Improving Video Object Detection by Seq-Bbox Matching

Hatem Belhassen[1,2,3], Heng Zhang[1], Virginie Fresse[2] and El-Bay Bourennane[3]
[1]*Beamtek SAS, 145 Rue Gustave Eiffel, 69330 Meyzieu, France*
[2]*Hubert Curien Laboratory, Jean Monnet University, Saint Etienne, France*
[3]*Univ of Burgundy - Franche-Comte, LE2I laboratory, France*

Keywords: Video Understanding, Real-time Video Object Detection, Online Object Detection.

Abstract: Video object detection has drawn more and more attention in recent years. Compared with object detection from image, object detection in video is more useful in many practical applications, e.g. self-driving cars, smart video surveillance, etc. It is highly required to build a fast, reliable and low-cost video-based object detection system for these applications. In this work, we propose a novel, simple and highly effective box-level post-processing method to improve the accuracy of video object detection. The proposed method is based on both online and an offline settings. Our experiments on ImageNet object detection from video (VID) dataset show that our method brings important accuracy gains, especially to more challenging fast-moving object detection, with quite light computational overhead in both settings. Applied to YOLOv3, our system achieves so far the best speed/accuracy trade-off for offline video object detection and competitive detection improvements for online object detection.

1 INTRODUCTION

Despite the great success of object detection from image, object detection in video is still challenging due to motion blur, rare pose, camera defocus, etc. State-of-the-art works focus on utilizing contextual and temporal information to overcome these effects. (Zhu and al., 2017) improved video object detection accuracy on two levels: box level and feature level. Even though feature-level methods are generally considered to achieve a higher improvement than box-level methods in terms of accuracy. But we have to consider that models optimized by feature level methods need to be trained on video-based datasets, which are more difficult to create, less available, and usually contain fewer object categories compared with image-based dataset. So, box-level methods are still important as they can be used as a complementary to feature-level methods, (e.g., Flow-Guided Feature Aggregation (Zhu and al., 2017) (FGFA for short) use box-level methods on the basis of their proposed feature-level method to improve video object detection accuracy). Moreover, box-level methods can be implemented on detectors trained on more common image-based dataset (e.g. Microsoft-COCO, Pascal VOC, etc), which makes them more universal and practical for industrial applications.

Existing box-level methods mostly rely on optical flow, object tracking (Kang and al., 2016) or heavy bounding box (bbox) across time sorting (Han and al., 2016). However, optical flow is naturally expensive in terms of computational complexity and has no direct relation with the main task of object detection. The use of object tracking adds a heavy computational overhead which makes the detection system less effective. As reported by (Han and al., 2016), bbox across time sorting (Han and al., 2016) is more effective than other box-level methods, while it still adds a heavy computational overhead and can not be applied to online video object detection.

Our work aims to design a novel, simple and much more effective box-level post-processing method. It can be applied to online or offline video object detection, to replace existing box-level methods. The principle is rescoring bboxes and linking tubelets (which are essentially sequences of associated bboxes across time) to avoid wrong or missed detections caused by low-quality frames in videos. Our proposed method, named *Seq-Bbox Matching*, is a simple and effective method to match bboxes across time to generate tubelets. The proposed following steps, named *Frame-level Bbox Rescoring* and *Tubelet-level Bbox Linking*, are applied to improve offline object detection. Then we modified our *Frame-level Bbox Rescoring* to adapt it to online object detection.

Experiments on ImageNet (Russakovsky and al., 2014) VID dataset prove the effectiveness of our method. It brings important accuracy gains, especially

Belhassen, H., Zhang, H., Fresse, V. and Bourennane, E.
Improving Video Object Detection by Seq-Bbox Matching.
DOI: 10.5220/0007260002260233
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 226-233
ISBN: 978-989-758-354-4

to more challenging fast-moving objects, with quite light computational overhead in both settings. Applied to YOLOv3 (Redmon and Farhadi, 2018), our system achieves so far the best speed/accuracy trade-off for offline video object detection and competitive accuracy improvements for online object detection, e.g., 80.9% of mAP on our offline setting and 78.2% of mAP on online setting both at 38 fps on a Titan X GPU or at 51 fps on a GTX 1080Ti GPU.

