SaccadeNet: A Fast and Accurate Object Detector

Shiyi Lan1*  Zhou Ren2  Yi Wu2  Larry S. Davis1  Gang Hua2
1University of Maryland, College Park  2Wormpex AI Research
sylan@cs.umd.edu, lsd@umiacs.umd.edu, {renzhou200622, ywu.china, ganghua}@gmail.com

Abstract

Object detection is an essential step towards holistic scene understanding. Most existing object detection algorithms attend to certain object areas once and then predict the object locations. However, neuroscientists have revealed that humans do not look at the scene in fixed steadiness. Instead, human eyes move around, locating informative parts to understand the object location. This active perceiving movement process is called saccade.

Inspired by such mechanism, we propose a fast and accurate object detector called SaccadeNet. It contains four main modules, the Center Attentive Module, the Corner Attentive Module, the Attention Transitive Module, and the Aggregation Attentive Module, which allows it to attend to different informative object keypoints, and predict object locations from coarse to fine. The Corner Attentive Module is used only during training to extract more informative corner features which brings free-lunch performance boost. On the MS COCO dataset, we achieve the performance of 40.4% mAP at 28 FPS and 30.5% mAP at 118 FPS. Among all the real-time object detectors, our SaccadeNet achieves the best detection performance, which demonstrates the effectiveness of the proposed detection mechanism.

1. Introduction

The human visual system is accurate and fast. As the first gate to perceive the physical world, our visual system glances at a scene and immediately understands what objects are there and where they are. This efficient and effective vision system enables human to perceive the visual world with little conscious thought. In machine intelligence, similarly a fast and accurate object detector is essential, which can allow machines to perceive the physical world efficiently and effectively, and unlock subsequent processes such as understanding the holistic scene and interacting within it.

*This work was done when Shiyi Lan was a research intern at Wormpex AI Research.

Many recent algorithms have been proposed to advance object detection. On the one hand, anchor-based methods [24, 23, 16, 18, 7] proposed to pre-define a large amount of anchor locations, and then either directly regress object bounding box locations, or generate region proposals based on anchors and decide whether each region contains a certain object category. These methods usually achieve competitive performance since they aggregate detailed image features within each region. However, the time-consuming region proposal stage is an bottleneck of inference speed.

On the other hand, researchers proposed anchor-free detectors [13, 30, 5, 29]. This type of methods proposed to directly regress object locations by utilizing features at certain pre-defined object keypoints, either in the object center or on the bounding box edges. Most edge keypoints based methods are not fast because of the time-consuming grouping process that combines multiple detected keypoints to form a single object bounding box. The recent proposed center keypoint based detectors [29] avoid the complex
grouping process and run much faster.

Most existing object detection algorithms steadily attend to certain object areas only once and then predict the object locations. During this one time of scanning for objects, different algorithms attend to different areas, either to the anchor boxes, to the proposed object regions, to the center keypoint, or to the edge keypoints. However, neuroscientists have revealed that [4], to understand an object’s location, human do not look at the scene steadily. Instead, our eyes move around, locating informative parts to understand the object location.

Inspired by such mechanism, we propose a fast and accurate object detector, named SaccadeNet, which effectively attends to informative object keypoints, and predicts object locations from coarse to fine. Our SaccadeNet contains four main modules: the Center Attentive Module, the Corner Attentive Module, the Attention Transitive Module, and the Aggregation Attentive Module. The Center Attentive Module predicts the object center location and category. Meanwhile, for each predicted object center, Attention Transitive Module is used to predict the rough location of corresponding bounding box corners. To extract informative corner features, the Corner Attentive Module is used to enforce the CNN backbone to pay more attention to object boundaries, so that the regressed bounding boxes are more accurate. Finally, the Aggregation Attentive Module utilizes the features aggregated from both the center and the corners to refine the object bounding boxes.

SaccadeNet adopts multiple object keypoints including the center point and the corners, which encode and extract multiple levels of rich-detailed objects features. Moreover, it barely has speed loss comparing to the fastest center keypoint based detectors, since we predict object center and its corresponding corners jointly. Thus we do not need a grouping algorithm to combine them. Extensive experiments on the PASCAL VOC and MS COCO datasets have shown that SaccadeNet is fast and accurate. As shown in Figure 1, on COCO dataset when using ResNet-18 [9, 32] as the backbone SaccadeNet achieves mAP of 30.5% at 118 FPS. With DLA-34 [28], SaccadeNet achieves 40.4% mAP at 28 FPS, which is much better than other real-time detectors [22, 29].

