Circular Anchor SSD

Fred Lu

Outline

Original Mobilenet SSD Structure

Designing the Circular Anchor Box Variation

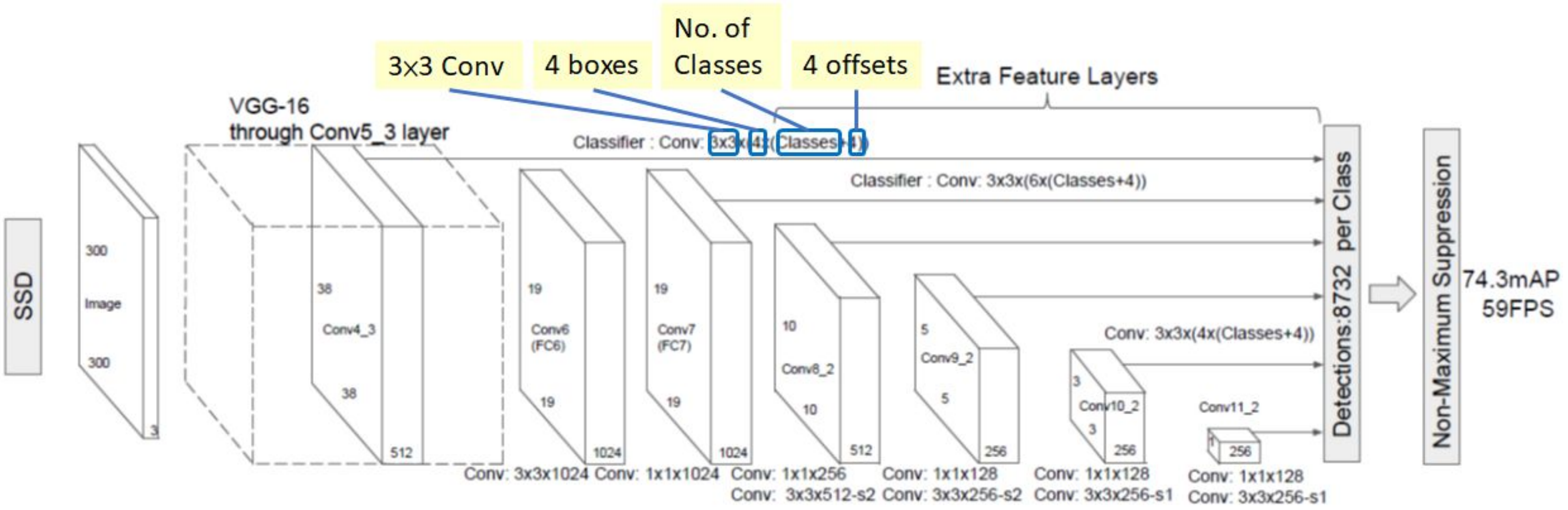
- Changes to Training
- Changes to Inferencing

Results

Potential Room for Improvement

Original Mobilenet SSD

Sample Diagram of SSD with VGG-300



Original Mobilenet SSD

The output has two parts:

1. Class logits representing confidence of object in an anchor being of a specific class

   #out channels $_{\text{classes}}$ = anchor boxes per location * #classes

2. Anchor box offsets representing the location and size of the object in relation to the anchor box

   #out channels $_{\text{box offsets}}$ = anchor boxes per location * 4: (dx, dy, d width, d height)

In order to have circular anchor boxes, we need to modify this setup a little bit.

Circular Anchor Mobilenet SSD

Goal

Make anchor boxes and output bounding boxes to be representative of circles.

Circular Anchor Mobilenet SSD

SSD Output

Since we're only changing the shape of the anchor boxes, the class logits remains the same dimensions:

#out channels $_{\textbf{classes}}$ = anchor boxes per location * #classes

Since the anchors are now circular and we want output bounding boxes to be also circular,

#out channels $_{\textbf{box offsets}}$ = anchor boxes per location * 3: (dx, dy, d radius)

Which allows us to represent output boxes as circles while reducing computations and parameters.