Key contributions of this paper are:

- Our proposed box-level post-processing method achieves important accuracy improvements to per-frame detection baseline for both online and off-line settings, e.g, applied to YOLOv3, our method brings 6.9% of mAP gains (from 74% to 80.9%) for offline object detection and 4.2% of mAP gains (from 74% to 78.2%) for online detection on ImageNet VID validation set;
- Quite light computational overhead in both settings makes our method applicable in most practical vision applications, i.e., less than 2.5ms/frame additional computation for both settings;
- Our proposed method could be applied to detectors trained on video-based dataset or image-based dataset, which makes it more universal and practical for industrial applications.

This paper is organized as follows. In Section 2, we sum up some representative related works on image/video object detection. In Section 3, we explain the theoretical details of our proposed box-level post-processing method. In Section 4, we present our experimental results on ImageNet VID dataset and compare our results with other state-of-art methods. Section 5 concludes the paper.

2 RELATED WORK

In this section we present related works for both image and video detection and then position our contribution regarding the state-of-the-art works.

2.1 Object Detection from Image

Object detection from image is one of the most central study areas in computer vision. Detectors based on deep neural network greatly exceeds the accuracy of classic object detectors based on hand-designed features, e.g., Viola-Jones (Viola and Jones, 2004), DPM (Felzenszwalb and al., 2010), etc. Modern convolutional object detectors can be divided into two paradigms: Two-stage object detectors and Single-stage object detectors. According to (Huang and al., 2016), the former are generally more accurate while the latter are usually faster.

The dominant paradigm is based on a two-stage approach, which firstly generates a sparse set of regions of interest (RoIs) then classifies these regions and refines their bboxes. Faster R-CNN (Ren and al., 2015) introduces the Region Proposal Network (RPN) that generates RoIs and shares full-image convolutional features with Fast R-CNN (Girshick, 2015). R-FCN (Dai and al., 2016) proposes position-sensitive score maps to share almost all computation on the entire image.

Single-stage detectors perform object classification and bounding box regression simultaneously on feature maps. YOLO (Redmon and al., 2015) divides the image into regions and predicts bounding boxes and classification scores for each region. SSD (Liu and al., 2015) predicts classification scores and bbox adjustments of each default boxes of different aspect ratios and scales per feature map location. RetinaNet (Lin and al., 2017) proposes a novel loss function, Focal Loss, to address class imbalance problem.

While evaluating CNNs on Microsoft-COCO dataset, the recently proposed YOLOv3 (Redmon and Farhadi, 2018) achieves competitive accuracy (60.6% of AP^{50}) with state-of-the-art more complicated object detectors, e.g., R-FCN (51.9% of AP^{50}) or RetinaNet (57.5% of AP^{50}) and it detects at a much faster detection speed (YOLOv3 at 20 fps, R-FCN at 12 fps and RetinaNet at 5 fps on a Titan X GPU).

2.2 Object Detection in Video

Object detection in video has drawn more and more attention in recent years. The introduction of ImageNet (Russakovsky and al., 2014) object detection from video (VID) challenge made the evaluation of CNN designed for video easier. Instead of simply split the video into frames and perform per-frame object detection, recent works focus on utilizing contextual and temporal information to accelerate detection speed or to improve detection accuracy.

In terms of improving accuracy, recent works are designed on two levels: box level and feature level.

For box-level methods: T-CNN (Kang and al., 2016) uses pre-computed optical flow and object tracking to propagate high-confidence bounding boxes to nearby frames; Seq-NMS (Han and al., 2016) uses high-scoring object detections from nearby frames to boost scores of weaker detections within the same video clip.

Feature-level methods are generally considered to achieve a more important improvement than box-level methods. FGFA (Zhu and al., 2017) uses optical flow

and nearby frame feature maps to reinforce current feature map and improve object detection performance. Detect to Track and Track to Detect (Feichtenhofer and al., 2017) (D&T for short) introduces a ConvNet architecture that jointly performs detection and tracking. Scale-Time Lattice (Chen and al., 2018) proposes a novel architecture to reallocate the computation over a scale-time space.

2.3 Contribution Positioning Regarding the State-of-Art Works

We chose YOLOv3 as our base object detector thanks to its good speed / accuracy trade-off. Our proposed method is different from all previous works in video: Firstly, instead of utilizing object tracking (e.g., T-CNN and D&T) or heavy bbox across time sorting (e.g., Seq-NMS), we introduce *Seq-Bbox Matching*, which matches detected bboxes of the same appeared object to generate tubelets. Secondly, apart from tubelet rescoring (implemented in T-CNN, Seq-NMS, D&T and Scale-Time Lattice), we added another step, named *Tubelet-level Bbox Linking*, to infer missed detections and improve detection recall.

