

# Fast Object Detection in Compressed Video

Shiyao Wang  
Tsinghua University

Hongchao Lu  
Tsinghua University

Zhidong Deng  
Tsinghua University

{sy-wang14, luhc15, michael}@mail.tsinghua.edu.cn

## Abstract

Object detection in videos has drawn increasing attention recently since it is more important in real scenarios. Most of the deep learning methods for video analysis use convolutional neural networks designed for image-wise parsing in a video stream. But they usually ignore the fact that a video is generally stored and transmitted in a compressed data format. In this paper, we propose a fast object detection model that incorporates light-weight motion-aided memory network (MMNet), which can be directly used for H.264 compressed video. MMNet has two major advantages: 1) For a group of successive pictures (GOP) in a compressed video stream, it runs the heavy computational network for I-frames, i.e. a few reference frames in videos, while a light-weight memory network is designed to generate features for prediction frames called P-frames; 2) Unlike establishing an additional network to explicitly model motion among frames, we directly take full advantage of both motion vectors and residual errors that are all encoded in a compressed video. Such signals maintain spatial variations and are freely available. To our best knowledge, the MMNet is the first work that explores a convolutional detector on a compressed video and a motion-based memory in order to achieve significant speedup. Our model is evaluated on the large-scale ImageNet VID dataset, and the results show that it is about 3x times faster than single image detector R-FCN and 10x times faster than high performance detectors like FGFA and MANet.

## 1. Introduction

Video is viewed as one of the next frontiers in computer vision, since it takes up above 70 percent of all internet traffics. In the past four years, deep learning has made historic progress in still image analysis. A lot of deep convolutional neural network based object detection methods have been proposed, including [30, 7, 27, 29, 24, 25]. Although there

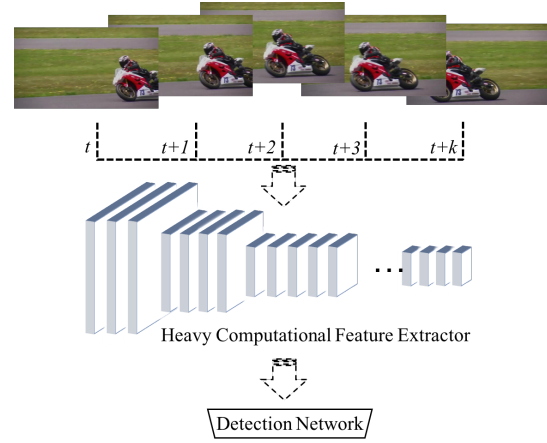


Figure 1. Most previous methods for video analysis use heavy computational networks to extract features frame by frame. They usually ignore the fact that redundant frames are stored and transmitted in a compressed data format.

has great success in static image object detection, it still remains challenging for accurate and efficient video detection task. Frames in videos are usually deteriorated due to motion blur or video defocus. On the other hand, instead of unaffordable computational model, an efficient detection model with low computational cost plays a crucial role in high-value scenarios, e.g., autonomous driving and video surveillance. Consequently, one of the main-stream methods focuses on improving frame-wise detection results [47, 40, 44]. Those methods effectively improve final performance, although computational redundancy of per-frame training and evaluation are still huge.

Another main-stream approach aims to speed up video object detection according to high temporal redundancy (see Figure 1). Majority of the detection frameworks comprise two major parts: feature extractor and detection network. ResNet101 [14] is often regarded as a popular backbone used to extract features. The features are then fed

into detection networks like Faster R-CNN [30] and R-FCN [7]. With respect to feature extractor, detection network is simpler in terms of computational complexity. Hence, [48] proposes a method to run expensive feature extractors only on sparse key frames and then propagates resulting deep feature maps to other frames. The key idea for propagation between frames is to calculate pixel-wise displacements through a flow estimation network [9]. Since the flow estimation network has a lower computational burden than feature extractor, it is able to help relieve computational bottleneck. [46] combines the feature aggregation [47] and the propagation [48] to make appropriate trade-offs. But the flow based motion estimation takes extra time for calculating displacements since the FlowNet still consists of more than 10 convolutional layers.

In this paper, we propose a fast and accurate object detection method for compressed videos. Most of the previous works focus on decoding video into frames and exploiting different techniques to retrieve motion cues, which is time-consuming and cumbersome. As a result, the proposed method attempts to directly process video streams. In fact, the frames in video can be divided into three types: I (inter)-frame, P (predicted)-frame, and B (bidirectional predicted)-frame. When encountering any I-frame, it runs an expensive feature network to extract intermediate features. In the period of the following GOP (group of pictures), we only use propagation from I-frames to have features of P/B-frames. Our method is different from the previous work, since both motion vectors and residual errors are employed to calibrate features rather than setting up an additional network for retrieving motion information.

In summary, the contributions of this paper include:

- We replace the flow estimation network by exploiting the motion signals which have already been encoded in the compressed video. Note that they retain necessary motion cues and are freely available.
- We propose a new motion-based memory network (MMNet) to quickly generate inter-frame features. This memory unit is utilized to refine features and filter out any unnecessary contents. It only comprises one convolutional layer.
- Our model is evaluated on the large-scale ImageNet VID dataset [31], and the results show that it is about 3x times faster than a single image detector R-FCN [7] and 10x times faster than high performance detectors like FGFA and MANet.