2. Related Work

Modern object detectors can be roughly divided into two categories: anchor-based object detectors and anchor-free object detectors.

2.1. Anchor-based Detectors

After the seminal work of Faster R-CNN [24], anchors have been widely used in modern detectors. It usually contains two stages. The first-stage module is a region proposal network (RPN), which estimates the objectness probabilities of all anchors and regresses the offsets between object boundaries and anchors. The second stage is R-CNN, which predicts the category probability and refines the boundary of bounding box.

Recently, anchor-based one-stage approaches [23, 16, 18, 7] have drawn much attention in object detection because the architectures are simpler and usually run faster [23]. They remove the RPN and directly predict the categories and regress the boxes of candidate anchors. However, the performance of anchor-based one-stage detectors are usually lower than multi-stage detectors due to the extreme imbalance between positive and negative anchors during training.

2.2. Anchor-free Detectors

Recently, anchor-free detectors have become more and more popular [10, 22, 31, 12, 19, 29, 13, 5, 30, 27]. They avoid the complex design of anchors and usually run faster. The object detection is usually formulated as a keypoint detection problem so that the techniques of fully convolutional network (FCN) used in semantic segmentation [20] and pose estimation [21] can be applied for detection [29].

YOLOv1 [22] is one of the most popular anchor-free detectors. On each location of final layer of network, it predicts the bounding box, confidence of the box, and the class probability. In DenseBox [10], Huang et.al extend the FCN [20] for face and car detection. The ground truth is a 5-channel map where the first one is a binary mask for the center of object and the other four are for the bounding box size.

After the seminal work of CornerNet [13], keypoint based anchor-free object detectors have drawn much attention. In CornerNet, the FCN directly predicts the corner heatmap, an embedding and a group of offsets for each corner. The embeddings are used to group the pairs of corner to form bounding boxes and the offsets remap the corners from low-resolutional heatmap to the high-resolutional input image. A corner pooling layer is proposed to better localize corners. ExtremeNet [30] introduces a method that predicts the extreme points instead of the corners of bounding box, and the centerness heatmap is introduced for grouping step. In [5], Duan et.al. extend CornerNet by adding a center keypoint. The center keypoint is used to define a central region heuristically and then they use this region to refine the grouped corners.

To avoid the complex grouping process, CenterNet [29] directly predicts the center keypoint and the size of object. Furthermore, it replaces IoU-based Non-Maximum Suppression (NMS) by peak keypoint extraction which can be run on GPU to reduce inference time. In [26] centerness is used to represent the objectiveness of the bounding box predicted at each location. In RepPoints [2], a set of sample points is learned to bound the spatial extent of an object.
under the keypoint prediction framework.

3. SaccadeNet

It has been discovered that human eyes pick up informative parts to understand object locations instead of looking at every detail of objects [4], which makes it fast and accurate. To balance the trade-off between speed and accuracy, on top of the object center point, we use four object bounding box corner points as the informative keypoints in SaccadeNet since it naturally defines the bounding box position. SaccadeNet attends to these informative keypoints sequentially and then aggregates their features to infer accurate object locations. In this section, we will introduce four main modules of SaccadeNet respectively: the Center Attentive Module (Center-Attn), the Attention Transitive Module (Attn-Trans), the Aggregation Attentive Module (Aggregation-Attn), and the Corner Attentive Module (Corner-Attn) used in training.

3.1. Center Attentive Module

Center-Attn provides SaccadeNet the first sight of an object at its center and predicts object center keypoints. It takes the feature from CNN backbone as input and predicts the centerness heatmap. The centerness heatmap is used to estimate the categories and the center locations of all objects in the image. The number of channels in centerness heatmap is the number of categories. Figure 2 shows Center-Attn together with its output. In Center-Attn, it contains 2 convolutional layers. This 2-convolutional structure is called head module. It is a basic component for building other modules of SaccadeNet. We will describe it in details in Section 4.

