predicted bounding box to non-key frames through motion and scales. DorT [24] combines detection and tracking for efficient detection.

Feature level methods [45, 44, 2, 35, 39, 43] aggregate feature maps from nearby frames to improve feature representation and recognition ability in a principled way. Feature maps to be aggregated are usually aligned before aggregation due to frame content displacement. Some methods [46, 45, 36] use FlowNet [7, 18] to provide flow information for feature map alignment, and the aligned feature maps are densely [46, 36] or recurrently aggregated [44]. STSN [2] applies deformable convolution [6] across multiple nearby frames for temporal inference. LWDN [43] uses locally weighted deformable units to align different levels of feature maps from nearby frames. STMN [39] uses Match-Trans module to align feature maps and recurrently aggregates temporal information from nearby frames. However, temporal information is still stored and propagated through entire feature maps. Our method also belongs to feature level. All the feature level methods above are our baselines.

External Memory. External memory for long term dependency and inference in neural networks has been studied on various tasks, such as learning simple algorithms [10], question answering [37, 31, 11], language modeling [31], dialog system [38] and meta learning [29]. Recurrent neural networks are known to suffer from the vanishing gradient problem. Some variations like Long Short Term Memory [16] and Gated Recurrent Units [4] address this problem but their memory (hidden states) size is usually small with limited capacity to remember the past and is not fully capable of carrying complex tasks that demand a search from the past during inference. To tackle this, in recent work on different tasks [10, 37, 31, 11, 31, 38, 29], external memory is used to enhance the sequential inference ability of neural networks. The external memory is usually an addressable external data matrix. It is maintained by an individual network, usually with attentional read and write process. Information can be retrieved from the external memory by an attention-based global search and aggregation. New information is written to the external memory on specific location, thus memory information can be explicitly reserved for a long period. Besides the attention or content-based addressing mechanism, in NTM [10], a location-based addressing is proposed for algorithmic tasks like sort and copy.

Attention for Memory Addressing. Attention is used as a "soft" addressing mechanism on external memory [10, 11, 37, 31]. In order to retrieve information from the memory, an attention weight is computed on the similarity between all memory locations and the input features that query the memory, then all memory is aggregated based on these attention weights. Besides external memory, such addressing mechanism has also been used in self-attention [1, 34, 17, 42], non-local neural networks [36] and convolutional feature extraction [40] for aligning features, and modeling long-term dependency of different locations within a sentence or an image. One advantage of this attention mechanism is that the modeling of dependency is irrelevant of feature locations and their distance in a sentence or an image. Based on this property, we are able to accurately propagate scattered memory through *read* without consideration of spatial location.

Hard attention is also used in memory addressing [41] and feature selection [40, 25]. Different from the soft attention, hard attention is non-differentiable and is trained with reinforcement learning [32] or predefined based on human knowledge. It improves computation efficiency because only highly relevant information is selected and processed. We leverage the box and class information produced by detection to define hard attention for feature selection and comparison. Soft and hard attentions are respectively used in our read and write operations.

3. Detection with Object Guided External Memory

3.1. Framework Overview and Detection Process

Our proposed framework is built upon the single image detector, by incorporating a novel object guided external memory network N_{mem} to accurately propagate/store temporal information into/from the single image detector at each frame's detection. N_{mem} consists of two external memory matrices M_{pix} and M_{inst}, where *pixel* and *instance* level features are stored respectively, and a set of *read* and *write* operations to access them. Detailed structure of N_{mem} is described in the next section. For a fair comparison with the previous work [45, 44, 35, 2, 43], we