3 IMPROVING VIDEO OBJECT DETECTION BY SEQ-BBOX MATCHING

3.1 Overview

So far, object detection from image has achieved great success while object detection in video is still challenging. For one thing, motion blur, rare pose, camera defocus and other effects in videos will lead to wrong or missed detections. For another, temporal and contextual information can be helpful to improve detection accuracy.

In principle, our method can be applied to any kind of existing object detector (e.g., YOLOv3, SSD, RetinaNet, R-FCN, Faster-RCNN, etc). It consists of two main steps: Firstly, the *Frame-level Bbox Rescoring* to correct wrong detections and secondly the *Tubelet-level Bbox Linking* designed to infer bboxes of missed detections. These two steps are based on one basic operation: *Seq-Bbox Matching*.

Seq-Bbox Matching aims to match detected bboxes of the same appeared object to generate tubelets. To accelerate the detection speed, we implement object detection sparsely and apply *Bbox Bilinear Interpolation* to intermedia frames.

Section 3.2 explains how *Seq-Bbox Matching* works. Section 3.3 and 3.4 describes how we apply *Frame-level Bbox Rescoring* and *Tubelet-level Bbox Linking* respectively. In section 3.5, we modified our *Frame-level Bbox Rescoring* to adapt it to online video object detection. Finally, in section 3.6, we introduce the application of *Bbox Bilinear Interpolation* to accelerate the detection speed.

3.2 Seq-Bbox Matching

We match current-frame detected bboxes with previous-frame detected bboxes according to their geographical closeness and semantic similarity. Matched bboxes are considered as same objects detected in different frames. Matching distance of two bboxes is defined as below:

$$distance = \frac{1}{similarity} = \frac{1}{IoU \times (Vctr_i \cdot Vctr_j)} \quad (1)$$

Where the Intersection over Union (IoU) between two bboxes describes the geographical closeness, and the dot product of two classification score vectors ($Vctr_i$ for bbox i and $Vctr_j$ for bbox j) represents the semantic similarity of any two bboxes. Obviously, if IoU or the dot product equals to zero, the returned distance would be infinity. But this hardly ever happens in videos since the IoU is greater than 20% for two successive images in 99.91% of the cases, in ImageNet VID validation dataset.

Algorithm 1: Matching algorithm for detected bboxes in nearby frames.

```
Input: I bboxes detected in previous frame and
       J bboxes detected in current frame

distance_matrix = create_matrix(I,J)
for i = 1 to I do
  for j = 1 to J do
    set distance_matrix(i,j) to distance between
    i-th bbox of previous frame and j-th bbox of
    current frame
  end for
end for
set pairs to empty list
repeat
  set i,j to line, column of minimum value in
  distance_matrix
  add (i,j) to pairs
  set i-th line of distance_matrix to infinity
  set j-th column of distance_matrix to infinity
until minimum value of distance_matrix is infinity

Output: pairs
```

Algorithm 1 (pseudo code) summarizes our matching algorithm: For I bboxes detected in the previous

frame and J bboxes detected in current frame, we generate a distance matrix where each element of coordonate (i, j) represents the distance between $i-th$ bbox of previous frame and $j-th$ bbox of current frame; We simply match the bbox pair with minimal distance then cut out their connections with other bboxes by setting the line and the column of the selected element in the distance matrix to infinity. We repeat this step until there exists no more connections between any two bboxes, i.e., the minimal distance in the distance matrix becomes infinity.

3.3 Frame-level Bbox Rescoring

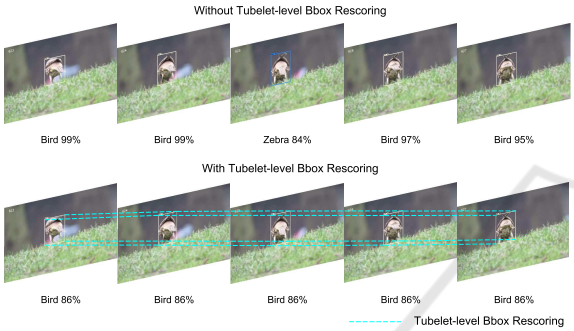

Figure 1: Frame-level Bbox Rescoring.

