ADASCALE: TOWARDS REAL-TIME VIDEO OBJECT DETECTION USING ADAPTIVE SCALING

Ting-Wu Chin [*1] **Ruizhou Ding** [*1] **Diana Marculescu** [1]

ABSTRACT

In vision-enabled autonomous systems such as robots and autonomous cars, video object detection plays a crucial role, and both its speed and accuracy are important factors to provide reliable operation. The key insight we show in this paper is that speed and accuracy are not necessarily a trade-off when it comes to image scaling. Our results show that re-scaling the image to a lower resolution will sometimes produce better accuracy. Based on this observation, we propose a novel approach, dubbed AdaScale, which adaptively selects the input image scale that improves both accuracy and speed for video object detection. To this end, our results on ImageNet VID and mini YouTube-BoundingBoxes datasets demonstrate *1.3 points* and *2.7 points* mAP improvement with *1.6× and 1.8× speedup*, respectively. Additionally, we improve state-of-the-art video acceleration work by an extra *1.25× speedup* with slightly better mAP on ImageNet VID dataset.

1 INTRODUCTION

Video object detection acts as a fundamental building block for visual cognition in future autonomous agents such as autonomous cars, drones, and robots. Therefore, to build systems with reliable performance, it is critical for the detectors to be fast and accurate. Though object detection is well-studied for static images (Dai et al., 2016; Girshick, 2015; He et al., 2014; Liu et al., 2016; Ren et al., 2015), there are unique challenges in the case of *video* object detection, including motion blur caused by the moving objects, failure of camera focus (Zhu et al., 2017a), and also real-time speed constraints when it comes to autonomous agents. Besides these *challenges*, however, video object detection also brings new *opportunities* to be exploited. Some of the prior work that focuses on video object detection tries to improve average precision by leveraging a unique characteristic of video (Zhu et al., 2017a; Feichtenhofer et al., 2017; Kang et al., 2017), which is the temporal consistency (consecutive frames have similar content). On the other hand, from a speed perspective, prior work (Zhu et al., 2017b; 2018b; Buckler et al., 2018) counts on the temporal consistency to reduce the computation needed for a standalone object detector. Similarly, we aim to leverage the temporal consistency, but to improve both speed and accuracy of the standalone object detectors with a novel technique called adaptive-scale testing, or *AdaScale*.

The scale of input image affects both the speed and accuracy of modern CNN-based object detectors (Huang et al., 2017). Prior work related to image scaling addresses two directions: (i) multi-scale testing for better accuracy, and (ii) down-sampling images for higher speed. Examples from the first category include re-sizing images to various scales (image pyramid) and pushing them through the CNN for feature extraction at various scales (Dai et al., 2016; Girshick, 2015; He et al., 2014), as well as fusing feature maps from different layers generated by a single-scale input image (Lin et al., 2017a; Cai et al., 2016; Bell et al., 2016). However, these approaches introduce extra computational overhead compared to object detectors with single-scale inputs. Examples from the second category include Pareto optimal search by tuning the input image scale (Lin et al., 2017b; Liu et al., 2016; Redmon & Farhadi, 2017; Huang et al., 2017) and dynamically re-sizing the image according to the input image (Chin et al., 2018). However, results for these approaches demonstrate that higher speed comes at the cost of lower accuracy when it comes to image scaling.

In contrast with prior work, we find that down-sampling images is sometimes beneficial in terms of accuracy. Specifically, there are two sources of improvement brought by image down-sampling: (i) Reducing the number of false positives that may be introduced by focusing on unnecessary details. (ii) Increasing the number of true positives by scaling the objects that are too large to a size at which the object detector is more confident. Fig. 1 shows images that are better when down-sampled in our experiments using Region-based Fully Convolutional Network (R-FCN) (Dai et al., 2016) object detector on ImageNet VID dataset.

[*]Equal contribution [1]Department of ECE, Carnegie Mellon University, Pittsburgh. Correspondence to: Ting-Wu Chin <tingwuc@cmu.edu>, Ruizhou Ding <rding@cmu.edu>.

Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA, 2019.

Motivated by this, our goal is to re-size the images to their "best" scale aiming for both higher speed and accuracy. In this work, we propose *AdaScale* to boost both the accuracy and the speed of the standalone object detector. Specifically, we use the current frame to predict the optimal scale for the next frame. Our results on ImageNet VID and mini YouTube-BB datasets demonstrate 1.3 points and 2.7 points mAP improvement with 1.6× and 1.8× speedup, respectively. Moreover, by combining with the state-of-the-art video acceleration work (Zhu et al., 2017b), we improve its speed by an an extra 25% with a slight mAP increase on ImageNet VID dataset.

2 Related Work

Our work focuses on applying adaptive scaling to video object detection in order to improve both speed and accuracy of object detectors. In the sequel we discuss prior work in scale-related object detection and video object detection.

2.1 Image Scale for Object Detection

We discuss two categories of scale-related object detection work: (i) single-shot detection by exploiting feature maps from various layers of the CNN with inherently different scales, and (ii) multi-shot detection with input images at multiple scales.

Single-Shot: In this category, object detectors are designed to take an input image once and detect objects at various scales. That is, this category of prior work treats deep CNNs as scale-invariant. Prior work (Bell et al., 2016) uses features from different layers in the CNN and merge them with normalization and scaling. Similar idea is also adopted by other work (Liu et al., 2016; Cai et al., 2016; Lin et al., 2017a; Zhou et al., 2017). From a different viewpoint, prior art (Liu et al., 2017) proposes to use a recurrent network to approximate feature maps produced by images at different scales. Though single-shot approaches have shown great promise in better detecting various scales, the scale-invariant design philosophy generally requires a large model capacity (Kanazawa et al., 2014; Liu et al., 2017). We note that, without perfect scale-invariance, different image scales will result in different accuracy, and prior art often uses a fixed single scale, *e.g.* 600 pixels on the smallest side of the image. Hence, this line of work could be further improved in terms of speed and accuracy when augmented to adaptive scaling.

Multi-shot: This refers to scaling a single input image to various scales, forwarding each scaled image through the object detector, and merging the obtained results. Some work (He et al., 2014; Girshick, 2015; Ren et al., 2017; He et al., 2016; Dai et al., 2016) forwards multiple scales of images to obtain feature maps with various scales. More recently, prior work (Dai et al., 2018) leverages multiple scales of images to infer multiple detection results and merge them using Non-Maximum Suppression. While multi-shot object detection alleviates the problem of imperfect scale-invariance, it incurs significant extra computation overhead, *i.e.*, up to 4× (Girshick, 2015).

Our work is aiming to alleviate the imperfect scale-invariance by selecting the best scale for each image, and hence, improves the accuracy compared to single-shot methods. Moreover, to improve the speed in the meantime, we consider down-sampling rather than up sampling. We note that our method could possibly be extended to multi-shot version, *i.e.*, adaptively select multiple scales for a given image, and we leave it for the future work.

2.2 Video Object Detection

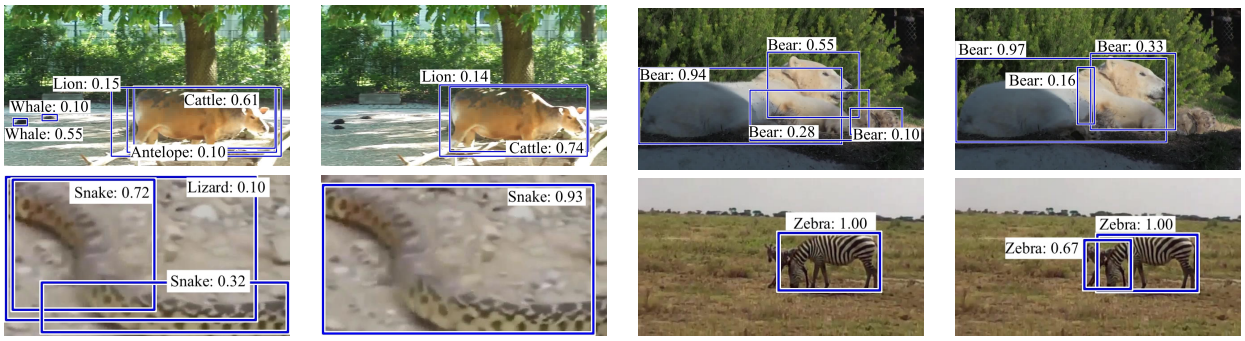
We discuss prior work that aims at improving speed and/or accuracy of video object detection.

Speed: Optical flow was proposed to reduce detection overhead by prior work (Zhu et al., 2017b). Similar to our idea, some prior art (Chin et al., 2018) proposes to adaptively scale the image to improve the detection speed. However, both works improve speed at the expense of accuracy loss. *Accuracy:* Prior work (Kang et al., 2017) proposes to leverage contextual and temporal information across the video while some work (Zhu et al., 2017a) uses the idea from Deep Feature Flow (DFF) (Zhu et al., 2017b) to incorporate temporal information across consecutive frames. Another study (Feichtenhofer et al., 2017) proposes to integrate detection with tracking into an end-to-end trainable deep CNN. *Both*: Some prior work (Zhu et al., 2018a) extends (Zhu et al., 2017a) and (Zhu et al., 2017b) to use both feature aggregation and propagation. Additionally, they propose to regress a quality metric of the optical flow to decide when and how to propagate the features.