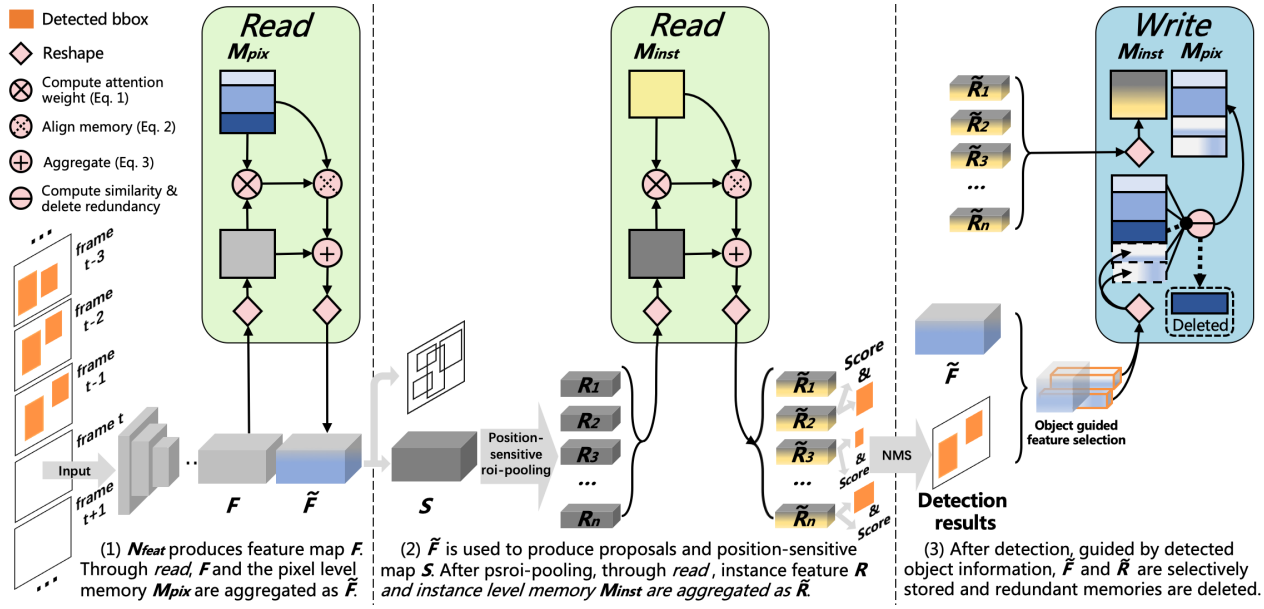


Figure 2: Our proposed framework using object guided external memory network. *Best viewed in color.*

use R-FCN [5] as the backbone detector.

The detection process of our proposed framework is straight-forward, as shown in Figure 2. Given a video, our framework detects each frame in an online order. Suppose that frame t is the current detecting frame of a video. First, the convolutional feature map F is produced by the feature network N_{feat}. Through *read* operation, memory in M_{pix} is propagated to F and they are aggregated as enhanced feature map $\tilde{F}$. $\tilde{F}$ is used to produce both the position-sensitive map S and the region proposals. Second, S is position-sensitive roi-pooled (psroi-pooled) to produce instance features $R = \{R_i, \cdots, R_n\}$ with each $R_i, i \in \{1, \cdots, n\}$ being a psroi-pooled instance feature. Through *read* operation again, memory in M_{inst} is propagated to R and they are aggregated as the enhanced instance features $\tilde{R} = \{\tilde{R}_i, \cdots, \tilde{R}_n\}$. Each $\tilde{R}_i, i \in \{1, \cdots, n\}$ is pooled to produce one detection result. Third, Non-Maximum Suppression (NMS) is applied to remove duplicates. After detection of frame t, guided by information of detected objects, N_{mem} can select valuable features from $\tilde{F}$ and $\tilde{R}$, and store them into the external memory through *write* operation. Meanwhile, redundant features in the external memory are deleted. We define feature vectors from $\tilde{F}$ as *pixel* level features, and $\tilde{R}$ as *instance* level features. These are the two levels of features stored into the external memory matrices M_{pix} and M_{inst} respectively.

3.2. Object Guided External Memory Network

The proposed object guided external memory network N_{mem} contains two external memory matrices M_{pix} and M_{inst}, and novel *read*, *write* operations, which are described later in this section. Note that the external memory only stores features from the same video.

External memory matrices store *pixel* and *instance* features, with each row of M_{pix} being a feature vector from some locations in a convolutional feature map $\tilde{F}$, and each row of M_{inst} being one psroi-pooled instance feature $\tilde{R}_i, i \in \{1 \cdots, n\}$ that is reshaped into a vector. The size of the external data matrix grows when new features are written into the memory, and decreases when features are deleted from the memory. Unless a video ends or *write* operation explicitly deletes a feature, features in external memory matrices are permanently stored.