We use the Gaussian heatmap as ground truth [13]. The ground-truth heatmap for keypoints is not defined as either 0 or 1 because locations near the target keypoint should get less penalization than locations far away. Suppose the keypoint is at location $X_k$, the value at location $X$ on the ground-truth heatmap is defined as $e^{\frac{\|X - X_k\|^2}{2\sigma^2}}$. $\sigma$ is set to $1/3$ of the radius, which is determined by the size of objects to ensure that all locations inside the area could generate a bounding box with at least $t$ IoU with the ground-truth annotations. We follow the previous work [13, 5, 29] and set $t$ as 0.3.

Besides, a variant of focal loss [16] is applied to assist the Gaussian heatmap:

$$L_{i,j}^h = \begin{cases} (1 - p_{i,j})^\alpha \log(p_{i,j}), & \text{if } y_{i,j} = 1 \\ (1 - y_{i,j})^\beta (p_{i,j})^\alpha \log(1 - p_{i,j}) & \text{otherwise} \end{cases}$$

where $p_{i,j}$ is the score at location $(i, j)$ of heatmap and $y_{i,j}$ is the corresponding ground truth value.

3.2. Attention Transitive Module

Attn-Trans predicts the corners for all locations of the deep feature map. The output shape is $w_f \times h_f \times 2$ for a single image, where $w_f, h_f$ indicate the width and the height of feature map, respectively. The last dimension is designed to be 2 meaning the width and height of the bounding box. After we get the width and height of bounding box for each center at location $(i, j)$, we can compute the corresponding corners as $(i - w_{i,j}/2, j - h_{i,j}/2), (i - w_{i,j}/2, j + h_{i,j}/2), (i + w_{i,j}/2, j - h_{i,j}/2), (i + w_{i,j}/2, j + h_{i,j}/2)$. In training, we adopt the L1 regression loss. With Center-Attn and Attn-Trans, SaccadeNet can generate object detections with coarse boundary.

3.3. Aggregation Attentive Module

Aggregation-Attn is proposed to attend to object center and bounding box corners again to predict a refined loca-
Aggregation-Attn is a light-weight module for object boundary refinement. Let \(w_{i,j}, h_{i,j}\) indicate the width and height prediction at \((i,j)\). Then, we calculate the corresponding top-left, top-right, bottom-left, bottom-right corners centering at position \((i,j)\) by \((i - w_{i,j}/2, j - h_{i,j}/2), (i + w_{i,j}/2, j - h_{i,j}/2), (i - w_{i,j}/2, j + h_{i,j}/2), (i + w_{i,j}/2, j + h_{i,j}/2)\). Since previous work [8] shows that bilinear sampling is helpful for the downsampled feature map, Aggregation-Attn takes the corners and center from the output of Attn-Trans, Center-Attn and samples features from the backbone output by bilinear interpolation. The structure of Aggregation-Attn is a revised head module. We change the input of the first convolutional layer and let it take features of center and corners of object as input.

Finally, Aggregation-Attn regresses the residual offsets to refine the boundary of objects by incorporating both the features from the corners and the center. The output of Aggregation-Attn consists of residual width and residual height. We adopt L1 loss to train this module.

### 3.4. Corner Attentive Module in Training

To extract informative corner features, we propose an auxiliary Corner-Attn branch (only in training) to enforce the CNN backbone to learn discriminative corner features. As shown in Figure 2, Corner-Attn uses one head module to process feature and output 4-channel heatmap including top-left, top-right, bottom-left, bottom-right corners. Note that this branch is used only during training so that it is a free lunch for the increased inference accuracy.

The training of Corner-Attn is also based on the focal loss and Gaussian heatmap. We tried agnostic and non-agnostic heatmaps, meaning whether different object categories share the same corner heatmap output or not. In our experiments, there is no significant difference between their performance. For shorter training time and easier implementation, we use agnostic heatmaps for Corner-Attn in our experiments.

### 3.5. Relation to existing methods

We will compare our work with other related work to address one of our contributions: SaccadeNet solves the issue of lacking holistic perception existed in edge-keypoint-based detectors and the issue of missing local details presented in center-keypoint-based detectors.

**Edge-keypoint-based detectors** infer objects by assembling edge-keypoints, like corners [13] or extreme keypoints [30]. They first predict edge keypoints and then use the grouping algorithm to generate object proposals. There are two possible problems that may make corner-keypoint-based fail to model holistic information: (a) Feature of corner encodes less holistic information since most corner-keypoint-based detectors [30, 5] still need feature of centers to assemble corner keypoints. (b) Corner keypoints often locate at background pixels which may encode less information than center keypoints do. Although SaccadeNet also utilizes corner keypoints for bounding box estimation, it is still able to capture holistic by inferring bounding boxes directly from center keypoints. Meanwhile, SaccadeNet is very fast since it avoids the time-consuming grouping.