## 2. Related work

### 2.1. Object detection

#### 2.1.1 Object detection from still images

State-of-the-art methods for general object detection consist of feature networks [23, 33, 36, 14, 35, 17, 6] and detection

networks [12, 11, 30, 7, 24, 8, 32]. [12] is a typical proposal based detector which uses extracted proposals [38]. Faster R-CNN [30] further integrates proposal generation step into CNNs, resulting in a much higher proposal quality as well as the computation speed. R-FCN [7] takes advantage of region proposals, and eliminates per-ROI computation by moving the ROI-pooling operation to the end of the network, which has comparable performance and higher speed compared to Faster R-CNN. We use R-FCN as our baseline and its computation speed is further improved for video object detection.

#### 2.1.2 Object detection in videos

An object detector for videos is different from one for still images because it also considers the temporal information. One of the main-stream approaches is based on per-frame complete detection and improves the detection quality by leveraging temporal coherence. And other previous works tried to speedup the computation through using temporal redundancy.

**For high performance**, [47, 19, 40, 44, 46, 3] propose end-to-end learning models to enhance the features of individual frame in videos. They use different techniques to align the features. [47, 40, 46] presents flow-guided feature [9] aggregation to leverage temporal coherence on the feature level. [19] provides a novel tubelet proposal network to efficiently generate spatiotemporal proposals. [44] computes the correlation between neighboring frames and introduces a memory module to aggregate their features. [3] uses deformable convolutions across time to align the features from the adjacent frames. [3, 20, 10] are based on detected bounding box rather than feature-level aggregation. [13, 21, 20] propose mapping strategies of linking still image detections to cross-frame box sequences in order to boost scores of weaker detections within the same video. D&T [10] is the first work to joint learn ROI tracker along with detector and the tracker is also exploited to link the cross-frame boxes. All of these mentioned works achieve high detection performance however they use a computationally expensive feature extractor for generating per-frame features.

**For fast inference**, [48] utilizes optical flow network for calculating pixel-level correspondence and propagating deep feature maps from key frames to other frames. The flow estimation and feature propagation are faster than feature networks. Thus, significant speedup is achieved.[5] introduces temporal propagation on box-level. They first produce bounding boxes on key frames, and then generate boxes of other frames through a coarse-to-fine network. [26] propagates feature maps across frames via a convolutional LSTM. They only use the appearance features without explicitly capturing motion cues. Although their model

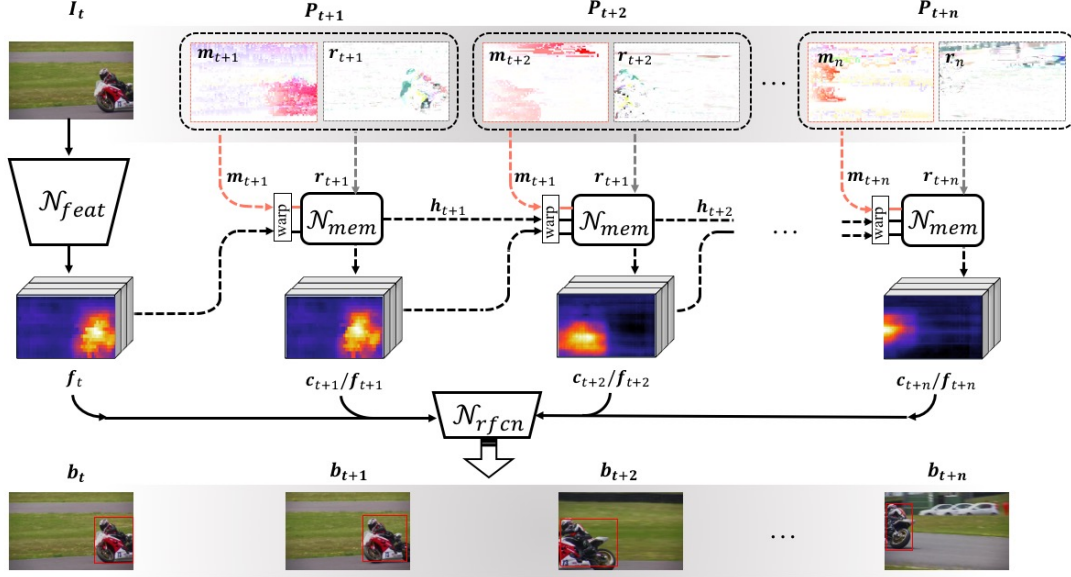


Figure 2. The overall framework of the proposed Motion-aided Memory Network. The feature extractor  $\mathcal{N}_{feat}$  only runs on reference frame  $I_t$ , and the other features of  $P_{t+k}$  are generated by the memory network  $\mathcal{N}_{mem}$ . Motion vectors  $m_{t+k}$  and residual errors  $r_{t+k}$  are fed into memory network in order to provide motion cues. Finally, all the features in one GOP(group of pictures) are transferred to detection network  $\mathcal{N}_{rfcn}$ , producing bounding boxes simultaneously.

is faster than existing methods, the performance is much degraded. Moreover, [41, 16] also focus on model accelerating. However, they aim to build light weight deep neural networks which are unrelated to any specific tasks.