Compared to the aforementioned related work, the main contribution that sets our work apart is that our work focuses on fixing the problem where existing object detectors use fixed single scale for each of the image while the object detectors are not scale-invariant, which is different from most of the prior work that focuses on exploiting the relationship of the detection results among the neighboring frames (Kang et al., 2017; Zhu et al., 2017b;a; Feichtenhofer et al., 2017; Zhu et al., 2018a). Moreover, we show that we are complementary to state-of-the-art video object detection acceleration technique (Zhu et al., 2017b).

3 Adaptive Scaling

Fig. 2 provides an overview for AdaScale methodology. It includes fine-tuning the object detector, using the resulting detector to generate the optimal scale labels, training the



(a) 600 (b) 240 (c) 600 (d) 480

Figure 1. Examples where down-sampled images have better detection results. Blue boxes are the detection results, and the numbers are the confidence. The detector is trained on a single scale (pixels of the shortest side) of 600. Column (a) and (c) are tested at scale 600. Column (b) is tested at scale 240 and column (d) is tested at scale 480.

scale regressor with the generated labels, and the deployment of AdaScale in video object detection. We discuss each component in detail in the following sections.

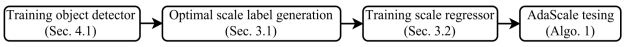


Figure 2. The AdaScale methodology.

3.1 Optimal Scale

To define the optimal scale (pixels of the shortest side) of a given image, we need to first define a finite set of scales S (*e.g.*, in our case $S = \{600, 480, 360, 240\}$) and we must have a metric that evaluates the quality of the detection results at these different scales. Naively, we can use the commonly used mean average precision (mAP) to compare different scales, and define the scale with the largest mAP as the optimal scale. However, the mAP evaluated for a single image is sparse due to limited number of ground truths per image. Hence, we opt to count on the loss function that is used to train the object detector as the metric to compare results at different scales. In general, the loss function for an object detector used in training often includes the bounding box regression loss and classification loss (Girshick, 2015; Ren et al., 2015; Dai et al., 2016):

$$L(\mathbf{p}, u, \mathbf{t}, \hat{\mathbf{t}}) = L_{cls}(\mathbf{p}, u) + \lambda[u \geq 1]L_{reg}(\mathbf{t}, \hat{\mathbf{t}}), \quad (1)$$

where $\mathbf{p}$ is a vector of predicted probability for each predefined class, u is the ground truth class label (0 means background), $\hat{\mathbf{t}}$ is a four-dimension vector that indicates the location information of the bounding box (Girshick, 2015), and $\mathbf{t}$ is also a four-dimension vector that represents the ground truth location of the bounding box. Noted that $[u \geq 1]$ indicates that regression loss only applies to the bounding box whose ground truth label is not background. Generally (Dai et al., 2016; Lin et al., 2017b), a predicted bounding box is assigned to foreground when there is at least one ground truth bounding box that has over 0.5 Jaccard overlap (intersection over union) (Erhan et al., 2014) with it; otherwise, it is assigned to background. However, since this loss function naturally assumes that the regression loss for background is 0, directly using it to *assess* different image scales will favor the image scale with fewer foreground bounding boxes.

Hence, to deal with this, we devise a new metric that focuses only on the same number of foreground bounding boxes to compare different image scales. To explain our proposed metric, we denote $L^m_{i,a}, m \in S$ as the loss of predicted bounding box a of image i at scale m using (1), and denote $\hat{L^m_i}, m \in S$ for image i at scale m, as our proposed metric. To obtain $\hat{L^m_i}$, we first compute the number of predicted foreground bounding boxes, $n_{m,i}$, for image i at each scale $m \in S$, then let $n_{min,i} = \min_m(n_{m,i})$. Concretely, the proposed metric can be computed as: $\hat{L^m_i} = \sum_{a \in A_{m,i}} L^m_{i,a}$, where $A_{m,i}$ is a set of predicted foreground bounding boxes of image i at scale m and $|A_{m,i}| = n_{min,i}$. To obtain $A_{m,i}$, for each scale, we sort the predicted foreground bounding boxes of image i with respect to $L^m_{i,a}$ in ascending order and pick the first $n_{min,i}$ predictions into the set $A_{m,i}$. The visual illustration of the process is shown in Fig. 3. With the proposed metric, we define optimal scale $m_{opt,i}$ for image i as:

$$m_{opt,i} = argmin_m \hat{L^m_i}. \quad (2)$$

3.2 Scale Regressor

Now that we understand which scale is better for a given image, we may be able to predict the optimal scale for the image. Intuitively, if the object is large or has simple texture, it is likely that we would down-sample the image to let the object detector focus on the salient objects rather than the distracting details. On the other hand, if the object is small or there are many salient objects, the image should remain in a large scale. Since R-FCN head (Dai et al., 2016)

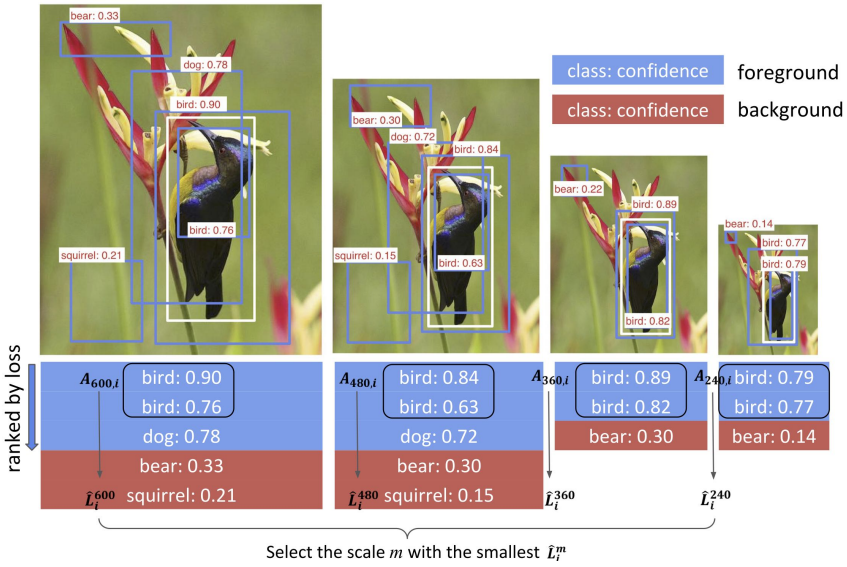


Figure 3. Optimal scale determination. First, the same number of predicted foregrounds from four scales are selected as $A_{m,i}$. Then, the scale with the lowest loss $\hat{L}_i^m$ is selected as the optimal scale.

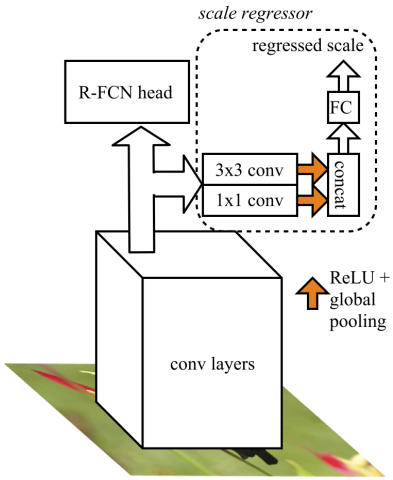


Figure 4. The scale regressor module.

counts on the deep features (*i.e.*, the last convolutional layer of the backbone feature extractor) to regress bounding box locations, we think that the channels of the deep features already contain size information. As a result, we build a scale regressor using deep features to predict the optimal scale, as shown in Fig. 4. Specifically, we use a 1x1 convolutional layer to capture the size information from different feature maps. Additionally, we use a parallel 3x3 convolutional layer to capture the complexity of each 3x3 patch in the feature maps. After the non-linear unit, we use global pooling that acts as a voting process. Lastly, we combine the two streams with a fully connected layer to regress the output scale. To be precise, we define the deep features as $X \in R^{C\times H\times W}$, where C is number of channels, H and W are height and width of the deep feature maps. We define our regressor as $g : R^{C\times H\times W} \rightarrow R$. It is important to note that we do not regress the optimal scale m_{opt} directly since what matters is the content instead of the image size itself. Hence, we regress a relative scale so that the module learns to react (up-sample, down-sample, or stay the same) given the current content of the image. Specifically, the target of the regressed scale for image i is defined as:

$$t(m_i, m_{opt,i}) = 2 \times \frac{m_{opt,i}/m_i - m_{min}/m_{max}}{m_{max}/m_{min} - m_{min}/m_{max}} - 1, \tag{3}$$

where m_i is the scale of the image i, m_{min} is the minimum defined scale, *e.g.*, 128, while m_{max} is the maximum defined scale, *e.g.*, 600. That is, we are regressing to normalized, *i.e.*, [-1, 1], relative scales. To generate labels for the regressor, we calculate (2) over the training data to obtain $m_{opt,i}\ \forall i \in D_{train}$, where D_{train} is the training data. As commonly used in regression problems, we adopt mean square error (4) as the loss function to train the regressor:

$$L_{scalereg} = \frac{1}{|D_{train}|} \sum_{i\in D_{train}} (g(X_i) - t(m_i, m_{opt,i}))^2. \tag{4}$$

To incorporate adaptive scaling, or AdaScale, in the video setting, we impose a temporal consistency assumption. More precisely, we assume that the optimal scales for the two consecutive frames are similar; our results empirically justify this assumption. Algorithm 1 shows an example of leveraging AdaScale for video object detection, which is elaborated in section 4.2.