As shown in Figure 1, we adopt a simple rescoring method, named *Frame-level Bbox Rescoring*, to avoid object classification error caused by motion blur, rare pose or other reasons. As nearby frames are similar and usually contain a certain number of moving objects, we consider detection results in multiple nearby frames as multiple detection results of the same objects. We match detected bboxes in previous and current frame according to Algorithm 1 to generate tubelets and then simply rescore all the bboxes of the same tubelet by averaging their classification scores. Even if there exists several wrong detections (e.g., on the central image of Figure 1, the bird is wrongly detected as zebra due to motion blur), after averaging along the whole tubelet, these errors can be corrected.

3.4 Tubelet-level Bbox Linking

As shown in Figure 2, we add a simple tubelet linking method, named *Tubelet-level Bbox Linking*, to infer missed detections and improve recall. Similar as Algorithm 1, but instead of matching previous-frame bboxes with current-frame bboxes, we try to match the last bbox of one tubelet and the first bbox of another. Note that we mark isolated bboxes, i.e., bboxes that are never matched after going through the whole video, as first and last bboxes of tubelets in the same

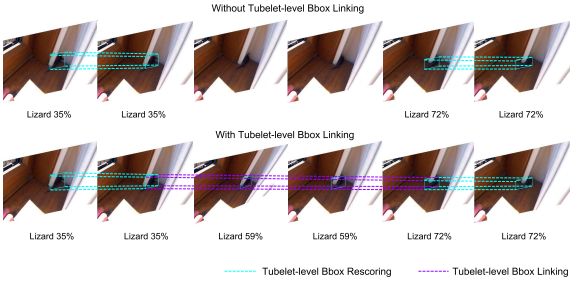

Figure 2: Tubelet-level Bbox Linking. Zoom for details.

time, because they are considered as tubelets of length one.

We set a thresh κ to make sure that two tubelets are not temporally too far. Tubelets with a temporal interval longer than κ frames are ignored to be matched.

If the last bbox of one tubelet and the first bbox of another match, their two tubelets are considered as detection of a same object and there might be missed detections between these two tubelets. As shown in equation (4), we infer the locations of the intermedia missed bboxes between matched tubelets by bilinear interpolation of matched bboxes and the classification scores are calculated as the averaged scores of two matched tubelets w.r.t the lengths of tubelets.

$$Vctr^{add} = \frac{Cnt^f}{Cnt^l + Cnt^l} Vctr^f + \frac{Cnt^l}{Cnt^f + Cnt^l} Vctr^l \quad (2)$$

We use $*^f$ for former tubelet and $*^l$ for later tubelet.

Along these lines, *Tubelet-level Bbox Linking* helps to infer missed detections and improve detection recall, e.g., the missed detections of lizard (because of rare pose) in the central two images of Figure 2 are refound by our *Tubelet-level Bbox Linking*. Suppressing bboxes by utilizing contextual information or tubelet information could eventually further improve the detection accuracy, but for simplicity, we leave it for future work.

3.5 Online Frame-level Bbox Rescoring

It is meaningful to propose an online object detection method as it is a common scenario in many real-time computer vision applications, e.g., self-driving cars. Online video object detection asks to perform object detection in an image flow without access to future frames or overall contextual information, which is not the case of the previously presented two steps. So we modified our *Frame-level Bbox Rescoring* to adapt it to online video treatments.

Instead of rescoring bboxes of a tubelet after going

through all the video, each time one bbox is matched, we update its classification scores by averaging all historical classification scores of the same object. This is realized by applying a dynamic averaging method: we add a count variable for each bbox, and each time a bbox is matched, its variable count increases. The averaged historical classification score vector of current bbox becomes:

$$Vctr^{cur} = \frac{Vctr^{cur}}{Cnt^{pre}+1} + \frac{Cnt^{pre}}{Cnt^{pre}+1} Vctr^{pre} \quad (3)$$

and then:

$$Cnt^{cur} = Cnt^{pre} + 1 \quad (4)$$

We use Cnt for Count, $*^{cur}$ for current and $*^{pre}$ for previous.

In this way, our *Frame-level Bbox Rescoring* could be performed through an online manner and the last bbox of one tubelet holds the averaged classification scores of the whole tubelet.