## 2.2. Deep learning model on compressed videos

H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) [34] is one of the most commonly used formats for recording, compression, and distribution of videos. It is a block-oriented motion-compensation-based video compression standard. To our knowledge, only a few prior works applied deep models directly on compressed videos.[22, 37] utilize signals from compressed video to produce non-deep features. [42] resembles our model the most and they aim to improve both speed and performance on video action recognition which focuses on producing video-level features. But the video object detection needs to produce per-frame bounding boxes which has per-frame feature quality requirements. [1] approximates a location of a target object in the current frame by using MVs from HEVC bit stream. It is related to box-level movements prediction which is complementary to our feature-level prediction.

## 3. Approach

### 3.1. Overview

The proposed motion-aided memory network is presented in Figure 2.

For the input video, we use H.264 as illustration since these compression techniques that leverage successive frames are usually similar. There are three types of frames (or pictures) are used in video compression: I, P, and B-frames. An I-frame (denoted as  $I_t \in \mathbb{R}^{h \times w \times 3}$ ) is a complete image.  $h$  and  $w$  are the height and width of the original frame. P and B frames are also known as deltaframes, denoted as  $P_{t+k}$  and  $B_{t+k}$ , respectively. They can be reconstructed based on offsets, called motion vectors  $m_{t+k}$  and residual error  $r_{t+k}$ . They are computed by comparing the  $I_t$  and  $P_{t+k}$ . In Figure 2, we show a typical GOP on the top line, denoted as  $\{I_t, P_{t+1}, \dots, P_{t+k}, \dots, P_{t+n}\}$ . A B-frame can be regarded as a special P-frame, whose motion vectors are computed bi-directionally. For encoded videos, we use only backward references, namely I and P frames.

For the core modules, there are three networks in the core module: feature extractor, memory unit and detection network, indicated as  $\mathcal{N}_{feat}$ ,  $\mathcal{N}_{mem}$  and  $\mathcal{N}_{rfcn}$ , respectively. The I-frame  $I_t$  is fed to the  $\mathcal{N}_{feat}$  in order to generate semantic features  $f_t \in \mathbb{R}^{\frac{h}{16} \times \frac{w}{16} \times c}$ . Since the feature extractor usually composes poolings, the dimension of height and width are reduced by stride 16, and  $c$  is the number of chan-

nels. The features of following frames  $\mathbf{f}_{t+k}$  within the same GOP are fast produced by the light-weight motion-aided memory network  $\mathcal{N}_{mem}$ . It receives motion vectors  $\mathbf{m}_{t+k}$  and residual errors  $\mathbf{r}_{t+k}$  as input and output features of the current frame  $\mathbf{f}_{t+k}$ . Note that  $\mathbf{m}_{t+k}$  and  $\mathbf{r}_{t+k}$  are also respectively resized to  $(\frac{h}{16} \times \frac{w}{16} \times 2)$  and  $(\frac{h}{16} \times \frac{w}{16} \times 3)$  in order to match the dimension of appearance features. The above procedure is formulated as:

$$\mathbf{f}_t = \mathcal{N}_{feat}(\mathbf{I}_t) \quad (1)$$

$$\mathbf{f}_{t+k} = \mathcal{N}_{mem}(\mathbf{f}_{t+k-1}, \mathbf{m}_t, \mathbf{r}_t) \quad s.t. \quad k > 1 \quad (2)$$

$$[b_t, b_{t+1}, \dots, b_{t+n}] = \mathcal{N}_{rcn}([\mathbf{f}_t, \mathbf{f}_{t+1}, \dots, \mathbf{f}_{t+n}]) \quad (3)$$

where  $[\mathbf{f}_t, \mathbf{f}_{t+1}, \dots, \mathbf{f}_{t+n}]$  denotes the concatenation of the features of a GOP. It means  $\mathcal{N}_{rcn}$  will receive the features of the whole GOP, and predict their bounding boxes simultaneously.

Previous works [45] indicates that motion vectors  $\mathbf{m}_{t+k}$  only represent movements for a macroblock (e.g.  $16 \times 16$ ) in a picture. It is based on the position of this macroblock in reference frames. They lack refined motion information of pixels. But we use these vectors on feature-level propagation, where features are also downsampled to  $\frac{h}{16} \times \frac{w}{16}$  in terms of the original images. So the macroblock-level movements are reasonable for our feature propagation.

### 3.2. Light-weight Motion-aided Memory Network

In this subsection, the light-weight motion-aided memory network is introduced to propagate the features across frames. We use LSTM [15] to transfer the features. There are two modifications to the conventional LSTMs. One is the motion vector aided feature warping and another is residual error based new input.

Although the gates in LSTM focus on selecting and updating representations, it is hard for the memory to forget the object after it has moved to a different position [44]. Experiments in Section 4.4 prove this concern. It is called mis-aligned features across frames. Hence, we propose a motion vector based feature warping which helps to calibrate cell/hidden features before running through the memory module (see Figure 3). We warp the feature maps from the neighboring frames to the current frame as follows:

$$\begin{aligned} \mathbf{c}_{t+k-1 \rightarrow t+k} &= \mathcal{W}(\mathbf{c}_{t+k-1}, \mathbf{m}_{t+k}) \\ \mathbf{h}_{t+k-1 \rightarrow t+k} &= \mathcal{W}(\mathbf{h}_{t+k-1}, \mathbf{m}_{t+k}) \end{aligned} \quad (4)$$

where  $\mathbf{c}_{t+k-1}$  and  $\mathbf{h}_{t+k-1}$  are outputs of the memory module at time  $t+k-1$ . We set  $\mathbf{c}_t$  and  $\mathbf{h}_t$  to  $\mathbf{f}_t$ . The warping operation  $\mathcal{W}$  is implemented by bi-linear function which is applied on each location for all feature maps. It projects a location  $\mathbf{p} + \Delta\mathbf{p}$  in the frame  $t+k-1$  to the location  $\mathbf{p}$  in