4 Experiments

4.1 Setup

All of our experiments are done using Nvidia GTX 1080 Ti. We base our implementation on the code released by prior work (Zhu et al., 2017b), where MXNet (Chen et al., 2015) is used as the deep learning framework. We conduct our experiments mainly on the ImageNet VID dataset (Russakovsky et al., 2015), which contains 3862 and 555 training and validation video snippets, respectively. We use a pre-trained R-FCN model (Zhu et al., 2017b), which is trained on both ImageNet DET and ImageNet VID training set. For DET dataset, only the 30 categories that overlap with the VID dataset are selected for training. The evaluation of ImageNet VID is performed on validation set, which follows

Algorithm 1 Pseudo-code for using AdaScale in the testing phase.

```
Input: detector, video, S: pre-defined scale set
image = video.next_frame();
targetScale = 600; // Initialize image scale
while image do
  image = resize(image, targetScale);
  base_size = minimum(image.height, image.width);
  // Regress t of Eq. (3)
  bboxes, scores, targetScale = detector.detect(image);
  // Invert Eq. (3)
  targetScale = decode(targetScale, base_size, S);
  targetScale.clip_(min(S), max(S)).round_();
  image = video.next_frame();
end while
```

prior work (Zhu et al., 2017b). In addition to ImageNet VID, we also evaluate our performance on the recently released YouTube-BB dataset (Real et al., 2017), which contains 23 categories and around 380,000 video segments. Due to resource and time limitation, we randomly sample 100 segments per category and cut 20 frames per segment to form our mini training set. We also sample 10 segments per category for the validation set to form our mini testing set. To train the model for mini Youtube-BB dataset, we use the model trained on ImageNet VID and DET as a pre-trained model to further fine-tune on mini Youtube-BB.

4.2 Training and Testing

Object Detector: First, to avoid the object detector to be biased toward a single scale, we fine-tune the R-FCN model pre-trained at scale 600, for four epochs using multi-scale training (Girshick, 2015). The hyperparameters used follow prior work (Zhu et al., 2017b). Specifically, we use a learning rate of 0.00025 and divide it by 10 after 1.3 and 2.6 epochs, respectively. We use two GPUs with a single image per GPU. Therefore, the training batch size is two. In a addition, we pick the scale (the shortest side of the image) from the set $S_{train} = \{600, 480, 360, 240\}$, and use the maximum bound for the longer side as 2000. Our resizing protocol follows Fast R-CNN (Girshick, 2015). In the following sections, we will refer to the shortest side size as the *image scale*. All the detection results in this work use Non-Maximum Suppression (NMS) with threshold 0.3 (Dai et al., 2016). For each image, the top-300 confident bounding boxes after NMS are selected as the final output.

Scale Regressor: With the multi-scale trained object detector, we generate the scale label for each frame in the training data with a set of pre-defined scales using the proposed metric in section 3.1. To enable adaptive scaling, the regressor needs to learn to scale up or down according to the current content. To best train the regressor, we should scale the image to every possible scales for the regressor to learn the dynamics. That is, when training the regressor, the input image scale is randomly drawn from a uniform distribution of the pre-defined scale set S_{reg}. In practice, we find $S_{reg} = \{600, 480, 360, 240, 128\}$ is enough to cover the dynamics between 600 and 128. Note that we pick 128 since it is the scale of smallest pre-defined bounding box or anchor used in the Region Proposal Network (Ren et al., 2015) inside R-FCN and we want to push the image to an as small as possible scale for the largest potential speed improvement. With the generated label, we then train our scale regressor using the training data and freeze the weights of the entire network, except for the scale regressor module. We train the scale regressor for two epochs with an initial learning rate of 10^{-4} and divide by 10 after 1.3 epoch. For the testing phase, as shown in Algorithm 1, we begin every video snippet by re-sizing the first frame to 600. Then, we use the decoded regressed scale for the next frame. As for decoding the regressed scale, we first count on the inverse of (3) to obtain a scale in floating point. Then, we round it to an integer, and clip it to the range $[S_{min}, S_{max}]$.

4.3 Evaluation

To evaluate the proposed AdaScale, we progressively compare the three methods: (i) SS/SS - a detector trained and tested at 600, which is usually adopted by prior art (Ren et al., 2015; Dai et al., 2016; Zhu et al., 2017b;a; Feichtenhofer et al., 2017), (ii) MS/SS - a detector trained at S_{train} and tested at 600, and (iii) MS/AdaScale - a detector trained at S_{train} and tested on an adaptively changing scale between 128 and 600, given the range of S_{reg}. Note that the scale for MS/AdaScale can be any integer value within this range since it is predicted by the scale regressor. The evaluation results are shown in Table 1. From this point on, for the sake of simplicity, we base our analysis on ImageNet VID only. The analysis holds for mini YouTube-BB as well.

Accuracy: Compared to the baseline SS/SS, MS/AdaScale increases mAP by 1.3 points. For better visualization, blue numbers in Table 1 indicate ≥ 1 AP improvement while red numbers represent ≥ 1 AP degradation. Our approach achieves ≥ 1 AP improvements in half of the categories with only three categories having ≥ 1 AP degradation. In general, multi-scale training can enrich the training data and achieve better generalization of the model. However, this is not always the case. For categories like *red panda* and *bear*, there is a huge AP degradation for all the multi-scale training-based approaches. We find that multi-scale training could potentially lead to some confusion for certain categories. We leave the in-depth study of this phenomena to future work.

We further dive into the precision-recall curve to understand the dynamics of precision and recall for all the methods.

Table 1. Evaluation of the proposed method. We denote methods by their approach of training and testing, *e.g.*, MS/SS stands for multi-scale (MS) training and single-scale (SS) testing. Blue text and red text indicate ≥ 1 AP improvement and degradation compared to SS/SS, respectively.

(a) ImageNet VID

Method	airplane	antelope	bear	bike	bird	bus	car	cattle	dog	cat	elephant	fox	g.panda	hamster	horse
SS/SS	88.9	84.5	86.0	65.8	72.2	76.1	58.3	71.0	69.4	76.0	76.4	87.2	81.6	89.8	69.6
MS/SS	88.5	86.2	75.3	62.8	74.2	74.5	55.4	71.2	72.1	75.8	77.1	87.5	82.1	87.5	74.4
MS/AdaScale	88.2	87.0	80.2	67.4	73.7	75.3	57.8	73.4	74.1	81.7	77.7	89.1	81.5	93.5	75.6

lion	lizard	monkey	motorcycle	rabbit	r.panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra	mAP(%)	Runtime(ms)
51.9	79.1	51.2	84.0	63.4	76.8	56.3	75.6	53.9	89.5	82.4	79.0	65.1	74.5	91.3	74.2	75
57.1	78.7	51.2	83.8	61.0	58.7	61.5	68.9	57.3	89.8	81.4	78.2	64.5	74.3	89.2	73.3	75
62.6	78.7	52.2	84.6	63.6	66.4	62.2	73.0	61.0	90.7	82.3	79.7	65.6	75.6	90.4	75.5	**47**

(b) Mini YouTube-BB

Method	person	bird	boat	bike	bus	bear	cow	cat	giraffe	p.plant	horse	motorcycle	knife	airplane	skateboard	train	truck	zebra	toilet	dog	elephant	umbrella	car	mAP(%)	Runtime(ms)
SS/SS	24.9	45.3	39.3	49.1	83.1	67.8	71.8	86.5	83.7	55.0	74.4	51.8	65.1	89.9	54.2	86.7	87.1	88.5	79.7	53.5	82.8	61.1	83.5	68.0	75
MS/SS	22.4	49.4	42.0	61.7	84.2	71.0	71.3	85.1	85.9	49.5	69.3	52.1	62.1	88.8	56.1	88.1	86.8	89.2	83.1	52.5	79.9	61.5	83.4	68.5	75
MS/AdaScale	26.2	53.2	41.9	63.6	83.4	72.6	72.0	87.6	86.8	57.8	75.4	59.0	70.4	89.7	52.5	86.7	87.2	89.0	83.8	53.3	81.4	66.4	85.7	70.7	**41**

Fig. 5 shows that precision-recall curves for three most improved categories (a)-(c), one on-par category (d), and two most degraded categories (e)-(f). To give a more comprehensive analysis, we add multi-scale training and multi-scale testing (MS/MS) here for comparison. In addition, we also compare with multi-scale training and random testing scenario, which selects one of the five scales in S_{reg} randomly at test time. Compared to random scaling, MS/AdaScale clearly learns the dynamics of when and how to scale to be able to have consistently higher average precision. Additionally, we can tell from the figure that irrespective of getting better or worse compared to SS/SS, MS/AdaScale follows the curve of MS/MS closely.

Speed: Our scale regressor incurs only 2ms of overhead, which is 3% of the runtime of R-FCN. To see the speed improvement brought by the MS/AdaScale, Fig. 10(a) shows the size distribution produced by the scale regressor on ImageNet VID validation dataset and we conduct speed sensitivity analysis on scale set S_{train} in section 4.7. We note that, to profile the runtime, we warm up the GPU memory in order to remove the impact of memory allocation overhead of MXNet (Chen et al., 2015).