3.6 Bbox Bilinear Interpolation

If we divide the trajectory of a moving object into many short pieces, these short pieces could be seen as short-range nearly uniform movements. In this case, we only need to perform detection in sparse keyframes. Object positions of intermedia frames could be estimated by Bilinear Interpolation of positions of the two nearby keyframes. We adopt bilinear interpolation of matched bboxes for intermedia positions reconstruction to accelerate the detection. We introduce a hyperparameter β to control the length of intermedia frames for Bilinear Interpolation. β is proportional to the detection speed (in fps), e.g., β=1 represents that our detection speed is two times faster. However, if β becomes too large, the movements between sparse keyframes are too complicated to be seen as nearly uniform movements which will lead to detection accuracy drop.

Adaptive *Bbox Bilinear Interpolation* interval could obviously further improve the speed/accuracy trade-off, which is left for future work.

4 EXPERIMENTS

4.1 Experimental Setting

4.1.1 Dataset

All experiments are conducted on the ImageNet (Russakovsky and al., 2014) object detection from video (VID) dataset. It is a large scale video-based dataset, that consists of 3862 video clips for training, 555 for validation and 937 for testing. Each clip is fully annotated with bounding boxes of 30 different object classes. Challenges of this dataset include camera defocus, partial occlusion, motion blur, crowded instance, background confusion, rare pose, etc. Since the annotations of the test set are not publicly available, we evaluate our method on the evaluation set using the mean Average Precision (mAP) metric. For motion-specific evaluation, we use the code provided by (Zhu and al., 2017).

4.1.2 Implementation Details

Similar as (Kang and al., 2016), (Zhu and al., 2017), (Feichtenhofer and al., 2017) and (Chen and al., 2018), we combine the Imagenet DET train set and VID train set to train our base object detection model. As DET dataset has more object classes than VID dataset (200 for DET *vs* 30 for VID), we suppress images without target classes and only preserve annotations for the 30 target classes in the DET dataset. The VID set is presented in the form of videos. We do not use these videos directly because of the redundancy between nearby frames. We sample the VID train set by a certain percentage to maintain a ratio of 1:1 between DET train set and VID train set in the final train set (55k images for each set).

Among all state-of-the-art convolutional object detectors, we choose the recently proposed YOLOv3 (Redmon and Farhadi, 2018) as our per-frame object detector thanks to its good speed/accuracy trade-off. Following (Redmon and Farhadi, 2018) and (Redmon and Farhadi, 2016), we implement k-means to determine the anchor scales and aspect ratios of each detection layer. Our model is pre-trained on imagenet classification task, then we fintune it on our training set with the learning rate 0.001 for the first 80k iterations and 0.0001 for the last 30k iterations. All training are performed on two GTX 1080Ti GPUs with batch size 64. After 110k iterations, the training loss converged. Note that as YOLOv3 has a fully convolutional architecture and uses multi-scale training, our trained model can be evaluated at different resolutions. It achieves 74% of mAP at resolution 608×608 and 71.2% at 416×416 on VID validation set.

For inference phase, after per-frame dense detection, we first apply non-maximum suppression (NMS) with a threshold 0.55 to select bboxes to be matched on each frame. Then we apply respectively the two settings of our box-level post-processing method: For offline seting, we perform successively *Frame-level Bbox Rescoring* and *Tubelet-level Bbox Linking* as presented previously (Section 3.3 and 3.4); For online setting, we simply apply *Online Frame-*

level Bbox Rescoring (Section 3.5). We use different input image resolutions, different temporal intervals β for *Bbox Bilinear Interpolation* and different thresh κ for *Tubelet-level Bbox Linking* to explore their impact on detection speed and accuracy.

4.2 Results

4.2.1 Speed/Accuracy Curve

We summarize the speed/accuracy curve of our method applied to our trained model at two resolutions (608×608 and 416×416) on two settings (offline and online) in Figure 3. Runtimes are evaluated by frame per second (fps) on a Titan X GPU and accuracy are evaluated by the conventional metric mean Average Precision (mAP) on the ImageNet VID evaluation set.