***Read* operation** propagates the memory matrices M_{pix} and M_{inst} into feature map F and instance features R respectively. It involves three steps, similar to the attention mechanism used in [34, 17, 42], compute attention weight (Eq. (1)), align memory (Eq. (2)) and aggregate memory with input feature map (Eq. (3)), as shown in the green boxes in Figure 2. The challenge is that the external memory size varies, leading to big difference of signal strength in the aligned memory. To address this, we introduce scale parameters θ and β to ensure that the features in aligned memory are in proper magnitude and signal strength of present/memory features in the output aggregated feature maps is balanced.

Let $\boldsymbol{m} \in \mathbb{R}^{l \times c}$ be the memory matrix, either M_{pix} or M_{inst}, with the i^{th} row denoted as $\boldsymbol{m_i}$. Let $\boldsymbol{x} \in \mathbb{R}^{n \times c}$ be the corresponding input feature matrix reshaped from F or R, with the j^{th} row denoted as $\boldsymbol{x_j}$. First, we compute the attention weights $w = \{w_{i,j}\}$ between $\boldsymbol{m}$ and $\boldsymbol{x}$:

$$w_{i,j} = \frac{exp(S(\boldsymbol{m_i}, \boldsymbol{x_j}))}{\Sigma_{j=1}^{n} exp(S(\boldsymbol{m_i}, \boldsymbol{x_j}))}, \tag{1}$$

where $S \in \mathbb{R}$ evaluates the similarity between $\boldsymbol{m_i}$ and $\boldsymbol{x_j}$.

Second, with the attention weights, external memory matrix $\boldsymbol{m}$ is aligned into $\tilde{\boldsymbol{m}} \in \mathbb{R}^{n \times c}$ with the j^{th} row denoted as $\tilde{\boldsymbol{m}}_{\boldsymbol{j}}$ being:

$$\tilde{\boldsymbol{m}}_{\boldsymbol{j}} = \theta_j \Sigma_{i=1}^{l} w_{i,j} \boldsymbol{m_i}, \tag{2}$$

where $\theta = \{\theta_j\} \in \mathbb{R}^n$ is a set of scale parameters to ensure $\tilde{\boldsymbol{m}}$ is in a similar magnitude to $\boldsymbol{x}$.

Third, the aligned memory $\tilde{\boldsymbol{m}}$ is added into the input features $\boldsymbol{x}$, producing the output of *read*, $\boldsymbol{y} \in \mathbb{R}^{n \times c}$ with the j^{th} row denoted as $\boldsymbol{y_j}$ being:

$$\boldsymbol{y_j} = \beta_j \boldsymbol{x_j} + (1 - \beta_j) \tilde{\boldsymbol{m}}_{\boldsymbol{j}}, \tag{3}$$

where $\beta = \{\beta_j\} \in \mathbb{R}^n$ is another set of scale parameters to balance $\boldsymbol{x}$ and $\tilde{\boldsymbol{m}}$. Finally, $\boldsymbol{y}$ is reshaped into the same shape as input feature map F, becoming the enhanced feature map $\tilde{F}$, or split into n instance features $\{\tilde{R}_i, \cdots, \tilde{R}_n\}$.

The similarity measure S is given by:

$$S(\boldsymbol{m_i}, \boldsymbol{x_j}) = \frac{\lambda (W_K \boldsymbol{m_i})^T (W_Q \boldsymbol{x_j})}{\|W_K \boldsymbol{m_i}\| \|W_Q \boldsymbol{x_j}\|}, \tag{4}$$

where $W_K, W_Q \in \mathbb{R}^{\bar{c} \times c}$ are learnable embedding weights, and they embed the original features into a sub-space for similarity comparison. $\lambda \in \mathbb{R}$ is a learnable temperature parameter to control the concentration of attention, as in NTM [10]. The scale parameters θ and β are given by:

$$\begin{aligned} \theta_j &= \frac{1}{max(\Sigma_{i=1}^{l} w_{i,j}, 1)}, \\ \beta_j &= \frac{1}{min(\Sigma_{i=1}^{l} w_{i,j}, 1) + 1}. \end{aligned} \tag{5}$$

Note that $\Sigma_{i=1}^{l} w_{i,j} \neq 1$, given how $w_{i,j}$ is computed in Eq. (1). This is a key difference from the other work [10, 34, 17], in which the denominator in Eq. (1) integrals along i axis. We do this to ensure that newly appeared features or background features are less affected by memory, and its effect can be observed in the attention visualization in Figure 3(a). θ attenuates the signal strength of $\tilde{\boldsymbol{m}}$ when too much memory information is concentrated on a same location of the input feature map. β ensures that $\tilde{\boldsymbol{m}}$ will not dominate $\boldsymbol{x}$ in the output $\boldsymbol{y}$.