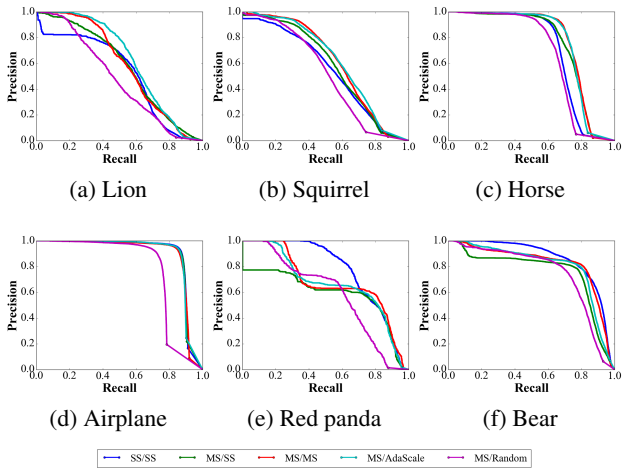


Figure 5. Precision-Recall curves for categories that MS/AdaScale has (a)(b)(c) better performance, (d) on-par performance, and (e)(f) worse performance compared to SS/SS.

4.4 Higher Precision with AdaScale

We further dig into what our method actually improves - precision or recall. As mentioned earlier in section 3.2, adaptive scaling could possibly increase true positives by scaling the object into a better scale for the detector or reduce false positives by not focusing too much on unnecessary details. To conduct this analysis, we compute the number of true positives and false positives across all the images in the validation set for method SS/SS, MS/SS, MS/MS,

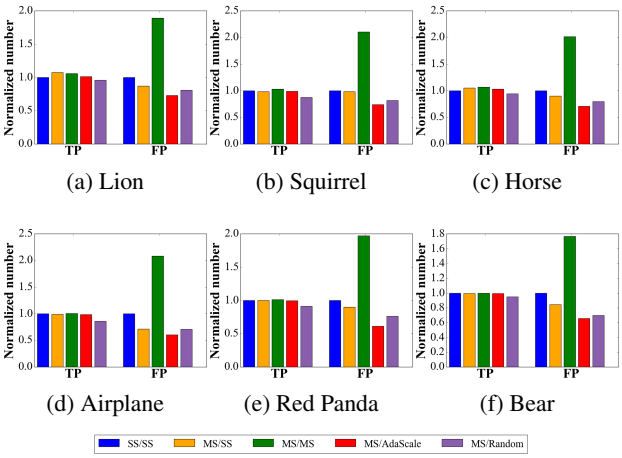


Figure 6. Normalized true positives and false positives for different methods across all the images in validation set for three selected categories.

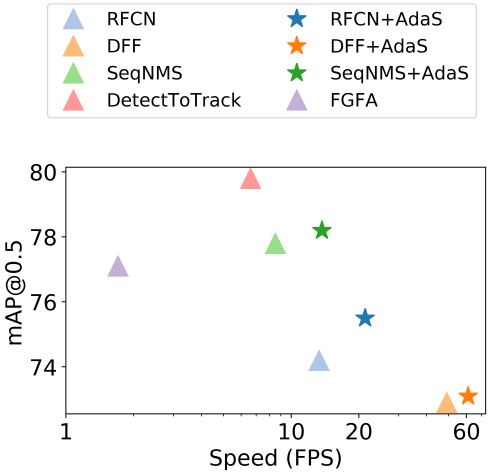


Figure 7. mAP and speed comparison with prior art on ImageNet VID dataset. Applying our AdaScale to RFCN (Dai et al., 2016), DFF (Zhu et al., 2017a) and SeqNMS (Han et al., 2016) can further improve both speed and accuracy.

MS/AdaScale, as well as MS/Random. Fig. 6 shows the number of true positives and the number of false positives normalized to method SS/SS (we present the results for all categories in Appendix). First, by comparing SS/SS and MS/SS, we can observe that multi-scale training is able to lower the number of false positives dramatically. This is reasonable since multi-scale training reduces the chance that the classifier counts on scale information as a discriminating feature. The results of MS/SS and MS/Random show that simply down-sampling images can also reduce false positive, but it reduces true positives as well. In addition to the false positive reduction brought by multi-scale training and image down-sampling, MS/AdaScale manages to reduce even more false positives, with true positives comparable to SS/SS. In general, MS/AdaScale is able to increase precision at a slight cost of recall degradation.

4.5 Qualitative Results

In Fig. 8, we show some example images for the detection results of both the baseline SS/SS and MS/AdaScale. First, we observe that the regressor learns to down-sample the image when there is a large object in the image. On the other hand, it stays in higher scales if there is a small object in the image. Also, we notice that the regressor learns to scale to the right size to avoid false positives and even correct predictions with false classes.

To understand AdaScale more in terms of the sequential decisions, Fig. 9 shows the AdaScale dynamics of three clips. Specifically, it shows that (i) it stably down-samples images with a large object; (ii) it stably scales the images into larger scales when the object is small; and (iii) it jitters when there are multiple objects with varying sizes. The scale jittering in the third clip indicates that if there are size-varying multiple objects in the frame, it is harder to decide what constitutes a better size, which can also be observed in the watercraft of Fig. 8. To enhance the current design, it is possible to apply AdaScale recursively on the attention of the given image, to obtain results from multiple regressed scales. We leave improvements of the current design to future work.

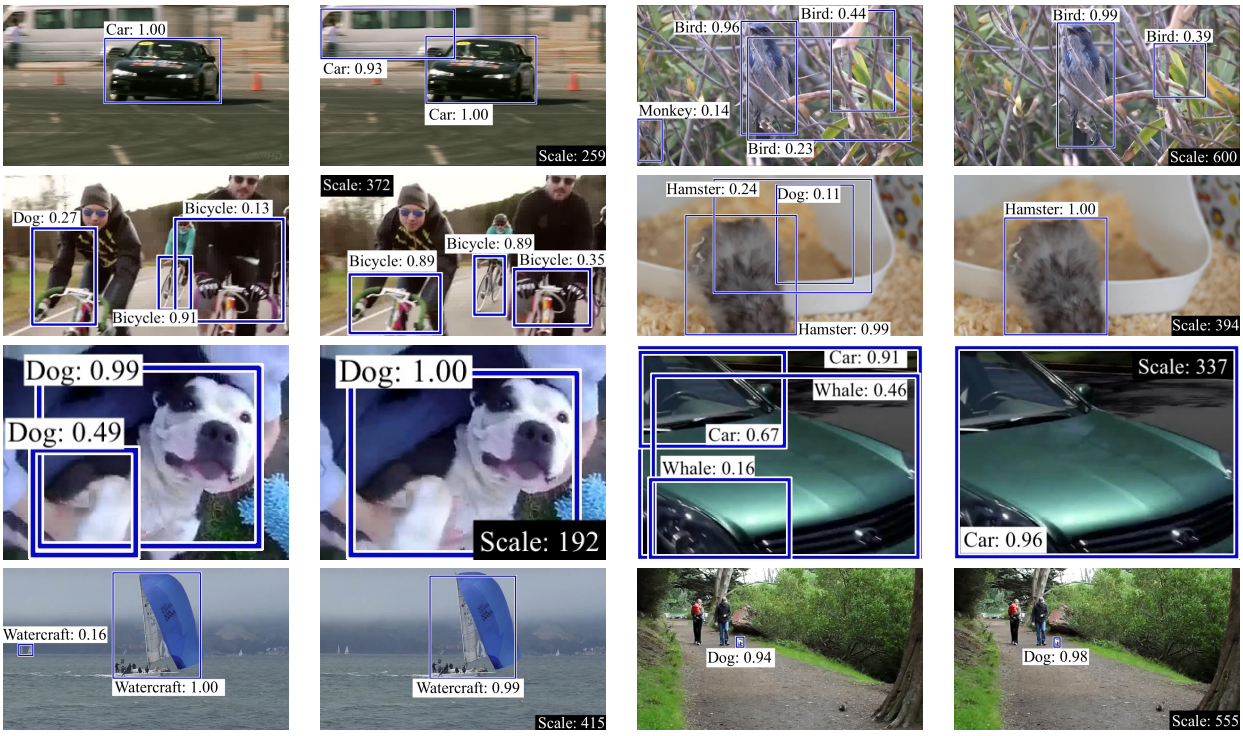
4.6 Comparison with Prior Work

To our best knowledge, our work is the first to exploit the use of images with smaller scales for improving both speed and accuracy, rather than treating them as a trade-off (Huang et al., 2017; Dai et al., 2016; Redmon & Farhadi, 2017; Chin et al., 2018; Lin et al., 2017b). For video object detection, our work is complementary to some of the prior work that tries to benefit from the detection results of multiple frames to improve accuracy or speed.

In Fig. 7, the baseline object detector is R-FCN (Dai et al., 2016) with 74.2 mAP and 13.3 frame-per-second (FPS). We run the prior work approaches (Zhu et al., 2017a;b; Han et al., 2016; Feichtenhofer et al., 2017) that provide source code for our experiment setup to profile both speed and mAP. Additionally, we combine our work with SeqNMS (Han et al., 2016) and Deep Feature Flow (DFF) (Zhu et al., 2017b) to further push the Pareto frontier by maintaining the accuracy while speeding up testing by an additional 61% and 25%, respectively.