**Center-keypoint-based detectors** propose objects from center points [29]. It outputs center heatmap and regresses boundary directly. However, center point may be far from the boundary of object so they may fail to estimate accurate boundary on some cases, especially for the large objects (as shown in Figure 3). On the other hand, corner keypoints are naturally proximal to the boundaries, so it may encode more local accurate information. Lack of modeling corners may be harmful for the center-keypoint-based detectors. Therefore, SaccadeNet utilizes corner keypoints to alleviate this issue so that it can estimate more accurate boundary.

SaccadeNet bridges the gap between edge-keypoint-based detectors and center-keypoint-based detectors.

### 4. Experiments

The experiments are conducted on 2 datasets, PASCAL VOC 2012 [6] and MS COCO [17]. MS COCO dataset contains 80 categories, including 105k images for training (train2017) and 5k images for validation (val2017). Pascal VOC consists of 20 categories and it contains a training set of 17k images and a validation set of 5k images. This setting is the same as previous work [13, 5, 8, 29].

#### 4.1. Implementation

**Backbone.** Our backbone consists of down-sampling layers and up-sampling layers. The down-sampling layers are from the CNN for image recognition, *e.g.* [28, 9]. The up-sampling layers use a couple of convolutional layers and skip connections to fuse high-level and low-level feature, *e.g.* [15]. We choose DLA-34 [28] and ResNet-18 [9] as the down-sampling backbone and use the up-sampling layers adopted in CenterNet [29], where deformable convolutions [32] are used. The size of the backbone output is 1/4 of the input. The high-resolution output help SaccadeNet recognize and locate small objects. For fair comparison and to illustrate the effectiveness of SaccadeNet, we keep all the settings of backbone the same as [29].

**Head module.** The head module is the basic component of building four modules of SaccadeNet as illustrated in Figure 2. We use the unified structure of 2 convolutional layers for all the head modules. The first convolutional layer is followed by a ReLU layer with a kernel.
size of $3 \times 3$ and 256-dimension output channels. The second convolutional layer uses a $1 \times 1$ kernel without activation function. Center-Attn contains one head module. The number of output channels of this module depends on the number of categories, e.g. 20 for Pascal VOC, 80 for MS COCO. Corner-Attn contains one head module which outputs a 4-channel heatmap representing the agnostic heatmap of 4 corner keypoints. Corner-Attn contains 2 head modules with 2-channel output, indicating the two directional center offset and the width and height of object, respectively. Aggregation-Attn contains one module with output of 2 channels denoting the residual offsets of width and height of object. The number of parameters of each head module is less than 200k.

**Training.** Our experiments were conducted on a machine with 4 GPUs of Geforce RTX 2080 Ti. It takes 10 days to train SaccadeNet-DLA34 and 5 days to train SaccadeNet-Res18. We use Adam [11] for network optimization. For data augmentation, we apply random flipping, random scaling (range from 0.6 to 1.3), cropping and color jittering. On MS COCO dataset, the size of input to the network is $512 \times 512$. We use a batch size of 32 (8 images on each GPU) with the initial learning rate of $1.25 \times 10^{-4}$ for 210 epochs. The learning rate is dropped to $1.25 \times 10^{-5}$ at the 181-th epoch. The same training settings are used for CenterNet [29]. We use different loss weights for the losses. The loss weights for $L_{Corner-Attn}$, $L_{Center-Attn}$ and $L_{Aggregation-Attn}$ are 1, 1, 0.1, respectively. Corner-Attn outputs center offsets and the center-corner offsets. We use 0.1 for the loss weight of center-corner offsets and 1 for the loss weight of center offsets. On PASCAL VOC 2012, we use a batch size of 32 on single GPU for training and the input shape of the network is $384 \times 384$. We set the initial learning rate to $1.25 \times 10^{-4}$ for 70 epochs. The learning rate is decreased to $1.25 \times 10^{-5}$, $1.25 \times 10^{-6}$ at the 46-th epoch, 61-th epoch, respectively. All the other settings are kept the same as our experiments on MS COCO dataset for training. We use the parameters pretrained on ImageNet [3] dataset to initialize the down-sampling layers. The parameters of up-sampling layers of backbone and head modules are randomly initialized.