In practice we use multi-head attention [34, 17]. N sets of embedding weights $\{W_K^1, ..., W_K^N\}$, $\{W_Q^1, ..., W_Q^N\}$ are applied, thus N sets of attention weights $\{w^1, ...w^N\}$ are computed by Eq. (1). Meanwhile, $\boldsymbol{m}$ is embedded to N sub-spaces by $\{W_V^1, ..., W_V^N\}$, where $W_V^k \in \mathbb{R}^{\frac{c}{N} \times c}, k = 1, \cdots, N$ and Eq. (2) is changed to

$$\tilde{\boldsymbol{m}}_{\boldsymbol{j}}^{\boldsymbol{k}} = \theta_j \Sigma_{i=1}^{l} w_{i,j}^k (W_V^k \boldsymbol{m_i}). \tag{6}$$

All $\{\tilde{\boldsymbol{m}}_{\boldsymbol{j}}^{\boldsymbol{1}}, ..., \tilde{\boldsymbol{m}}_{\boldsymbol{j}}^{\boldsymbol{N}}\}$ are concatenated into one vector $\tilde{\boldsymbol{m}}_{\boldsymbol{j}} = [\tilde{\boldsymbol{m}}_{\boldsymbol{j}}^{\boldsymbol{1}}, ..., \tilde{\boldsymbol{m}}_{\boldsymbol{j}}^{\boldsymbol{N}}]$ to compute the output $\boldsymbol{y}$. For computing θ and β, we have $w_{i,j} = \frac{1}{N} \Sigma_{k=1}^{N} w_{i,j}^k$.

***Write* operation** selects and stores *pixel* and *instance* level features into the external memory, meanwhile it removes redundant features from the external memory, as shown in the blue box in Figure 2. The methods are different for the two levels of features. For *pixel* level features, *write* is guided by detected objects. It first selects high quality feature vectors belonging to detected objects in current enhanced feature map $\tilde{F}$, and then it stores the selected features at new rows in external memory matrix M_{pix}. Redundant features in M_{pix} are deleted after new features are stored. We define the features belonging to an object b as all vectors $\{\tilde{\boldsymbol{x}}_{\boldsymbol{k_i}}\}$ whose locations in feature map $\tilde{F}$ are within the bounding box of object b:

$$O_b = (\tilde{\boldsymbol{x}}_{\boldsymbol{k_1}}, ..., \tilde{\boldsymbol{x}}_{\boldsymbol{k_{n_b}}})^T, \tag{7}$$

where n_b is the total number of vectors belonging to O_b. We also define the quality of O_b as the classification score s of object b.

To select O_b into the external memory, first, the feature quality of O_b needs to be larger than a threshold τ. Second, O_b should either be dissimilar to any O_p that is already in the external memory and shares the same class with b or be similar to some O_p in the external memory but O_b has higher quality. Once O_b is stored in the external memory, any O_p that is similar to O_b is considered redundant and is removed out of M_{pix}.

Similarity between two objects is calculated with off-the-shelf $w = \{w_{i,j}\}$ calculated by Eq. (1). Let $\hat{i} \in \{k_1, ..., k_{n_b}\}$ denotes the indices of the feature vectors that belong to O_b, and $\hat{j} \in \{k_1, ..., k_{n_p}\}$ denotes those of O_p, then the similarity between O_b and O_p is computed as:

$$D(O_b, O_p) = \frac{1}{n_b} \Sigma_{\hat{i}} \Sigma_{\hat{j}} w_{\hat{i},\hat{j}}. \tag{8}$$

$D \in \mathbf{R}$ measures how much the attention of memory object O_p is concentrated on the area of O_b in feature map F. Since attention calculated by Eq. (1) is based on feature similarity, focused attention between two objects suggests that they are similar. For objects b and p of the same class, they are similar if $D(O_b, O_p) > max(\frac{n_p}{n}, \alpha)$ with n being the total number of locations in feature map F and α being a constant.