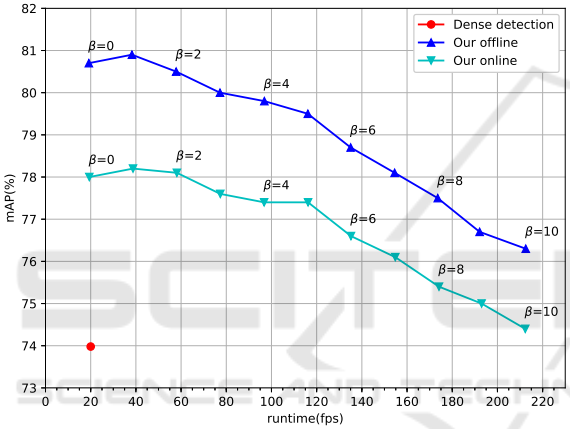

Figure 3: Speed/accuracy curves of our proposed method.

As shown in Figure 3, top left and right plots are for the speed/accuracy trade-off of our offline and online settings; bottom left and right are for the mAP improvements of our two settings. The speed/accuracy trade-off was made under different *Bbox Bilinear Interpolation* intervals β and different input image resolutions (416×416 and 608×608).

From Figure 3 we observe that: Firstly, both settings of our method highly improve the accuracy of our per-frame detection baseline; Secondly, long *Bbox Bilinear Interpolation* intervals will lead to tragic accuracy drops, especially when $\beta > 5$; Thirdly, for the choice of input image resolution, on both settings, 608×608 achieves better accuracy when $fps < 175$; Fourthly, our method achieves generally higher accuracy improvements for our model with the input image resolution 416×416.

4.2.2 Ablation Study

For simplicity, all experiments were conducted on our trained model with input image resolution 608×608 with the bilinear interpolation interval $\beta = 1$.

Step Validation. Table 1 compares our proposed method with the per-frame dense detection baseline and its variants.

Method (a) is the per-frame dense detection baseline of our trained model. We simply divide each video in ImageNet VID validation set into frames and perform per-frame object detection. The only post-processing is NMS with thresh 0.55. Without bells and whistles, our base model achieves 74.0% of mAP for per-frame dense detection with 50ms of runtime on a Titan X GPU. We find our trained model much faster in terms of speed and competitive in terms of accuracy with other trained baseline models in recent works, e.g., FGFA(Zhu and al., 2017), Scale-Time Lattice(Chen and al., 2018) and D&T (Feichtenhofer and al., 2017).

Following FGFA(Zhu and al., 2017), we continued to evaluate our model on three motion groups: slow, medium and fast. It turns out that our model is good at detecting slow-moving objects with mAP of 81.5% but struggling at detecting fast-moving objects, where the mAP drops directly to 50.5%.

Method (b) is the implementation of Seq-NMS (Han and al., 2016). Even though it boosts the mAP from 74.0% to 77.8%, it takes too much time (71.5ms/frame) for sorting sequences of bboxes.

Method (c) implements our *Frame-level Bbox Rescoring* in an offline way (Section 3.3). This method improves the detection accuracy by 5.4% for slow-moving object, 7.6% for medium and 11.8% for fast, which shows that our bbox rescoring methods can largely boost the detection accuracy for more challenging fast-moving objects. Compared with (b), same as offline video object detection, our method requires less post-processing time (1.7ms/frame for ours *vs* 71.5ms/frame for Seq-NMS) while achieves a higher mAP gains (6.6% for ours *vs* 3.8% for Seq-NMS), which proves that ours outperforms Seq-NMS.

Note that up until *Method (c)*, we do not suppress any detected bboxes or add any extra bboxes. All mAP gains come from rescoring methods that correct potential wrong detections. *Method (d)* is the offline setting of our method, which adds the *Tubelet-level Bbox Linking* (Section 3.4) on the basis of (c). The objective of *Tubelet-level Bbox Linking* is to infer missed detections between nearby tubeletes, and this step boosts our final mAP to 80.9%. We find the additional computation overhead for matching tubelets acceptable (0.6ms/frame) and this step more helpful for more

Table 1: Accuracy and runtime of different methods on ImageNet VID validation. All experiments are conducted at the input image resolution 608×608, (c), (d) and (e) are results with the bilinear interpolation interval $\beta = 1$. The relative gains compared to the single-frame baseline (a) are listed in the subscript.