**Inference.** On MS COCO dataset, the size of input image is $512 \times 512$. Flipped testing is optional for better performance. When the flipped and the original images are both used as inputs, we average the outputs of Center-Attn, Corner-Attn, Aggregation-Attn. For higher speed, we use peak-picking NMS proposed in [29] instead of IoU-based NMS for post-processing. Peak-picking NMS is a $3 \times 3$ pooling-like operator, which eliminates all non-peak activation. After NMS, we select the object proposals with top-100 centerness scores provided by Center-Attn. For Pascal VOC, we do not apply data augmentation for testing. We use Peak-picking NMS instead of IoU-based NMS.

### 4.2. Comparison with State-of-the-art Methods

Table 1 shows the comparison results of our approaches with previous work. SaccadeNet achieves state-of-the-art performance with higher speed.

SaccadeNet-DLA34 achieves 40.4 mAP at 28 FPS. It outperforms CenterNet-DLA34 [29] by 1.2% AP without visible speed loss due to the light-weight head modules. Besides, our approach outperforms the classic two-stage detector, MaskRCNN [8]. Meanwhile, we achieve approximately 3 times speed of it. Compared with RetinaNet [16], SaccadeNet-DLA34 performs approximately 4 times faster with only 0.4% drop in accuracy. As shown in Table 1, SaccadeNet-DLA34 is faster and much more accurate than YOLOv3 [23]. We compare the results of SaccadeNet-DLA34 and CenterNet-DLA34 [29] with different IoU thresholds and of different sizes. The average precision gains $+0.5$, $+0.7$ of IoU@$0.5$, IoU@0.7 and gains $+0.5$, $+0.8$, $+1.4$ of objects with small, medium, large size, respectively. SaccadeNet benefits more for high IoU and large object proposals than others. We will study how Aggregation-Attn and Corner-Attn affect the object proposals of different quality and various size in Section 4.3.1. Figure 3 shows the qualitative results of SaccadeNet and CenterNet. With the help of Aggregation-Attn, SaccadeNet is able to locate more accurate boundaries of objects.

Another version of our approach is based on ResNet-18 with deformable convolutions. SaccadeNet-Res18 is the first real-time anchor-free detector that achieves more than 30% mAP on MS COCO val2017 with speed faster than 100 FPS.

#### 4.2.1 Efficiency Study

We will discuss 4 main factors of efficiency: backbone, head modules, data augmentation, non-maximum suppression.

**Backbone.** We use DLA-34 [28] and ResNet-18 [9] with additional up-sampling layers used in CenterNet [29] as backbone. DLA-34 runs at 18.4 ms per image. ResNet-18 runs at 6.8 ms per image. The total inference time of SaccadeNet with DLA-34 and ResNet-18 is 20 ms, 8.5 ms per image, respectively. The efficiency of backbone is the major bottleneck of speed.

**Head modules.** There are $64 \times 256 \times 3 \times 3 + 256 \times C_{out}$ parameters for each head module, where $C_{out}$ denotes the number of output channels. There are only 3 head modules during inference. The largest head module is the predictor of Center-Attn, which only contains 168k parameters. The only concern is that the inputs of Aggregation-Attn depend on the outputs of Center-Attn and Corner-Attn. It may cause sequential execution that may increase the inference time. Fortunately, the execution turns out to be very fast. The inference time of all the head modules is much smaller
than the backbone, which only cost 1.5 ms and 1.6 ms for SaccadeNet-DLA34 and SaccadeNet-Res18. The performance of SaccadeNet with and without Aggregation-Attn is illustrated in Table 3. Obviously, Aggregation-Attn is important for the performance improvement.

**Data augmentation.** For better performance, we feed the network with both the flipped image and the original image. Although this technique will double the inference time theoretically, it significantly improves the performance. Figure 3 illustrates the performance of SaccadeNet with and without flip testing.

**Non-maximum Suppression.** In SaccadeNet, we replace the popular IoU-based NMS with peak-picking NMS. Peak-picking NMS performs $3 \times 3$ pooling on the output heatmap of Center-Attn. The inference time of it is less than 0.1ms. In comparison, the IoU-based NMS needs 2 ms for post-processing. Table 3 shows the comparison between IoU-based NMS and peak-picking NMS.