In practice, in case that no object is detected, we also include object-irrelevant features into the external memory. We select these features by hard attention using L^2-norms of feature vectors [25]. For a feature map with shape $n \times c$, a feature vector $\boldsymbol{x_j}$ is selected if $exp(f_j)/\Sigma_{i=1}^{n} exp(f_i) > \frac{1}{n}$, where $f_j = \frac{\|\boldsymbol{x_j}\|}{\sqrt{c}}$. Unlike features belonging to detected objects, when storing these object-irrelevant features into

the external memory, they replace all older features that are also picked based on L^2-norms.

For *instance* level features, psroi-pooling is a natural hard attention on $\tilde{R} = \{\tilde{R}_i, \cdots, \tilde{R}_n\}$. Since they are already selected when Non-Maximum Suppression (NMS) is adopted on proposals before psroi-pooling, *write* operation chooses $\tilde{R}$ of a frame, and stores them into the memory data matrix M_{inst} to replace all older *instance* memory. Each $\tilde{R}_i, i = 1, \cdots, n$ is reshaped into a vector, and all the reshaped $\{\tilde{R}_i\}$ are packed into a matrix to be stored in M_{inst}.

3.3. Efficient Detection with External Memory

As observed in previous work [46], some adjacent frames have similar content. Meanwhile, our proposed memory network enables us to propagate memory to feature maps of different sizes. For time efficiency, we set a key frame schedule, and down-sample all non-key frames to reduce their computation time in N_{feat}, which is the bottle neck in detection inference time. We update external memory only on full size key frames, and propagate the high quality features to down-sampled non-key frames to compensate for deterioration caused by down-sampling. The attention-based *read* operation is irrelevant to spatial location, as well as image size, thus features from different size inputs can be seamlessly aggregated. As a result, apart from high performance with our full model, we also seek for speed-accuracy tradeoff with this method in online video object detection.

4. Experiments

4.1. Experiment Set-up and Implementation Details

Dataset. We evaluate our method on the ImageNet [28] VID dataset. The ImageNet VID is introduced in 2015 as a large scale benchmark for video object detection. There are 3862 videos in the training set and 555 videos in the validation set. It contains 30 object categories. These categories are a subset of the categories of ImageNet DET set.

Training and testing. As a common practice in [45, 46, 44, 35, 2], we train on both the ImageNet VID and the DET datasets. For the DET dataset, we select the same 30 categories from the DET dataset as in the VID dataset. We train R-FCN for 200K iteration, with a learning rate 0.0005 for the first 120K epochs and 0.00005 for the last 80K epochs. Each batch contains 1 randomly sampled image. The images are resized to have a shorter side of 600 pixels. The feature network is initialized with weights pretrained on the ImageNet dataset. Then we add embedding weights from *read* to train them end-to-end. The *read* operation is added between feature maps and position-sensitive roi-pooled position-sensitive score maps from consecutive training samples respectively. We first only train embedding weights for 60K iterations with a learning rate 0.0001 and then train the whole network for 10K iterations with a learning rate 0.00005. Each batch contains 4 images randomly sampled from a same video. A maximum frame interval is set to 20. W_Q and W_K are initialized with a normal distribution $N(0, 0.01)$. Temperature parameter λ is initailized as 1. For W_V, it is initialized as an identity matrix to pick up long term dependency as suggested in IRNN [21].

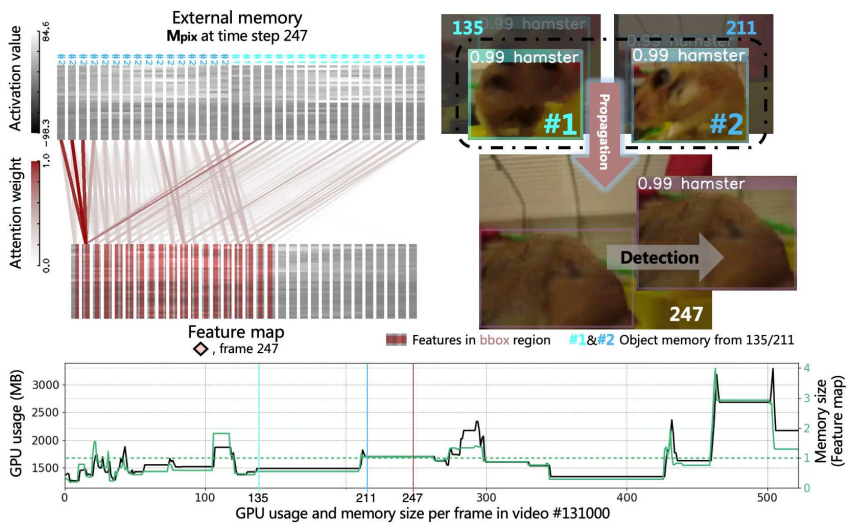
Following [45, 44, 35, 2, 43], we evaluate our performance on the validation set and use mean average precision (mAP) as the evaluation metric. All our experiments are performed on two Nvidia GTX 1080 Ti.