method	(a)	(b)	(c)	(d)	(e)
mAP(%)	74.0	77.8 $\uparrow$ 3.8	80.6 $\uparrow$ 6.6	**80.9** $\uparrow$ **6.9**	78.2 $\uparrow$ 4.2
mAP(%) (slow)	81.5	83.8 $\uparrow$ 2.3	86.9 $\uparrow$ 5.4	**87.1** $\uparrow$ **5.6**	85.3 $\uparrow$ 3.8
mAP(%) (medium)	71.7	76.6 $\uparrow$ 4.9	79.3 $\uparrow$ 7.6	**79.6** $\uparrow$ **7.9**	76.3 $\uparrow$ 4.6
mAP(%) (fast)	50.5	58.0 $\uparrow$ 7.5	62.3 $\uparrow$ 11.8	**62.7** $\uparrow$ **12.2**	57.2 $\uparrow$ 6.7
runtime(ms) (all)	50	121.5	51.7	52.3	51.7
runtime(ms) (post-processing)	0.01	71.5	1.7	2.3	1.7

Table 2: Performance comparison on the ImageNet VID validation set. All experiments are conducted at the input image resolution 608×608 with the bilinear interpolation interval $\beta = 1$. The average precision (in %) for each class is shown.

Methods	airplane	antelope	bear	bicycle	bird	bus	car	cattle	dog	d.cat
Baseline	**93.3**	83.3	71.3	82.3	70.9	67.2	65.6	72.6	56.3	76.1
offline	93.2	**86.4**	**81.5**	**85.2**	**75.3**	**71.2**	**66.7**	**82.4**	**70.1**	**92.3**
Online	93.1	85.6	78.3	83.0	74.0	70.0	66.1	77.5	66.0	85.1
Methods	elephant	fox	g.panda	hamster	horse	lion	lizard	monkey	moto	rabbit
Baseline	74.8	89.3	86.4	89.2	76.1	53.0	73.2	48.2	86.8	65.7
offline	**79.3**	**97.0**	**88.0**	**95.8**	**82.7**	**72.8**	**83.6**	**53.7**	**91.7**	**80.0**
Online	78.3	93.2	86.6	92.4	79.9	65.9	76.4	51.4	89.3	74.8
Methods	r.panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra
Baseline	80.9	69.6	64.6	54.7	88.4	85.0	75.6	64.0	65.8	90.6
offline	**96.0**	**78.8**	**70.7**	**64.5**	**90.4**	**86.5**	**78.7**	**67.5**	**68.4**	**95.3**
Online	94.9	75.7	68.4	62.9	89.3	85.7	76.5	66.8	64.2	93.8

challenging fast-moving object detection.

Method (e) is the online setting of our method. It is realized by our *Online Frame-level Bbox Rescoring* (Section 3.5). We use dynamique averaging method to rescore the bbox of one object according to its historical classification scores. Compared with (b), which is implemented in a offline manner, our online method achieves a higher improvement in terms of accuracy (4.2% for ours *vs* 3.8% for Seq-NMS). Compared with(c), (e) is relatively less effective due to the lack of future information. Note that the computational overhead of (e) is quite light (1.7ms/frame) and affordable for most real-world applications.

As to runtime of (c), (d) and (e), only half of the frames in the validation set are actually detected and others are estimated by *Bbox Bilinear Interpolation*. All mentioned runtimes are for one detected frame instead of average runtimes of a whole video.

Per-class Analyse on the ImageNet VID Validation Set. Due to movement blur or other reasons, our per-frame baseline has a very poor performance for some classes (e.g., lion with 53% of AP) as a result of wrong or missed detections. Both our online and offline settings highly improve the detection accuracy. Some class-AP scores are improved significantly, e.g., for the class lion, 19.8% of gain on offline setting and 12.8% on online setting; for the class red panda, 15.1% of gain on offline setting and 14.0% on online setting. These AP gains result from the rise of classification accuracy and object detection recall, which are realized by our *Frame-level Bbox Rescoring* and *Tubelet-level Bbox Linking* respectively. For the airplane class presenting heavy occlusions and fast movements, the matching of bboxes became more challenging which leads to slight accuracy drop.

Thresh κ Analyze. In Figure 4, we show the accuracy trends with thresh κ varying from 4 to 24.