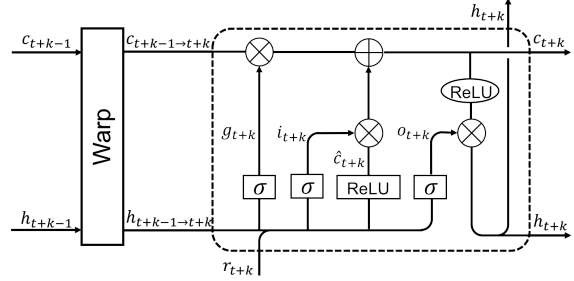


Figure 3. Light-weight Motion-aided Memory Network. Motion vectors are used to calibrate cell/hidden features before they run through the memory module. Residual errors are employed to correct appearance errors.

the frame  $t+k$  which can be formulated as:

$$\begin{aligned} \Delta\mathbf{p} &= \mathbf{m}_{t+k}(\mathbf{p}) \\ \mathbf{c}_{t+k-1 \rightarrow t+k}(\mathbf{p}) &= \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p} + \Delta\mathbf{p}) \mathbf{c}_{t+k-1}(\mathbf{q}) \end{aligned} \quad (5)$$

where  $\Delta\mathbf{p}$  is obtained through  $\mathbf{m}_{t+k}$ .  $\mathbf{q}$  enumerates all spatial locations in the feature maps  $\mathbf{c}_{t+k-1}$ , and  $G(\cdot)$  denotes bi-linear interpolation kernel as follow:

$$G(\mathbf{q}, \mathbf{p} + \Delta\mathbf{p}) = \max(0, 1 - \|\mathbf{q} - (\mathbf{p} + \Delta\mathbf{p})\|) \quad (6)$$

Hidden features  $\mathbf{h}_{t+k-1 \rightarrow t+k}$  can also be obtained through above operations. Then  $\mathbf{c}_{t+k-1 \rightarrow t+k}$  and  $\mathbf{h}_{t+k-1 \rightarrow t+k}$  are used as the input from previous time to the current memory module.

For conventional LSTMs, the current complete frame will be used as the new information. In our model, we use the residual errors as new input. Through the motion vector, the previous features can be matched to the current state, but it still exists the differences between them. So the video encoder computes the residual errors, whose values are known as the prediction error and need to be transformed and sent to the decoder. After spatial alignment, the residual errors can be used as the complementary information which is more crucial than the whole appearance features of the complete image. In order to better match the residual errors from image-level to the feature-level, we use one convolutional layer to rescale the values.

After obtaining the warped features and new input, the memory can generate the new cell features as follow:

$$\begin{aligned} \mathbf{g}_{t+k} &= \sigma(\phi_g(\mathbf{h}_{t+k-1 \rightarrow t+k}, \mathbf{r}_{t+k})), \\ \mathbf{i}_{t+k} &= \sigma(\phi_i(\mathbf{h}_{t+k-1 \rightarrow t+k}, \mathbf{r}_{t+k})), \\ \hat{\mathbf{c}}_{t+k} &= \text{ReLU}(\phi_c(\mathbf{h}_{t+k-1 \rightarrow t+k}, \mathbf{r}_{t+k})), \\ \mathbf{c}_{t+k} &= \mathbf{g}_{t+k} \otimes \mathbf{c}_{t+k-1 \rightarrow t+k} + \mathbf{i}_{t+k} \otimes \hat{\mathbf{c}}_{t+k} \end{aligned} \quad (7)$$

where  $\oplus$  and  $\otimes$  are element-wise addition and multiplication.  $\mathbf{g}_{t+k}$  can be regarded as selection mask in order to decide which previous features would go to store in next representations.  $\hat{\mathbf{c}}_{t+k}$  is new information that carried com-

plementary representations.  $\mathbf{c}_{t+k}$  is regarded as features for the current frame  $\mathbf{f}_{t+k}$ . Then the hidden features can be generated:

$$\begin{aligned} \mathbf{o}_{t+k} &= \sigma(\phi_o(\mathbf{h}_{t+k-1 \rightarrow t+k}, \mathbf{r}_{t+k})), \\ \mathbf{h}_{t+k} &= \mathbf{o}_{t+k} \otimes \text{ReLU}(\mathbf{c}_{t+k}) \end{aligned} \quad (8)$$

Based on this architecture, we can transform former features to the current state and they will be passed to the next step until encountering another new I-frame.