Implementation Details. In Multi-head matching [34, 17] in Eq. (6), we set $N = 8$ head embeddings, and for each $W_K, W_Q \in \mathbb{R}^{\bar{c} \times c}$, $\bar{c} = 32$. We set $\tau = 0.9$ for selecting high quality object feature and $\alpha = 0.47$ for object similarity threshold based on empirical assumption. Specifically, we assume that the attention distribution from an object in memory to a detecting frame is a two-dimensional Gaussian distribution, centering at a similar object's box center and let standard deviation σ equal half of the box's diagonal length. Based on the empirical one sigma rule, we consider this two objects are similar if most of the attention is distributed within one σ range. For a fair comparison with previous work [45, 44, 43, 8, 35, 2], ResNet-101 [15] and R-FCN [5] are used as the feature network and backbone detector in the same way. All layers after "conv5" are dropped. Feature map stride of the final output is reduced from 32 to 16 by setting the stride of "conv5" from 2 to 1. A dilated 3x3 convolution is attached after "conv5" to reduce the feature dimension from 2048 to 1024. The reduced 1024-d feature map from feature network is used as *pixel* level features, and it is sliced to two 512-d feature maps for the RPN and the R-FCN sub-network respectively. In the RPN sub-network, 9 anchors are predicted on each position and it outputs 300 proposals per image for the detection. 7×7 position-sensitive roi-pooling is utilized in the R-FCN sub-network. We limit the number of objects belonging to the same class in *pixel* level memory to 10 for efficiency.

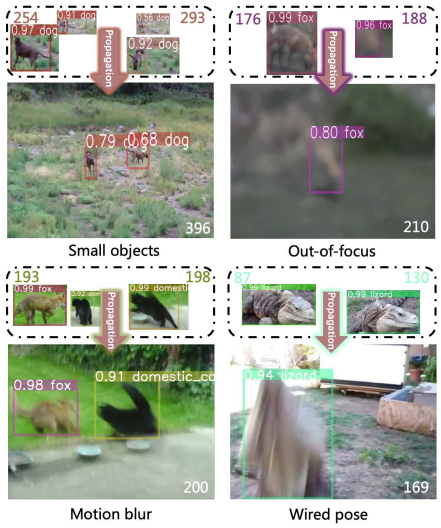
4.2. Visualization of external memory

In Figure 3, we illustrate some of our detection results on the ImageNet VID validation set and inference process of our external memory. These frames are chosen to show typical cases in video object detection, such as motion blur, out-of-focus, occlusion,and rare object appearance like weird pose and small size, where general image detectors, e.g., R-FCN [5], fail.

A reasoning process for a heavily occluded hamster is shown in Figure 3(a). We can clearly see what memories are stored, and how these memories lead to a correct detection on a deteriorated image. At time step 247, there are features of detected hamster ($\#1$ and $\#2$) stored in the external memory M_{pix} from frame 135 and 211 respectively.



(a) Cross-frame inference with external memory on heavily occluded example/GPU usage and memory size during test. For visualization, we pool every 60 feature vectors into one row.



(b) More examples. Memory from most recent two steps (colored number) is illustrated.

Figure 3: Visualization of our method on ImageNet [28] VID dataset. *Best viewed in color.*

Given the feature map of frame 247, the attention of M_{pix} (red lines) is concentrated on the area of a potential object (red feature vector), although no particular information like bounding box is given. This exemplifies that our design of computation in Eq. (1) can amplify attention weights on similar features and attenuate the attention on background. Lead by attention, memories (#1 and #2) are propagated to the potential object area, convincing the detector to recognize this object as a hamster. We also report the GPU usage and memory size per frame in this video (#131000). The memory size (green line) varies. In most time, it is less than the size of a feature map (dashed green line). We can observe that *write* operation consistently erases redundant features to clear up memory. More hard examples are shown in Figure 3(b). The importance of long term information is also addressed in these examples, because in some examples the memories are extracted more than 100 time steps before the current frame.