**4.3. Ablation Study**

In this section, we will study the characteristics of SaccadeNet. We conduct the experiments with SaccadeNet-
### Table 2

This table shows the results of SaccadeNet with or without Aggregation-Attn and Corner-Attn. We use 6 metrics of different IoU thresholds and object sizes. All experiments are conducted on Pascal VOC. For our approaches, we show both the mAP and the mAP gain (+) or loss (-) compared with the baseline.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Aggregation-Attn</th>
<th>Flip</th>
<th>NMS</th>
<th>FPS</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>52</td>
<td>37.9</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>50</td>
<td>38.8</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>28</td>
<td>39.9</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>28</td>
<td>40.7</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>IoU</td>
<td>45</td>
<td>39.3</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>IoU</td>
<td>27</td>
<td>40.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. All experiments are conducted on MS COCO val2017. PP and IoU represent peak-picking NMS and IoU-based NMS, respectively.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Aggregation-Attn</th>
<th>Flip</th>
<th>NMS</th>
<th>FPS</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>52</td>
<td>37.9</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>50</td>
<td>38.8</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>28</td>
<td>39.9</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>PP</td>
<td>28</td>
<td>40.7</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>IoU</td>
<td>45</td>
<td>39.3</td>
<td></td>
</tr>
<tr>
<td>DLA</td>
<td>✓</td>
<td>IoU</td>
<td>27</td>
<td>40.9</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4

This table shows the results of using different points for Corner-Attn on PASCAL VOC with ResNet-18. Corner represents the original SaccadeNet-Res18. Diag Pts@t (t is a float number) represents the points locating at \( p_{ct} \ast (1 - t) + p_{cr} \ast t \), where \( p_{ct} \) and \( p_{cr} \) represent the position of centers and corners. Similarly, Mid-edge Pts@t represents a points locating at \( p_{ct} \ast (1 - t) + p_{ml} \ast t \), where \( p_{ct} \) and \( p_{ml} \) indicate center points and middle points of an edge of object bounding box. Figure 4 describes the position of all points mentioned above.

<table>
<thead>
<tr>
<th>Points</th>
<th>mAP@50</th>
<th>mAP@70</th>
<th>mAP@90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corner</td>
<td>70.94</td>
<td>57.84</td>
<td>21.07</td>
</tr>
<tr>
<td>Diag Pts@0.8</td>
<td>70.92</td>
<td>57.32</td>
<td>18.27</td>
</tr>
<tr>
<td>Diag Pts@0.6</td>
<td>70.59</td>
<td>56.48</td>
<td>17.40</td>
</tr>
<tr>
<td>Diag Pts@0.4</td>
<td>70.43</td>
<td>56.11</td>
<td>17.31</td>
</tr>
<tr>
<td>Mid-edge Pts@1.0</td>
<td>70.64</td>
<td>55.85</td>
<td>17.34</td>
</tr>
<tr>
<td>Mid-edge Pts@0.8</td>
<td>70.43</td>
<td>55.33</td>
<td>17.29</td>
</tr>
<tr>
<td>Mid-edge Pts@0.6</td>
<td>70.51</td>
<td>55.10</td>
<td>16.98</td>
</tr>
</tbody>
</table>

Table 5. The results show that iterative refinement works for more accurate boundary. The finer bounding boxes get more improvement by iterative refinement. However, as a result of more sequential execution, the iterative refinement is not very efficient. Due to speed-accuracy trade-off, we only use:

**4.3.1 Benefits of Aggregation-Attn and Corner-Attn**

Our proposed Aggregation-Attn and Corner-Attn are designed to improve the quality of boundary. To study how much they affect high-quality/low-quality and large/small object proposals, we use different IoU thresholds to compute the mean average precision and evaluate it on the objects of different sizes. As shown in Table 2, larger objects and high-quality bounding boxes gain more benefits with Aggregation-Attn and Corner-Attn.

**4.3.2 Keypoint Selection**

Although our proposed SaccadeNet reveals that corners are very important for accurate boundary localization, it is still unknown whether other keypoints are helpful for bounding box regression. We try different kinds of points: middle-edge points and other inner-box points.