### 3.3. Gate Normalization

The activation functions in the memory network are sigmoid and ReLU [28]. Though Tanh is commonly used in LSTMs, we choose ReLU activations in our module. Since the memory is integrated into convolutional layers, we find it is important not to alter the bounds of the feature maps. It may due to the fact that ReLU nonlinearity is employed in the R-FCN convolutional layers. In order to make the training procedure more stable, we propose a normalization method along the gate dimension, called gate normalization. It is formulated as:

$$\hat{\mathbf{x}}_j = \frac{1}{\sigma_j}(\mathbf{x}_j - \mu_j) \quad (9)$$

Here  $\mathbf{x}$  are the features computed in the memory network. And  $j$  is the index along the channel axis.  $\mu$  and  $\sigma$  in Equation 9 are the mean and standard deviation (std) computed by:

$$\begin{aligned} \mu_i &= \frac{1}{m} \sum_{l \in \mathcal{S}_j} \mathbf{x}_l, \quad \sigma_i = \sqrt{\frac{1}{m} \sum_{l \in \mathcal{S}_j} (\mathbf{x}_l - \mu_i)^2 + \epsilon}, \\ \mathcal{S}_j &= \{\mathbf{x}_{(j,g)}, \mathbf{x}_{(j,i)}, \mathbf{x}_{(j,c)}, \mathbf{x}_{(j,o)}\} \end{aligned} \quad (10)$$

where  $\mathcal{S}_j$  is the set of pixels and  $m$  is the size of this set.  $\mathcal{S}_j$  consists of features computed through convolutional layers in gates  $\phi_g(\cdot)$ ,  $\phi_i(\cdot)$ ,  $\phi_c(\cdot)$  and  $\phi_o(\cdot)$  of Equation 7, denoted as  $\mathbf{x}_{(j,g)}$ ,  $\mathbf{x}_{(j,i)}$ ,  $\mathbf{x}_{(j,c)}$ ,  $\mathbf{x}_{(j,o)}$ . It means pixels from the four gates sharing the same channel index are normalized together, resulting the gate normalization.

**Relation to prior work.** Typical normalizations applied for CNN are Batch Norm [18], Layer Norm[2], Instance Norm[39] and Group Norm[43]. For the specific time  $t+k$ , one GPU usually holds only one image which limits BNs usage. Group norm divides the channels into groups and computes mean and variance within each group. If the group number is set to 1, it becomes layer norm. If the group number is set to the number of channels, it becomes the instance norm. If a group consists of the  $j$ -th channel features from different gates, it becomes the proposed gate normalization. The above normalizations assume features of one group make ‘‘similar contribution’’ inspired by different motivation. The gates within a memory network should not be independent of each other. By contrary, they can communicate and compete with the others. So we norm

features from different gates by the above equations.

Similar to the batch norm, we use additional learnable parameters  $\alpha$  and  $\beta$  in order to avoid reducing the representational power of the layer:

$$\hat{\mathbf{x}}_j = \alpha_j \hat{\mathbf{x}}_j + \beta_j \quad (11)$$

where  $\alpha$  and  $\beta$  are used to scale and shift the data.

## 4. Experiments

### 4.1. Dataset sampling and evaluation metrics

We evaluate the proposed MMNet on the ImageNet [31] object detection from video (VID) dataset. It is split into 3862 training and 555 validation videos. VID contains 30 classes labeled with ground truth bounding boxes on all video frames. We report the evaluation of previous state-of-the-art models on the validation set and use mean average precision (mAP) as the evaluation metric by following the protocols in [47, 48, 19, 48, 47].

The 30 object categories in ImageNet VID are a subset of 200 categories in the ImageNet DET dataset. We use the mixture of DET and VID as our training data since the redundancy among video frames make the training procedure less efficient. So we follow previous approaches and train our model on an intersection of ImageNet VID and DET set - 30 categories. To sum up, we use frames from each video in VID dataset and at most 2K images per class from DET dataset as our training samples.

### 4.2. Training and Evaluation

We perform two phase training by using SGO optimization with momentum of 0.9. In the first phase, the model is trained on the mixture of DET and VID for 12K iterations, with learning rates of  $2.5 \times 10^{-4}$  and  $2.5 \times 10^{-5}$  in the first 80K and 40K iterations, respectively. We use a batch size of 4 on 4GPUs, with each GPU holding one mini-batch. In the second phase, the motion-aided memory network is integrated into R-FCN for another epoch on VID dataset in order to learn fast feature prediction. In this phase, each GPU holds multiple samples in one GOP. It is already introduced by Section 3.1. The feature extractor ResNet101 model is pre-trained for ImageNet classification as default. In both training and testing, we use single scale images with shorter dimension of 600 pixels. For testing we run the whole networks only on I-frame and fast predict the bounding boxes in the rest frames. Non-maximum suppression (NMS) is applied with intersection-over-union (IoU) threshold 0.7 in RPN and 0.4 on the scored and regressed proposals.

### 4.3. Ablation Study

In this section, we conduct an ablation study to prove the effectiveness of the proposed network, including motion vectors, residual errors and memory module. We use

| Feature Extractor | ResNet-101 |      |      |
|-------------------|------------|------|------|
| Methods           | (a)        | (b)  | (c)  |
| MV+Res?           |            | ✓    | ✓    |
| LSTM?             | ✓          |      | ✓    |
| mAP(%)            | 66.3       | 70.3 | 72.1 |
| mAP(%) (fast)     | 27.7       | 38.5 | 44.2 |
| mAP(%) (medium)   | 68.2       | 71.2 | 72.0 |
| mAP(%) (slow)     | 82.6       | 83.5 | 83.6 |

Table 1. Accuracy of different methods on ImageNet VID validation, using ResNet-101 feature extraction networks.