4.3. Ablation Study

Performance of different memory. Table 1 compares the performance of methods incorporating different level memories, M_{pix} and M_{inst}, on different object motion speeds. The category of object motion speeds follows FGFA [45]. Given the intersection-over-union (IOU) of the same objects in nearby frames (± 10 frames), an object is considered in fast motion when the IOU is larger than 0.9, in slow motion when the IOU is smaller than 0.7, and in medium motion when between 0.7 and 0.9. Four methods with/without *pixel*/*instance* memory are compared.

Method (a) is the single frame baseline detector R-FCN.

Methods		(a)	(b)	(c)	(d)
With M_{pix}			✓		✓
With M_{inst}				✓	✓
mAP (%)	*overall*	73.8	$78.8_{\uparrow 5.0}$	$74.8_{\uparrow 1.0}$	$\mathbf{79.3}_{\uparrow \mathbf{5.5}}$
	slow	82.9	$85.8_{\uparrow 2.9}$	$83.1_{\uparrow 0.2}$	$\mathbf{86.2}_{\uparrow \mathbf{3.3}}$
	medium	72.4	$78.1_{\uparrow 5.7}$	$73.4_{\uparrow 1.0}$	$\mathbf{78.7}_{\uparrow \mathbf{6.3}}$
	fast	52.4	$60.0_{\uparrow 7.6}$	$53.3_{\uparrow 0.9}$	$\mathbf{61.1}_{\uparrow \mathbf{8.7}}$
Runtime (ms)		72	99	77	112

Table 1: Performance with *pixel* and *instance* memory.

Method (b) incorporates only M_{pix}. The performance gain shows that the object guided *write* operation design is effective. The features selected from detected bounding boxes provide rich temporal information to boost the detection performance. Besides the overall mAP, large improvements can be observed in fast moving objects. *Method (c)* incorporates only M_{inst}. Memory is updated every 10 frames. It suggests that roi-pooled sensitive-score maps $\{R_i, \cdots, R_n\}$ are also able to propagate temporal information and improve the performance of video object detection. *Method (d)* incorporates both M_{pix} and M_{inst}. With different memories, we have our highest performance. Compared to *method (a)*, *instance* features bring more performance gain in mAP (slow) and map (fast) for *method (b)*. This shows that *pixel* and *instance* features are complementary to each other in temporal propagation.

In all, our proposed method is capable of managing valuable temporal information and propagating them accurately

Non-key size	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$
With memory	✗	✓	✓	✓
mAP (%)	73.8	$76.8_{\uparrow 2.0}$	$74.4_{\uparrow 0.6}$	$71.5_{\downarrow 2.3}$
Runtime (ms)	72	67	59	52

Table 2: Speed-accuracy tradeoff using down-sample non-key frames. Both M_{pix} and M_{inst} are used.

Method	With DCN [6]	Memory size (#feature map)	mAP (%)
R-FCN [5]	✗	0	73.8
FGFA [45]		20	76.3
LWDN [43]		1	76.3
D&T [8]		1	75.8
MAN [35]		12	78.1
THP [44]	✓	1	78.6
STSN [2]		26	78.9
Ours	✗	~ 0.9	**79.3**
Ours_{def}	✓	~ 0.8	**80.0**

Table 3: Comparison with the state-of-the-art. All methods use ResNet-101 [15] as the feature network, and methods denoted with DCN use deformable units [6] to enhance the ResNet-101. The memory size denotes the number of feature maps to be aggregated for one inference. For our methods, it is calculated as the average number of feature vectors in M_{pix} and M_{inst} divided by the number of locations in a feature map.

to other frames, while *pixel* and *instance* level features are complementary for the performance. Besides the large improvement in mAP, our framework has similar inference time with the single frame baseline.