The middle-edge points of an object are the 4 points in the middle of 4 edges of a bounding box. We also re-place corners with points on the orthogonal lines of the bounding box. Figure 4 describes the keypoints mentioned above. We change the corners to other keypoints as inputs of Aggregation-Attn and the annotations from corners to other keypoints for Corner-Attn. Table 4 illustrates the results on Pascal VOC.

We find that the corners are the most helpful keypoints for SaccadeNet among all other keypoints except centers. We also find that keypoints closer to corners leads to higher performance for both Aggregation-Attn and Corner-Attn. One possible reason is that corners define the extent of the object and we use the bounding box for loss calculation.

**4.3.3 Does Iterative Refinement Help?**

An intuitive idea for improving SaccadeNet is to apply Aggregation-Attn iteratively. In the experiments, we use a couple of sequential modules of Aggregation-Attn. The outputs of the previous module are used as inputs in the next module. Table 5 shows the results on PASCAL VOC.

The results show that iterative refinement works for more accurate boundary. The finer bounding boxes get more improvement by iterative refinement. However, as a result of more sequential execution, the iterative refinement is not very efficient. Due to speed-accuracy trade-off, we only use
Figure 4. Purple points and yellow points denote centers and corners, respectively. On the left, the green lines denote the diagonal lines of bounding box. The blue point represents a Diag Pts computed by bilinear interpolation. On the right, the yellow points are middle points of bounding-box sides. The pink lines denote the middle line of the bounding box. The two end of middle line are two opposite yellow points. The blue point represents a Mid-edge Pts.

<table>
<thead>
<tr>
<th>Num of iter</th>
<th>mAP@50</th>
<th>mAP@70</th>
<th>mAP@90</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>71.02</td>
<td>56.42</td>
<td>18.96</td>
</tr>
<tr>
<td>1</td>
<td>70.94</td>
<td>57.84</td>
<td>21.07</td>
</tr>
<tr>
<td>2</td>
<td>71.09</td>
<td>58.18</td>
<td>21.32</td>
</tr>
<tr>
<td>3</td>
<td>71.12</td>
<td>58.42</td>
<td>20.70</td>
</tr>
</tbody>
</table>

Table 5. The table shows the results of applying iterative refinement on SaccadeNet with different IoU thresholds. All the experiments are based on ResNet-18 on PASCAL VOC. Num of iter means the number of iterations used for boundary refinement.

<table>
<thead>
<tr>
<th>Aggregation-Attn-Cls</th>
<th>mAP@50</th>
<th>mAP@70</th>
<th>mAP@90</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>70.92</td>
<td>57.49</td>
<td>18.96</td>
</tr>
</tbody>
</table>

Table 6. This table shows the results of using Aggregation-Attn for classification with different IoU thresholds. All Experiments are performed on Pascal VOC with ResNet-18.

4.3.4 Does Aggregation-Attn also Help Classification?

Object detection is the step to understand “what is where”. We have validated that Aggregation-Attn improves the localization of object by fusing feature of corner and center keypoints, namely it helps in terms of “where”. Now we want to study whether such information aggregation also helps in terms of “what”. We add another module, namely Aggregation Attentive Classifier (Aggregation-Attn-Cls) to refine classification scores. Its structure is the same as Corner Attentive Module. We use the classification scores to replace the original object classifier output. Table 6 illustrates the results. Unfortunately, the performance is degraded by Aggregation-Attn-Cls. One possible reason is that the feature of corner keypoints encode little high-level discriminative information for classification.

5. Conclusion

We introduce SaccadeNet, a fast and accurate object detection algorithm. Our model actively attends to informative object keypoints from the center to the corners, and predicts the object bounding boxes from coarse to fine. SaccadeNet runs extremely fast, because these object keypoints are predicted jointly so that we do not need a grouping algorithm to combine them. We extensively evaluate SaccadeNet on PASCAL VOC and MS COCO datasets, which both demonstrates its effectiveness and efficiency.

6. Acknowledgement

Gang Hua was supported partly by National Key R&D Program of China Grant 2018AA0101400 and NSFC Grant 61629301. We deeply appreciate the help of Xingyi Zhou and Zuxuan Wu.
References


[29] Xingyi Zhou, Dequan Wang, and Philipp Krahenbuhl. Objects as Points. *arXiv*, 2019. 1, 2, 3, 4, 5, 6