4.7 Ablation Study

Training Scales of Object Detector: To understand how multi-scale training affects the performance of AdaScale,



(a) SS/SS (b) MS/AdaScale (c) SS/SS (d) MS/AdaScale

Figure 8. Comparing the results of SS/SS and MS/AdaScale qualitatively. Column (a) and (c) are results produced by SS/SS; column (b) and (d) are results produced by MS/AdaScale. The scales used in MS/AdaScale are labeled in black rectangle with white text.

Table 2. mAP and runtime for different multi-scale training settings.

S_{train}	{600,480,360,240}		{600,480,360}		{600,360}		{600}	
testing method	SS	Ada.	SS	Ada.	SS	Ada.	SS	Ada.
mAP (%)	73.3	75.5	73.3	74.8	73.4	74.8	74.2	74.2
runtime (ms)	75	47	75	55	75	57	75	68

we try different sets of training scales S_{train} and the results are shown in Table 2 and Fig. 10. We find that a larger set of S_{train} improves both the mAP and speed of AdaScale. From Fig. 10(a)-(d), we can also observe higher speed with smaller *training* scales. We postulate that it is due to two reasons: (i) Multi-scale trained object detector is able to generate more meaningful labels for the regressor to learn since it is less biased toward some scales. (ii) The object detector becomes good at multiple scales that could be better exploited by the scale regressor.

Table 3. mAP and runtime for different regressor architectures.

kernel size	1	1&3	1&3&5
mAP (%)	75.3	75.5	75.5
runtime (ms)	51	47	50

Regressor Architectures: We try using different sizes of filter for the regressor module and we show the results in Table 3. Interestingly, since the *accuracy* of the regressor directly affects the *speed* of the object detector, both regressor's accuracy and the overhead of the module affect the final overall speed.

5 Conclusion

Given the importance of video object detection, we present a thorough study of the possibility of improving both speed and accuracy in video object detection with adaptive scaling. Our contributions are three-fold: (i) to the best of our knowledge, our work is the first work to demonstrate the use of down-sampled images for improving both speed and accuracy for video object detection, (ii) we provide comprehensive empirical results that demonstrate improvement in both ImageNet VID as well as mini YouTube-BB datasets, and (iii) we combine our technique with state-of-the-art video object detection acceleration techniques and further

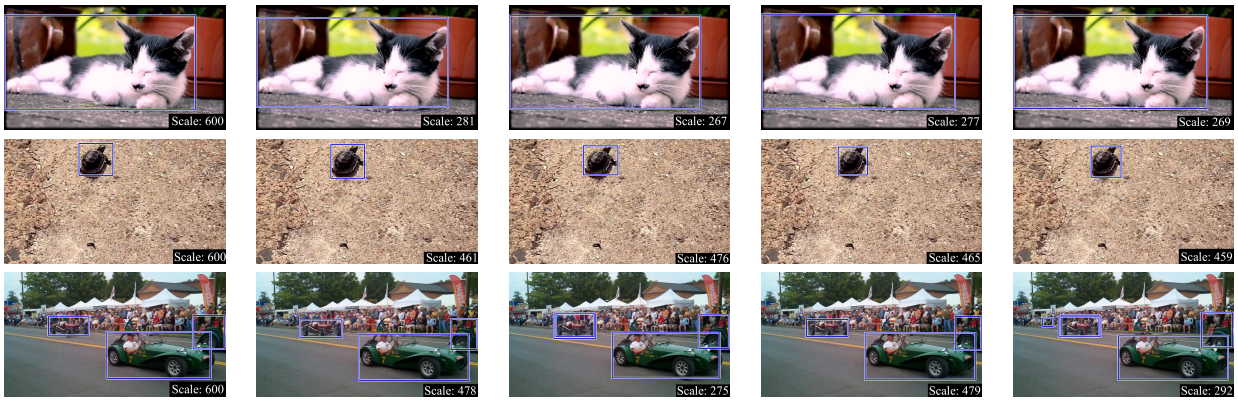
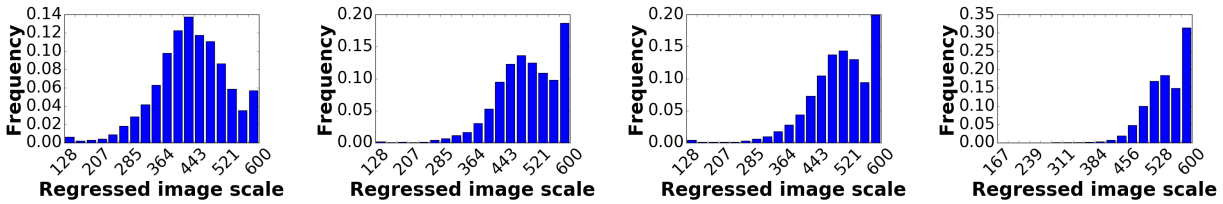


Figure 9. The investigation of the dynamics of AdaScale. The scales of the images are labeled in bottom-right.



(a) {600,480,360,240} (b) {600,480,360} (c) {600,360} (d) {600}

Figure 10. The regressed scale distribution of AdaScale tested on ImageNet VID validation set. (a)-(d) use different S_{train}.

improve the speed by an additional 25% with the added benefit of slightly higher accuracy.

ACKNOWLEDGEMENTS

This research was supported in part by National Science Foundation CSR Grant No. 1815780 and National Science Foundation CCF Grant No. 1815899. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562 (Nystrom et al., 2015). Specifically, it used the Bridges system, which is supported by National Science Foundation award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

REFERENCES

Bell, S., Lawrence Zitnick, C., Bala, K., and Girshick, R. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2874–2883, 2016.

Buckler, M., Bedoukian, P., Jayasuriya, S., and Sampson, A. Eva: Exploiting temporal redundancy in live computer vision. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 533–546, June 2018. doi: 10.1109/ISCA.2018.00051.

Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pp. 354–370. Springer, 2016.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *Proceedings of LearningSys*, 2015.

Chin, T. W., Yu, C. L., Halpern, M., Genc, H., Tsao, S. L., and Reddi, V. J. Domain-specific approximation for object detection. *IEEE Micro*, 38(1):31–40, January 2018. ISSN 0272-1732. doi: 10.1109/MM.2018.112130335.

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

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume 00, pp. 764–773, Oct. 2018. doi: 10.1109/ICCV.2017.89. URL doi.ieeecomputersociety.org/10.1109/ICCV.2017.89.

Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. Scalable object detection using deep neural networks. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147–2154, 2014.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Detect to track and track to detect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038–3046, 2017.

Girshick, R. Fast r-cnn. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 1440–1448. IEEE, 2015.

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

He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *european conference on computer vision*, pp. 346–361. Springer, 2014.

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

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, 2017.

Kanazawa, A., Sharma, A., and Jacobs, D. Locally scale-invariant convolutional neural networks. In *NIPS workshop*, 2014.

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

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *CVPR*, volume 1, pp. 4, 2017a.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. Focal loss for dense object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2980–2988, 2017b.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Liu, Y., Li, H., Yan, J., Wei, F., Wang, X., and Tang, X. Recurrent scale approximation for object detection in cnn. In *IEEE International Conference on Computer Vision*, 2017.

Nystrom, N. A., Levine, M. J., Roskies, R. Z., and Scott, J. R. Bridges: A uniquely flexible hpc resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, XSEDE '15, pp. 30:1–30:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3720-5. doi: 10.1145/2792745.2792775. URL http://doi.acm.org/10.1145/2792745.2792775.

Real, E., Shlens, J., Mazzocchi, S., Pan, X., and Vanhoucke, V. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7464–7473. IEEE, 2017.

Redmon, J. and Farhadi, A. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 6517–6525. IEEE, 2017.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Ren, S., He, K., Girshick, R., Zhang, X., and Sun, J. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481, 2017.

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

Zhou, H., Li, Z., Ning, C., and Tang, J. Cad: Scale invariant framework for real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 760–768, 2017.

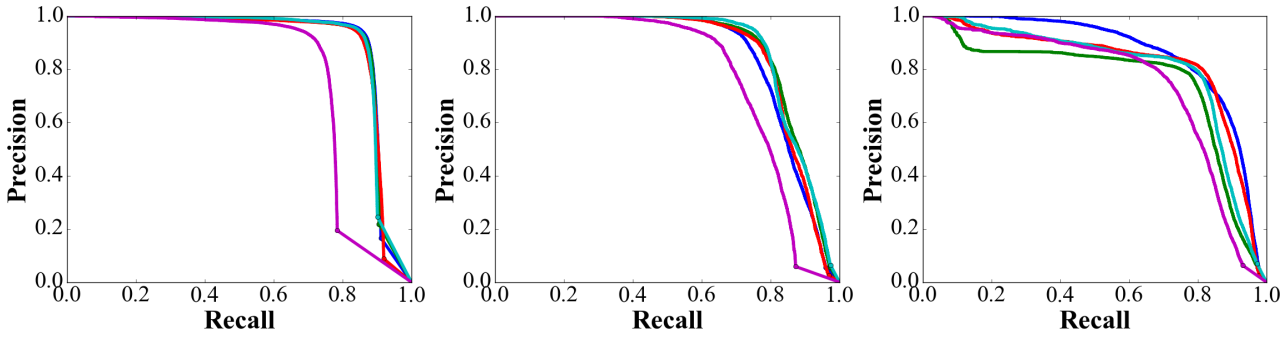
Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 408–417, 2017a.