ResNet-101 to extract I-frame features and adopt three different ways to propagate features to the following P-frames. The evaluation protocols follow the previous work [47]. They divide the ground truth objects into three groups according to their motion speed. They use object’ averaged intersection-over-union(IoU) scores with its corresponding instances in the nearby frames as measurement. It indicates that the lower the motion IoU ( $< 0.7$ ) is, the faster the object moves. Otherwise, the larger Motion IoU ( $score > 0.9$ ) denotes the object moves slowly. The rest is medium speed. Figure 1 also shows the runtime speed for these models.

**Method (a)** adopts LSTMs to transform features of reference frame to the following frames. LSTMs are utilized to encode both spatial and temporal information automatically. However, without explicit motion cues, LSTMs are unable to automatically align the temporal context from previous frames, leading to poor performance (66.3% in Table 1), especially for fast moving objects (27.7%).

**Method (b)** uses the motion vector and residual error to align features across time. Motion vector is used to warp the features. Residual errors are rescaled by one convolutional layer and directly added to the warped features. The parameters of this setting is smaller than method (a) because it requires no extra calculation for different gates. It can achieve much better performance when using motion guidance. And for the fast moving objects, the result is improved from 27.7% to 38.5 %.

**Method (c)** is the proposed motion-aided memory network. It also aligns features by motion vector among consecutive frames, and finally corrects them through residual errors. Compared to Method (b), It utilizes the LSTMs to learn filtering and updating the features. In addition to the spatial variations caused by motion, the gate functions focus on automatically refine and enhance the crucial features by end-to-end learning. Compared to method (b), it improves the overall performance by 1.8% and achieves 5.7% better

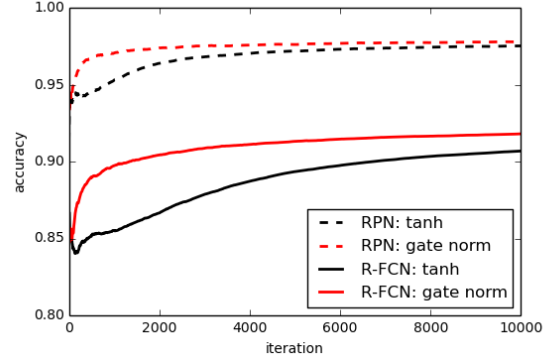


Figure 4. The convergence curves of RPN accuracy and R-FCN accuracy. The Gate Normalization based memory has faster convergence and obtain better accuracy.

result for fast moving objects.

**Gate Normalization** is proposed to accelerate training convergence. Figure 4 shows the convergence curves of RPN and R-FCN accuracy. Since we use two-stage detection, proposal network RPN is used to distinguish the object and background, and the detection network R-FCN is utilized to classify different categories. The memory network which uses the gate normalization and ReLU activation function has fast convergence and obtain better accuracy. This comparison proves the effectiveness of the proposed gate normalization integrated by ReLU.

To sum up, the motion vector and residual errors are necessary for modeling the motion among consecutive frames. They can speedup the detection procedure. Besides, memory network is employed to filter out unnecessary information, and complement new information through residual errors. The combination of these two core modules is capable of promoting the final feature representations collabora-

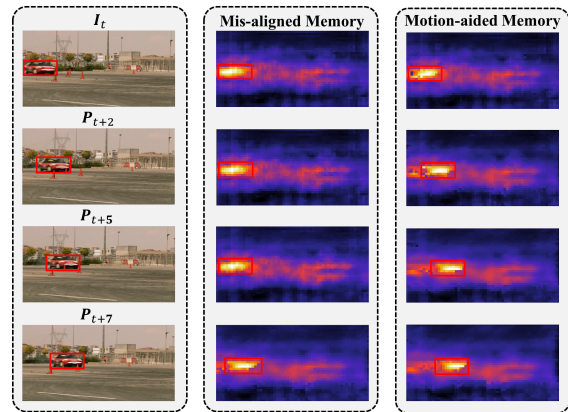


Figure 5. Memory visualization. Motion information is quite necessary for feature propagation. It helps MMNet align the feature when the objects move to a different position.



tively and reduce the runtime speed as well.

#### 4.4. Visualization

**Visualization of Memory.** We attempt to take a deeper look at intermediate features learned by motion-aided memory network. In Figure 5, the left part consists of decoded frames  $\{I_t, \dots, P_{t+2}, \dots, P_{t+5}, \dots, P_{t+7}\}$ . The car in video moves from left to the middle. We compare the visualization results of mis-aligned and motion-aided memory.

The features in the middle part are learned by LSTM. Although the gates within LSTM are designed for learning the changes among historical information and new input, it is incapable of aligning the spatial features across frames. It cannot capture motion cues only depended on appearance features. The right part presents the motion-aided memory network. Features of  $\{P_{t+2}, P_{t+5}, P_{t+7}\}$  are all based on  $I_t$ . The MMNet receives motion vectors and residual errors as input, warps the features of reference frame to the current states and correct them by residual errors. The neural of high response in the heatmap moves from the left to the middle as the original car.

From the above comparison, motion information is extremely important for feature propagation. It helps to align features when the objects move to a different position. Thus the motion-aided memory network can calibrate features and alleviate inaccurate localizations.

**Visualization of FlowNet and Motion Vector.** In order

to show the differences of motion cues between the flow estimation and motion vectors, we visualize two examples and their results in Figure 6. Each of the two examples contains the original short snippet, results of FlowNet [48, 9] and motion vectors (We use the tool provided with Sintel[4] to visualize the above motion information).