Efficient detection and speed-accuracy tradeoff. Table 2 reports the speed-accuracy tradeoff of our framework. A key frame interval is set to 5 for the experiments shown in Table 2, where full key frames are used to update memory and non-key frames are down-sampled to reduce inference time. Both *pixel* and *instance* level features are incorporated. The results shown in Table 2 suggest that object guided external memory network is well adapted to different scales of down-sampled inputs. High quality features stored in the external memory can propagate useful information to down-sampled frames and achieve good speed-accuracy tradeoff. When non-key frames are down-sampled to $1/2$ and $1/3$ of the original size, the mAP is better than the baseline R-FCN with full size inputs ($3.0\% \uparrow$, $0.6\% \uparrow$) and with less inference time (5 ms$\downarrow$ and 13 ms$\downarrow$). Specifically, our method on half sized inputs has better mAP than FGFA [45] on full size input images ($0.5\% \uparrow$) with significantly less inference time (67 ms and 246 ms on our testing machine).

Methods		FGFA[45]	MAN[35]	Ours
mAP (%)	*overall*	76.3	78.1	**79.3**
	slow	83.5	**86.9**	86.2
	medium	75.8	76.8	**78.7**
	fast	57.6	56.7	**61.1**

Table 4: Performance comparison on different motion speeds with flow-based methods.

4.4. Comparison with state-of-the-art results

Overall performance. We compare the overall performance with the state-of-the-art feature level methods. We report results on the ImageNet VID validation set, as in [45, 44, 8, 35, 2, 39, 43]. Tabel 3 presents the results. To show the generalization ability of our method and for fair comparison, following [44, 2], we also train our framework by adding deformable units (DCN) [6] in R-FCN [5], denoted as Our_{def}. STMN [39] only reports the result after post-process (80.5%). For fair comparison, we also apply Seq-NMS [12] to our method without and with DCN. The results are 80.8% and 81.6% in mAP, respectively. Among all methods in Table 3, our proposed framework has the best performance even without adding DCN as a stronger feature network, and without using extra information from future frames. Meanwhile, on average, our framework has less memory size compared with the other methods that directly store multiple feature maps. See details in Table 3.

Comparison on different motion speeds. We also compare on different motion speeds with the two flow-based methods, FGFA [45] and MAN [35], as shown in Table 4. Clear improvements of mAP are observed in fast motion ($3.5\% \uparrow$, $4.4\% \uparrow$) and medium motion ($2.9\% \uparrow$, $1.9\% \uparrow$). The improvements show the advantage of *read* operation in feature alignment over large object displacement. Flow estimation given by FlowNet [7] may fail when feature displacement between two frame is large, leading to error alignment of nearby frames. While by our attentional *read* operation, spatial position of features is no barrier for the alignment, leading to accurate feature propagation across large frame content displacement.

5. Conclusions

We propose a novel object guided external memory network to target the drawbacks of methods that use internal memory in video object detection. Storage-efficiency is handled by object guided hard attention and long-term dependency is protected in addressable external memory. Novel *read*/*write* operations guided by soft/hard attention are designed to access multi-level external memory. Moreover, we visualize the external memory to show the detailed temporal inference process. We evaluate our method on the ImageNet VID dataset and achieve state-of-the-art performance and good speed-accuracy tradeoff.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473, 2014.

[2] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *ECCV*, 2018.

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

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

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

[8] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *CVPR*, 2017.

[9] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[11] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

[12] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

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

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. *arXiv preprint arXiv:1711.11575*, 2017.

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[19] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2018.

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

[21] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

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

[24] Hao Luo, Wenxuan Xie, Xinggang Wang, and Wenjun Zeng. Detect or track: Towards cost-effective video object detection/tracking. In *AAAI*, 2018.

[25] Mateusz Malinowski, Carl Doersch, Adam Santoro, and Peter Battaglia. Learning visual question answering by bootstrapping hard attention. In *ECCV*, 2018.

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

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

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[29] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, 2015.

[32] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

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

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

[36] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[37] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[38] Chien-Sheng Wu, Richard Socher, and Caiming Xiong. Global-to-local memory pointer networks for task-oriented dialogue. *arXiv preprint arXiv:1901.04713*, 2019.

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

[40] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[41] Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*, 2015.

[42] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[43] Jiang Zhengkai, Gao Peng, Guo Chaoxu, Zhang Qian, Xiang Shiming, and Pan Chunhong. Video object detection with locally-weighted deformable neighbors. In *AAAI*, 2018.

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

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

[46] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, 2017.