Zhu, X., Xiong, Y., Dai, J., Yuan, L., and Wei, Y. Deep feature flow for video recognition. In *Proc. CVPR*, volume 2, pp. 7, 2017b.

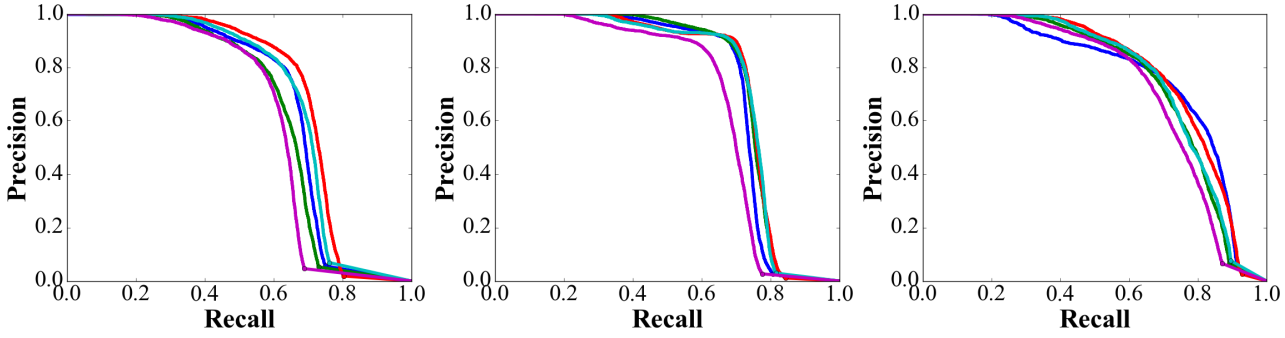
Zhu, X., Dai, J., Yuan, L., and Wei, Y. Towards high performance video object detection. In *IEEE CVPR*, 2018a.

Zhu, Y., Samajdar, A., Mattina, M., and Whatmough, P. Euphrates: Algorithm-soc co-design for low-power mobile continuous vision. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 547–560, June 2018b. doi: 10.1109/ISCA.2018.00052.

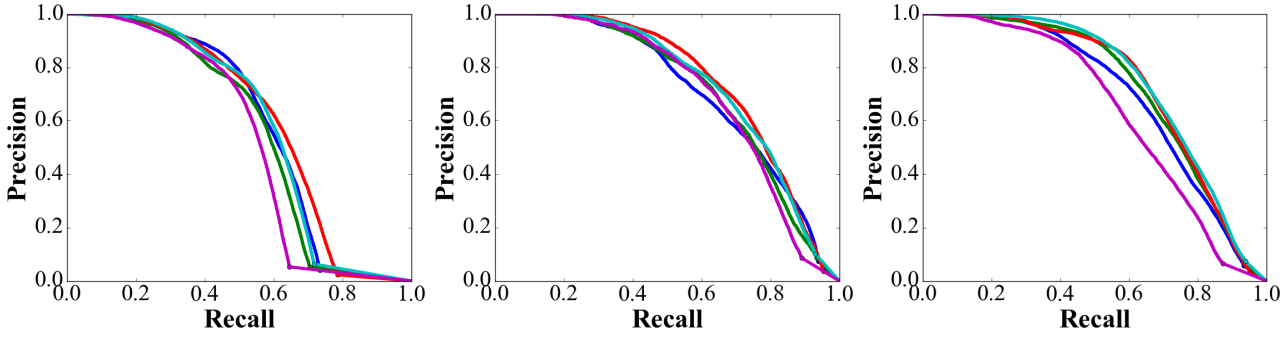
6 SUPPLEMENTARY MATERIALS



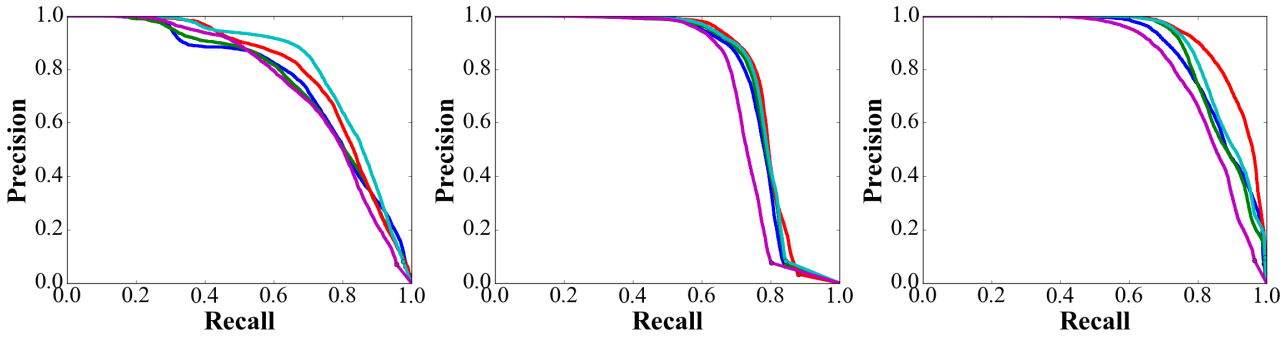
(1) airplane (2) antelope (3) bear



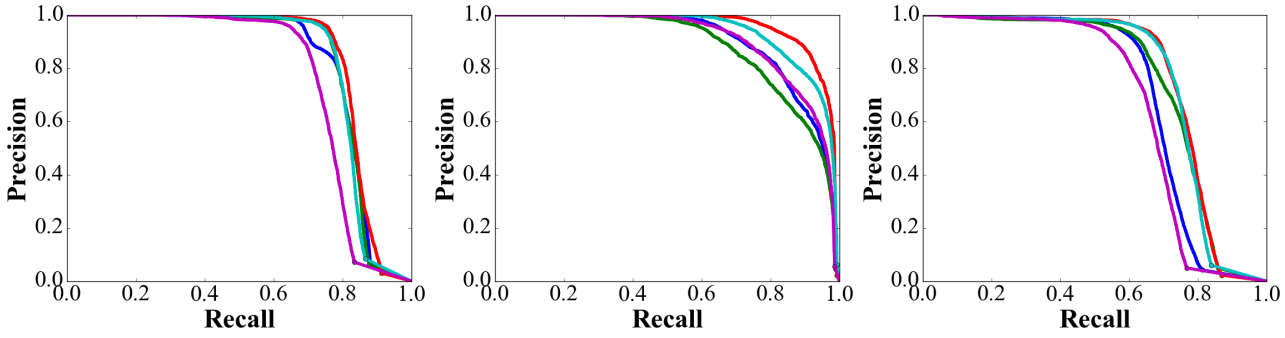
(4) bicycle (5) bird (6) bus



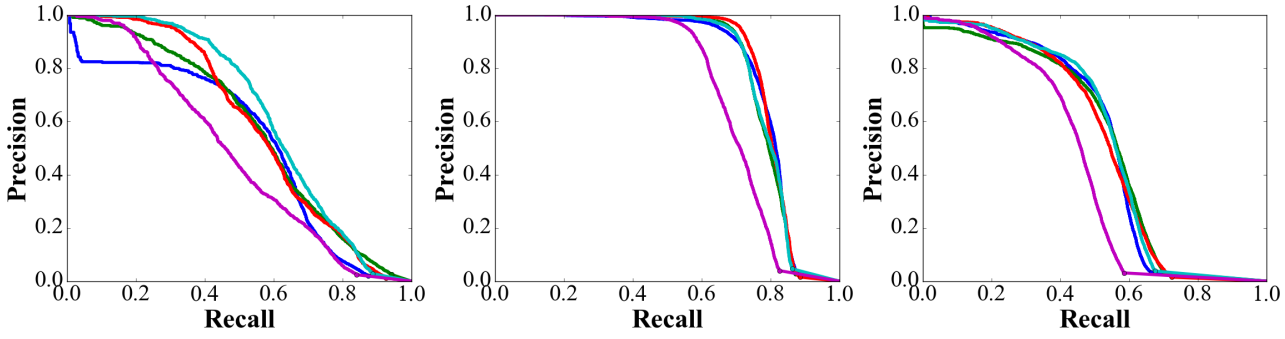
(7) car (8) cattle (9) dog



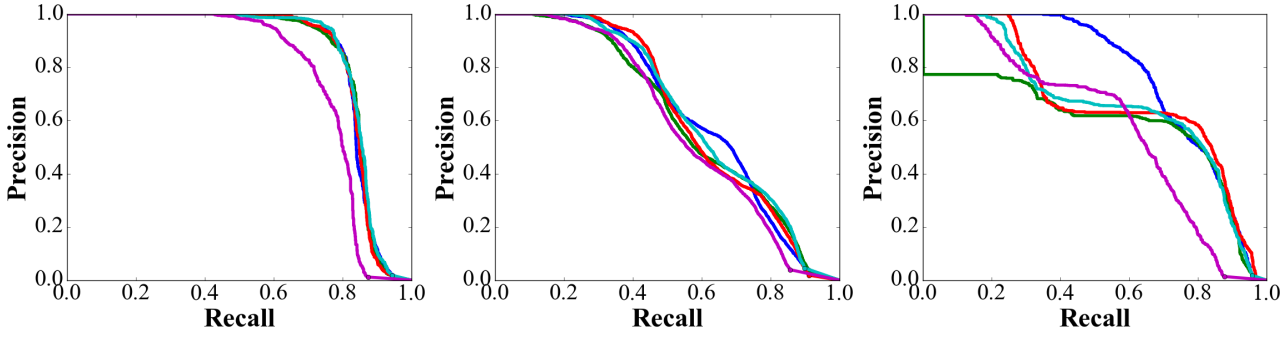
(10) domestic_cat (11) elephant (12) fox