The main advantage of motion vector is freely available. It requires no extra time or models to retrieve motion information because they have already been encoded in the compressed video. From the results of line 3 and 6 in Figure 6, even the motion vector is not as refined as FlowNet, it is enough to model the motion tendency of objects. All the features of frame  $P_{t+1}, P_{t+3}, P_{t+5}, P_{t+7}, P_{t+9}$  are propagated from the I-frame  $I_t$  by utilizing motion vectors, rather than using heavy computational network. Moreover, the bounding boxes location and recognition results are reasonable in terms of motion, and sometimes even exceed the flow estimation results.

For flow estimation, the motion information is more smooth. But this model composes more than 10 convolutional layers. For each neighboring frame, it should calculate the FlowNet first, which seems not elegant. The FlowNet has superior performance when the object is small and unclear (the bus in video #2). Since the motion vector is relative coarse, it cannot clearly capture the motion of the bus in the distance.

To sum up, the FlowNet is capable of building detailed

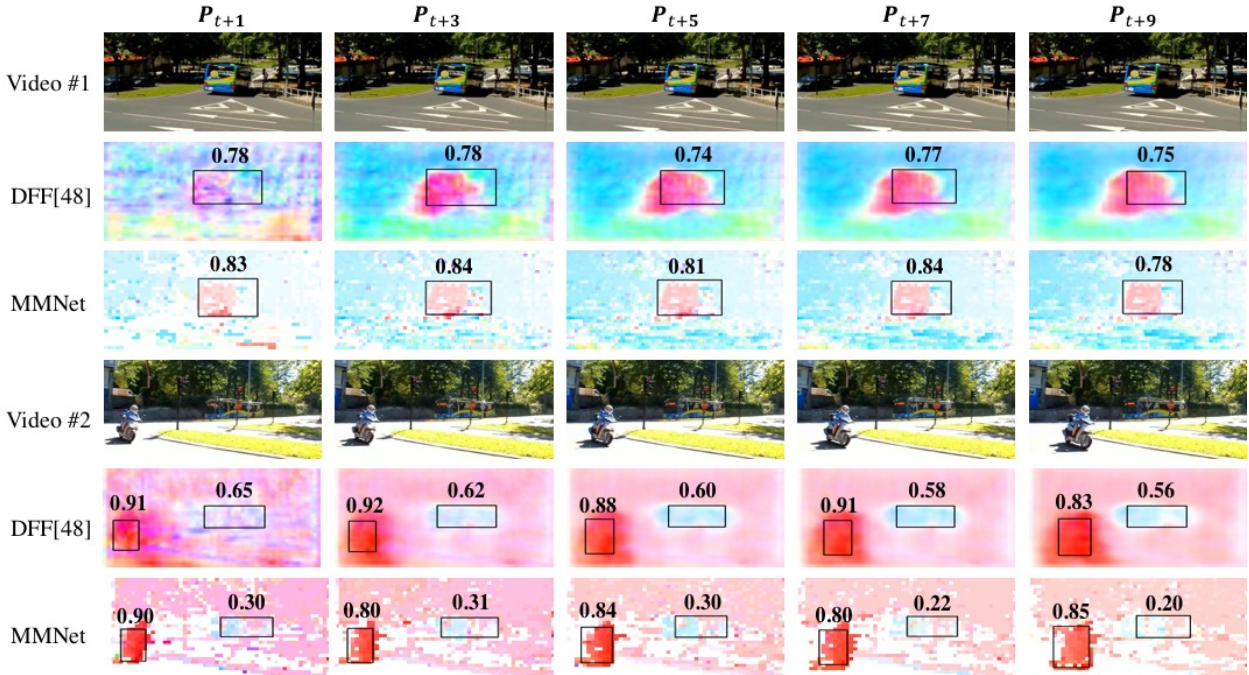


Figure 6. Visualization of FlowNet and Motion Vector. The FlowNet in [48] is capable of building detailed model information which is better for small objects (bus in video #2). And the motion vector can quickly provide the motion cues which helps to speed up detection procedure in most situations.

| Methods             | Method                | mAP(%) |
|---------------------|-----------------------|--------|
| Single Frame        | R-FCN[7]              | 73.6   |
| Box Propagation     | Scale-Time Lattice[5] | 77.8   |
|                     | TCNN[20]              | 73.8   |
|                     | Seq-NMS [13]          | 52.2   |
|                     | TCN[21]               | 47.5   |
| Feature Propagation | MANet[40]             | 78.1   |
|                     | FGFA[47]              | 76.5   |
|                     | DFF[48]               | 73.1   |
|                     | TPN[19]               | 68.4   |
|                     | Mobile[26]            | 54.4   |
|                     | <b>Ours (MMNet)</b>   | 72.1   |

Table 2. Performance comparison with state-of-the-art systems on the ImageNet VID validation set. The average precision (in %) for each class and the mean average precision over all classes is shown.

model information which is better for small objects. And the motion vector can quickly provide the motion cues which help to speed up detection procedure in most situations. This comparison shows the potential of compressed video based detection model. It fully exploits the encoded information and makes the model more elegant.