Circular Anchor Mobilenet SSD

SSD Output

Old: For each location of backbone's output, output box d(x, y, w, h) relative to the anchors in the location

```
def reg_block(self, level, out_channels, boxes_per_location):
    return nn.Conv2d(out_channels, boxes_per_location * 4, kernel_size=3, stride=1, padding=1)
```

New: For each location of backbone's output, output box d(x, y, r) relative to the anchors in the location

```
def reg_block(self, level, out_channels, boxes_per_location):
    return nn.Conv2d(out_channels, boxes_per_location * 3, kernel_size=3, stride=1, padding=1)
```

Circular Anchor Mobilenet SSD

Anchor Box Generation: Same Dims, but w = h

Since anchors boxes are just numbers (x, y, w, h), circular just means making w and h equal.

The aspect ratio configuration is replaced with zoom levels. Default max/min anchors unchanged.

```
_C.MODEL.PRIORS.MIN_SIZES = [60, 105, 150, 195, 240, 285]
_C.MODEL.PRIORS.MAX_SIZES = [105, 150, 195, 240, 285, 330]
_C.MODEL.PRIORS.ASPECT_RATIOS = [[2, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2, 3]]
_C.MODEL.PRIORS.BOXES_PER_LOCATION = [6, 6, 6, 6, 6, 6]
```

```
_C.MODEL.PRIORS.MIN_SIZES = [60, 105, 150, 195, 240, 285]
_C.MODEL.PRIORS.MAX_SIZES = [105, 150, 195, 240, 285, 330]
_C.MODEL.PRIORS.ADDITIONAL_ZOOMS = [0.3, 0.7]
_C.MODEL.PRIORS.BOXES_PER_LOCATION = [4, 4, 4, 4, 4, 4]
```

Circular Anchor Mobilenet SSD

Anchor Box Generation: Same Dims, but w = h

```
size = self.min_sizes[k]
h = w = size / self.image_size
for ratio in self.aspect_ratios[k]:
    ratio = sqrt(ratio)
    priors.append([cx, cy, w * ratio, h / ratio])
    priors.append([cx, cy, w / ratio, h * ratio])
```

```
min_h = min_w = min_size / self.image_size
max_h = max_w = max_size / self.image_size
for zoom in self.additional_zooms:
    # side_length(zoom) means side_length with zoom as a percentage from min size to max size
    side_length = min_size + zoom * (max_h - min_h)
    # TODO: Finish this and change IOU measure
    priors.append([cx, cy, side_length, side_length])
```

Circular Anchor Mobilenet SSD

Training: Different Loss function

The square ground truth boxes still have 4 offsets but the predictions only have 3. The Smooth L1 Loss function have to operate on the same dimensions. -> Remove the height offset from ground truth.

Trim dimension 2 of square ground truth boxes to preserve only the first three values (dx, dy, d width).

```
gt_boxes = torch.narrow(gt_boxes, 2, 0, 3)
```

Circular Anchor Mobilenet SSD

Training: Different Loss function

Now that GT and prediction bounding boxes have the same dimensions (# batches, # boxes per location, 3), flatten and run smooth L1 loss.

```
pos_mask = labels > 0
predicted_locations = predicted_locations[pos_mask, :].view(-1, 3)
gt_locations = gt_locations[pos_mask, :].view(-1, 3)
smooth_l1_loss = F.smooth_l1_loss(predicted_locations, gt_locations, reduction='sum')
```

Circular Anchor Mobilenet SSD

Inferencing: Expanding BBox Dimensions to 4

To be able to evaluate performance indices such as IOU against GT Boxes, we need to convert the 3-offset circular bounding boxes produced by SSD to the conventional 4-offset square representations.