(13) giant_panda (14) hamster (15) horse



(16) lion (17) lizard (18) monkey



(19) motorcycle (20) rabbit (21) red_panda

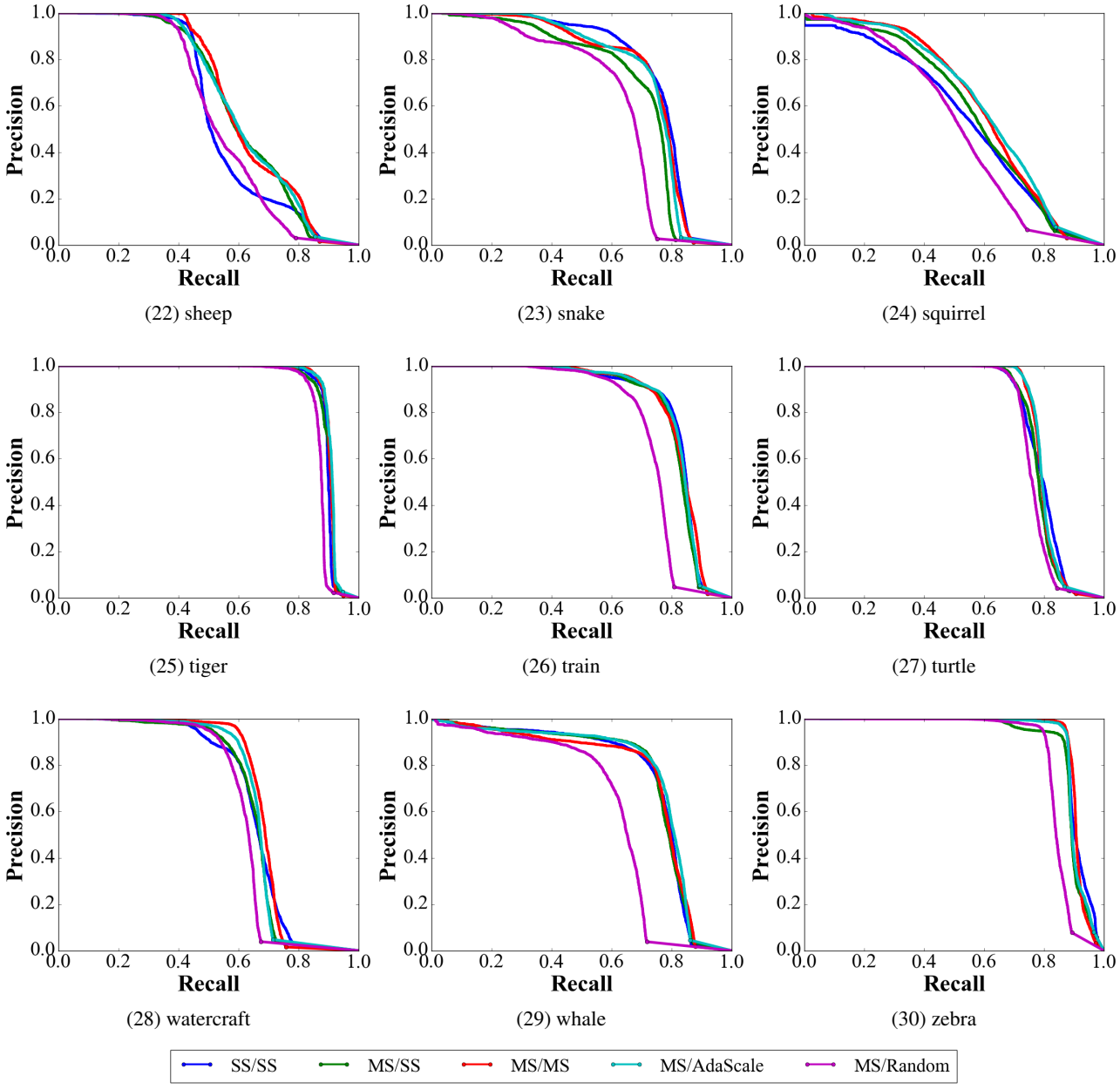
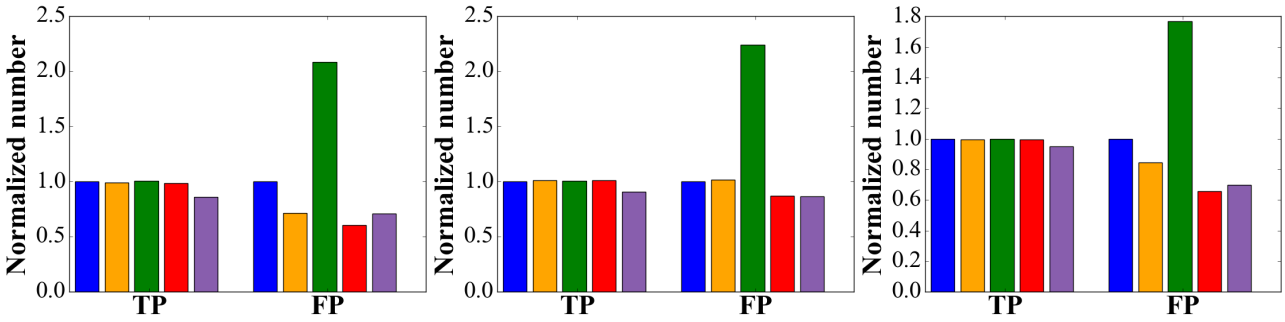
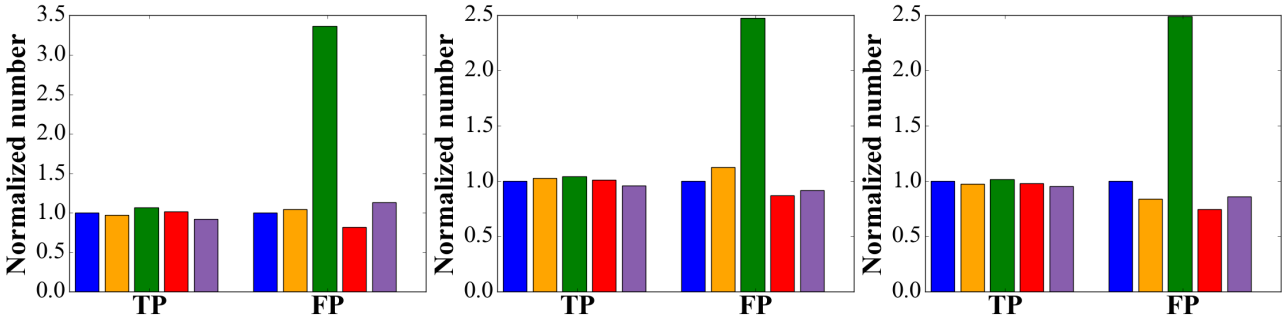


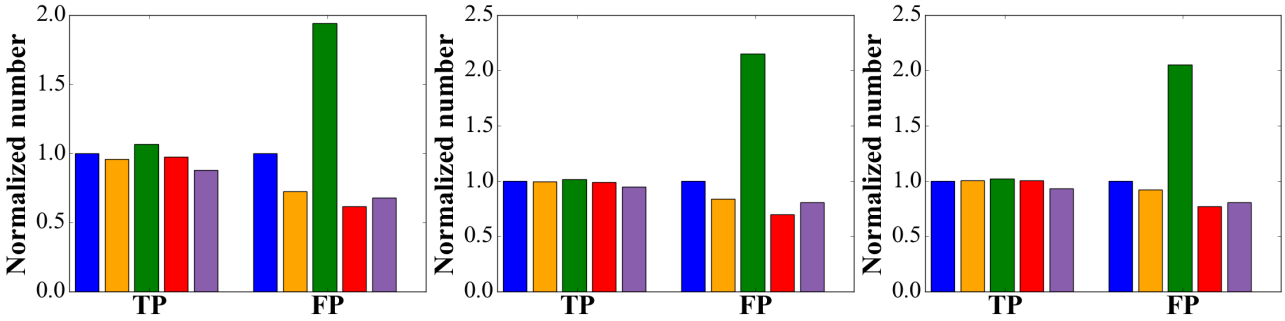
Figure 9. Precision-Recall curves for 30 categories on ImageNet VID dataset