#### 4.5. Comparison with state-of-the-art systems

In this section, we show speed and performance of the related methods in Table 2. They can be divided into three groups: single frame baseline [7], box-level propagation and feature-level propagation. The box-level propagation methods can be regarded as post processing techniques which are complementary with feature-level propagation. So the we only present detailed comparison among feature propagation methods in Figure 7.

We select four most related works and list their speed and accuracy in Figure 7. They represent: high performance models [40, 47], single frame baseline [7] and fast detector [48]. [40] has the best performance among these previous works, whereas it takes about 260ms to detect for one frame. They use FlowNet to predict motion information among  $k$  nearby frames and aggregate them to enhance the reference frame. The performance and fps (frame per second) for varying  $k$  are listed in Figure 7. And  $k = 4, 8, 12, 16$ . Although their performance is higher, the computational cost is intolerable in practical scenarios and [47] is similar to this work. The second group is single frame detector R-FCN [7]. We believe the individual detection is redundant in videos, since consecutive frames have similar appearance, especially for objects which move slowly.

The last group is fast detector DFF [48] and our model. After producing features on a key frame, they are propa-

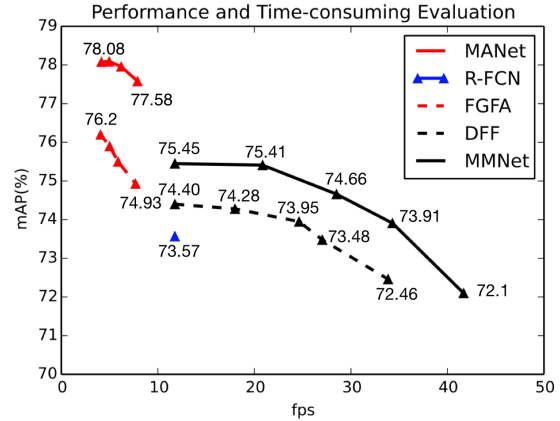


Figure 7. The detailed speed and accuracy of some typical methods. The runtime is measured on an NVIDIA Titan X Pascal GPU.

gated to  $k$  neighboring frames by flow estimation (for DFF) or motion vector (for ours),  $k \in [1, 3, 7, 11]$ . Compared with DFF, we believe that motion vector based propagation is more elegant than building another FlowNet to model the motion, and we also achieve better performance. All the models in Figure 7 use ResNet101[14] as key frame feature extractor and R-FCN [7] as the detection model.

## 5. Conclusions

In this paper, we propose a fast object detection model incorporating light motion-aided memory network called MMNet. It can be directly applied to compressed videos. Different from the previous work, we use motion and appearance information stored and transmitted within a video stream, rather than building another model to retrieve necessary motion cues. We use the I-frame as the reference frame, and explore the memory network to transfer features to next p-frames. The motion vector provides motion cues and residual error helps to correct appearance errors. All these operations are designed with respect to compressed videos. Although the MMNet cannot outperform the state-of-the-art models, it still shows the potential to speedup the detection framework.

## References

- [1] S. R. Alvar and I. V. Bajić. Mv-yolo: Motion vector-aided tracking by semantic object detection. *arXiv preprint arXiv:1805.00107*, 2018.
- [2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [3] G. Bertasius, L. Torresani, and J. Shi. Object detection in video with spatiotemporal sampling networks. *arXiv preprint arXiv:1803.05549*, 2018.



- [4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.
- [5] K. Chen, J. Wang, S. Yang, X. Zhang, Y. Xiong, C. C. Loy, and D. Lin. Optimizing video object detection via a scale-time lattice. *arXiv preprint arXiv:1804.05472*, 2018.
- [6] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *CoRR*, abs/1707.01629, 2017.
- [7] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.
- [8] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 764–773, 2017.
- [9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.
- [10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *International Conference on Computer Vision (ICCV)*, 2017.
- [11] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 580–587, 2014.
- [13] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269, 2017.
- [18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [19] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017.
- [20] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.
- [21] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 817–825, 2016.
- [22] V. Kantorov and I. Laptev. Efficient feature extraction, encoding, and classification for action recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 2593–2600, 2014.
- [23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [24] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 936–944, 2017.
- [25] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007, 2017.
- [26] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. *arXiv preprint arXiv:1711.06368*, 2017.
- [27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [28] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010.
- [29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [32] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 761–769, 2016.

- [33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [34] A. A. Sofokleous. Review: H.264 and MPEG-4 video compression: Video coding for next-generation multimedia. *Comput. J.*, 48(5):563, 2005.
- [35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4278–4284, 2017.
- [36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [37] B. U. Töreyin, A. E. Çetin, A. Aksay, and M. B. Akhan. Moving object detection in wavelet compressed video. *Sig. Proc.: Image Comm.*, 20(3):255–264, 2005.
- [38] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.
- [39] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.
- [40] S. Wang, Y. Zhou, J. Yan, and Z. Deng. Fully motion-aware network for video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 542–557, 2018.
- [41] Z. Wang, Z. Deng, and S. Wang. Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression. In *European Conference on Computer Vision*, pages 533–548. Springer, 2016.
- [42] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018.
- [43] Y. Wu and K. He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 3–19, 2018.
- [44] F. Xiao and Y. J. Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 485–501, 2018.
- [45] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2718–2726, 2016.
- [46] X. Zhu, J. Dai, L. Yuan, and Y. Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018.
- [47] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. 2017.
- [48] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. 2017.