Circular Anchor Mobilenet SSD

Inferencing: Code

Locations: Raw SSD Bounding Box Offsets d(x, y, radius). **Priors**: anchor boxes (x, y, w, h=w)

The logic for adding offsets (dx, dy) to anchors in the first line after cat() remains the same.

In the second line which gives the final box width and height, instead of doing element-wise multiplication of locations(w,h) * priors(w,h) like before, we just scale pirors(w,h) by a scalar, locations(d radius).

```
return torch.cat([
    locations[..., :2] * center_variance * priors[..., 2:] + priors[..., :2],
    torch.exp(locations[..., 2:] * size_variance) * priors[..., 2:]
], dim=locations.dim() - 1)
```

Circular Anchor Mobilenet SSD

Inferencing: Scaling and Drawing Bounding Boxes

Now that we have converted the SSD's output into conventional bounding boxes that can drawn on an image, it's time to show these boxes.

But there is one slight problem...

Circular Anchor Mobilenet SSD

SSD has a fixed input image size of 300x300, on which the bounding boxes are generated.

When scaling the SSD's bounding boxes back to the original image size, the height and width are no longer equal. As a result, the bounding boxes will be ellipses (or rectangles).

Circular Anchor Mobilenet SSD

To fix the scaling distortion, set both the w and h of the scaled bounding ellipses to min(w,h):

Results

mAP on VOC 2007 Testing Images

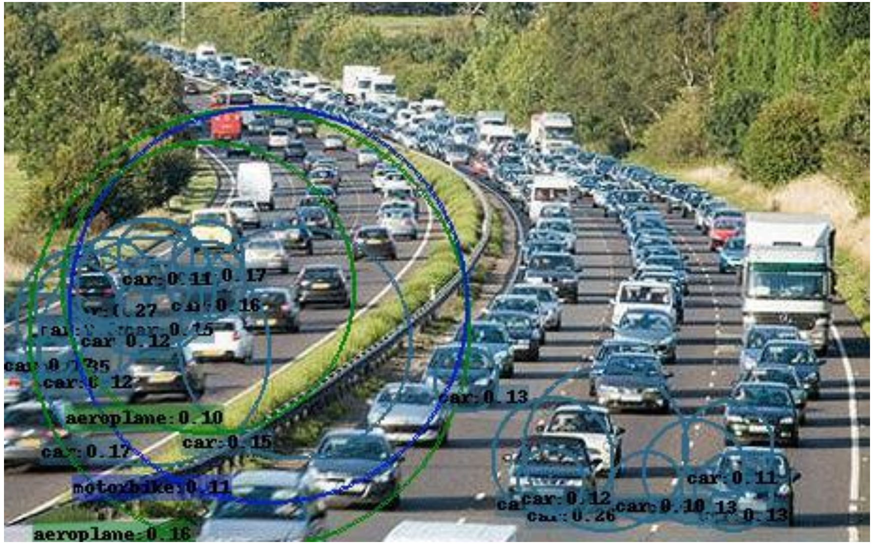
```
mAP: 0.4166
aeroplane       : 0.3834
bicycle         : 0.5246
bird            : 0.3553
boat            : 0.2318
bottle          : 0.0912
bus             : 0.5595
car             : 0.5498
cat             : 0.6720
chair           : 0.1535
cow             : 0.3686
diningtable     : 0.4292
dog             : 0.5667
horse           : 0.6152
motorbike       : 0.5634
person          : 0.2868
pottedplant     : 0.1353
sheep           : 0.4198
sofa            : 0.4566
train           : 0.5701
tvmonitor       : 0.3985
```

Results

Sample Output Images showing circles with class confidence > 0.1

Results

Sample Output Images showing circles with class confidence > 0.1



Room for Improvement

- **Could draw bounding circles with avg (or max) of scaled (w,h)**
- **Calculate IOU for circles instead of squares when generating class confidences for anchors.**
- **Give equal weight to box position and size offsets in Smooth L1 Loss.**
- **Test Different zoom configs for generating anchors for each location.**

Questions

Thank You