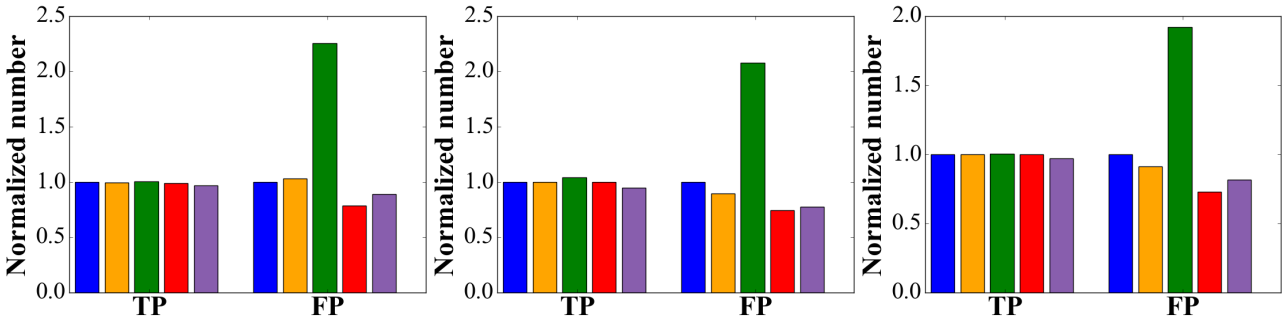
(1) airplane (2) antelope (3) bear



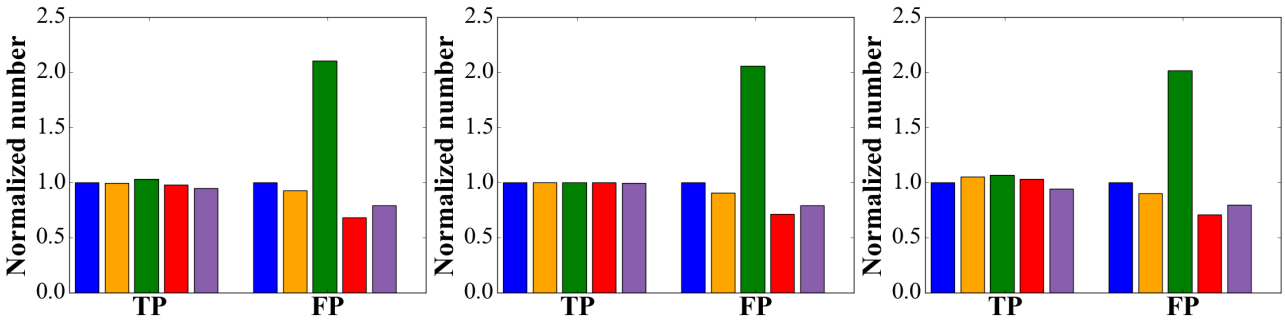
(4) bicycle (5) bird (6) bus



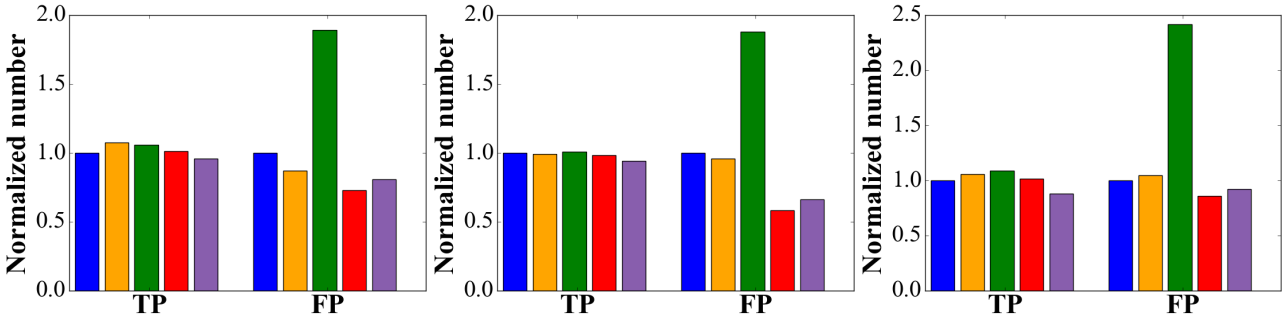
(7) car (8) cattle (9) dog



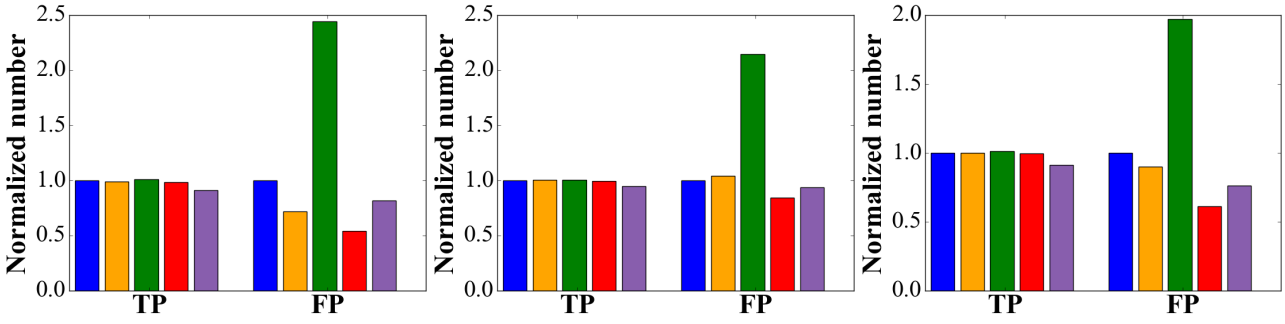
(10) domestic_cat (11) elephant (12) fox



(13) giant_panda (14) hamster (15) horse



(16) lion (17) lizard (18) monkey



(19) motorcycle (20) rabbit (21) red_panda

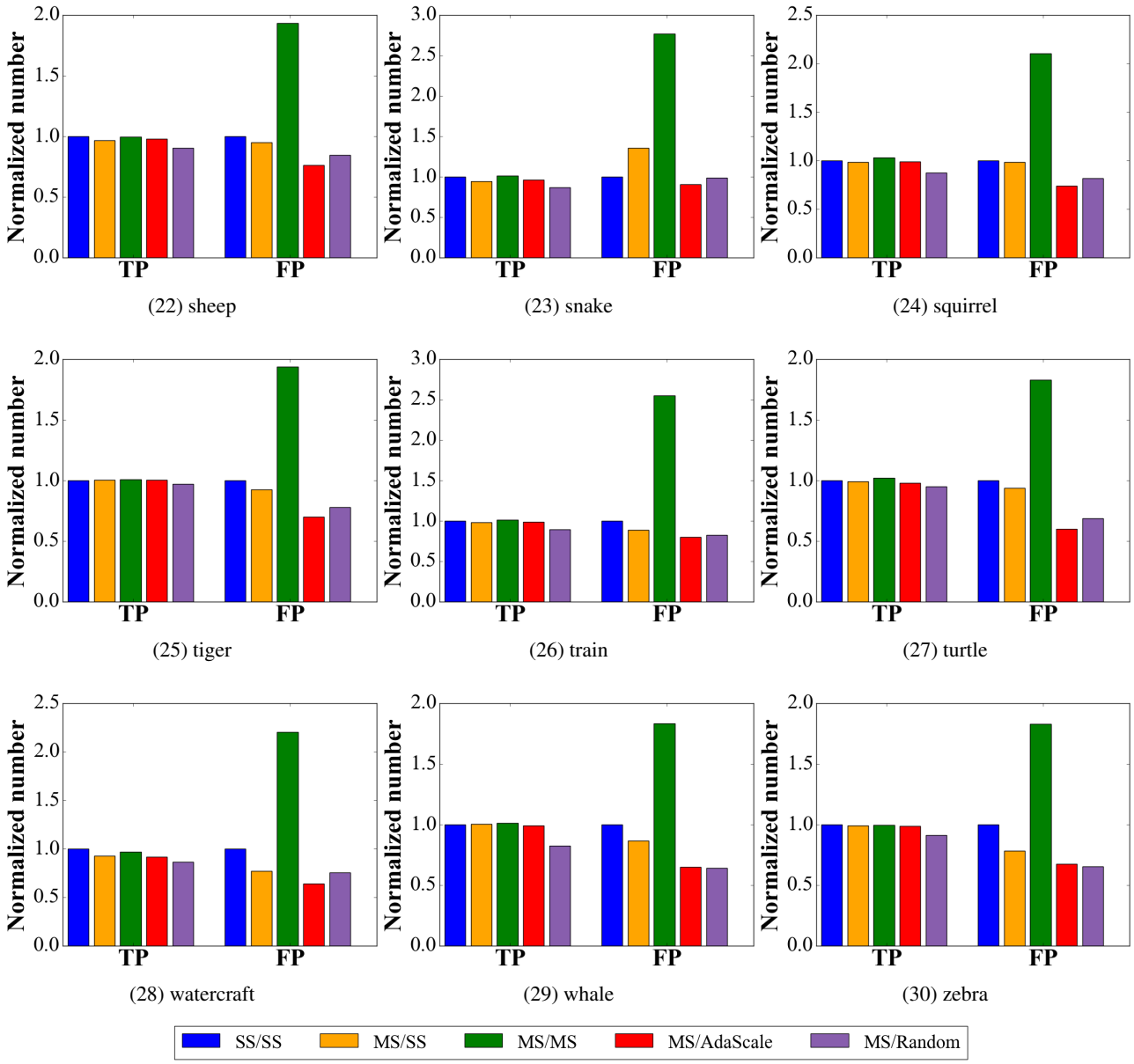


Figure 8. Normalized true positives and false positives for different methods across all the images in ImageNet VID validation set for 30 categories.