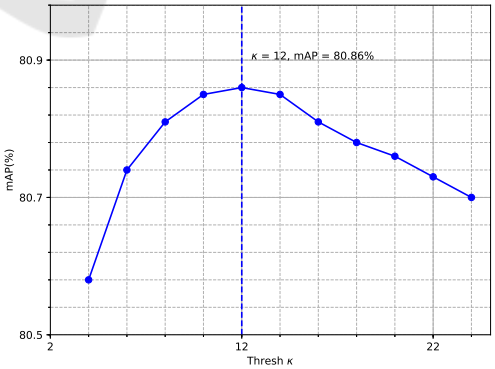

Figure 4: Accuracy *vs* thresh κ on ImageNet VID dataset.

The plot shows that the accuracy rises quickly from 80.6% to 80.9% as thresh κ rises from 4 to 12, then the accuracy drops slowly. This result can be interpreted as: with the raise of κ, not only more missed detections are infered but also more false-positive

bboxes are ”made”. After exceeding a critical value, i.e., 12 in our case, the latter becomes dominant, leading to accuracy drop.

4.3 Comparison with State-of-the-Art Methods

According to FGFA (Zhu and al., 2017), Seq-NMS (Han and al., 2016) obtains larger gain than T-CNN (Kang and al., 2016), so we implement Seq-NMS on our trained model with input image resolution 608×608 for comparison. Our method achieves a higher accuracy in both online and offline settings and both require at least 30 times less computation time (71.5ms for Seq-NMS *vs* 1.7ms and 2.3ms for our method), which proves that our box-level post-processing method is more effective. Moreover, Seq-NMS can only be applied to offline video object detection, while we also propose an online setting. We further compare our results with several state of the art video object detection methods. For fair comparison, we reported our detection speed on Titan X.

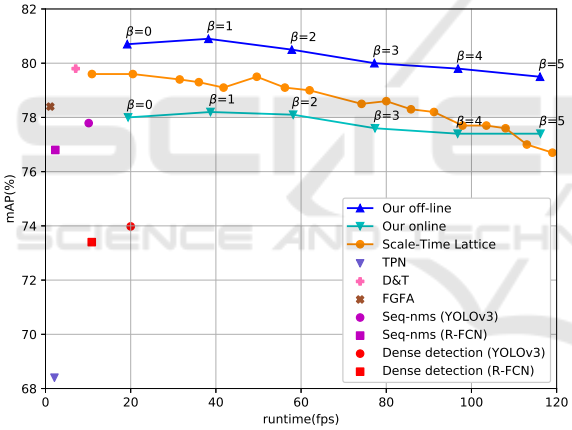

Figure 5: Speed/accuracy comparison of our proposed method applied to YOLOv3 with other state-of-the-art methods on the ImageNet VID validation set.

As shown in Figure 5, our offline setting hits the best speed/accuracy trade-off compared with other box-level or feature-level methods. In particular, it achieves 80.9% of mAP at 38 fps and 79.5% of mAP at 116 fps on a Titan X GPU. Our online setting gets competitive speed/accuracy trade-off and is more adaptive for more practical online object detection applications.

5 CONCLUSION

In this paper, we present a novel, simple and highly effective box-level post-processing method, named *Seq-Bbox Matching*, to improve video object detection. Experiments show that, applied to YOLOv3, our method is more effective than all existing box-level methods and hits the best speed/accuracy trade-off compared with other state-of-the-art methods. The online setting of our method performs video object detection without access to future frames and achieves competitive accuracy improvement. Despite the large detection accuracy improvement, the most important advantage of our method is its quite light computational overhead which makes it applicable in most real-world computer vision applications.

REFERENCES

Chen, K. and al. (2018). Optimizing video object detection via a scale-time lattice. *CoRR*, abs/1804.05472.

Dai, J. and al. (2016). R-FCN: object detection via region based fully convolutional networks. *CoRR*, abs/1605.06409.

Feichtenhofer, C. and al. (2017). Detect to track and track to detect. *CoRR*, abs/1710.03958.

Felzenszwalb, P. F. and al. (2010). Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645.

Girshick, R. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Han, W. and al. (2016). Seq-nms for video object detection. *CoRR*, abs/1602.08465.

Huang, J. and al. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012.

Kang, K. and al. (2016). T-CNN: tubelets with convolutional neural networks for object detection from videos. *CoRR*, abs/1604.02532.

Lin, T. and al. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Liu, W. and al. (2015). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Redmon, J. and al. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Ren, S. and al. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Russakovsky, O. and al. (2014). Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154.

Zhu, X. and al. (2017). Flow-guided feature aggregation for video object detection. *CoRR*, abs/1703.10025.