YOLO9000 (v2)

↳ 9000 obj categories
↳ trained simultaneously on ImageNet and COCO.

Better

→ Batch normalization after all conv layers. It provides regularization and let's them remove dropout w/o overfitting.
↳ This time they trained the network w/ 448×448 inputs instead of directly using ImageNet trained w/ 224×224 imgs.
- Predicting offsets instead of w,h makes learning easier for the network.
↳ Replaced fully connected layers w/ anchor boxes to predict bboxes. Output feature map: 13×13 (↑)
↳ Dimension clusters: choose anchor box sizes using k-means
↳ dist func: d(box, centroid) = 1 − IoU(box, centd)

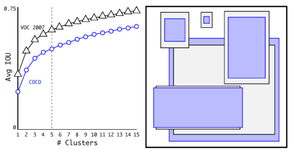
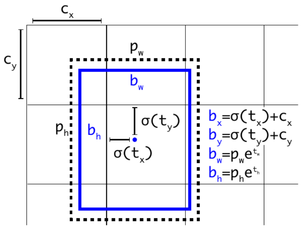


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

→ Network predicts 5 bboxes at each cell
For each bbox, net predicts 5 coordinates:
t_x, t_y, t_w, t_h, t_o
↓
cell offset from img's top left: c_x, c_y
↳ bbox prior has width, height: p_w, p_h

$b_x = \sigma(t_x) + c_x$ $\quad b_y = \sigma(t_y) + c_y$
$b_w = p_w e^{t_w}$ $\quad b_h = p_h e^{t_h}$

$Pr(obj) \times IoU(b, obj) = \sigma(t_o)$



→ Since it uses only conv and pooling layers, images can be resized on the fly. Model is trained for that, too. Every 10 batches net changes img dimension

	YOLO								YOLOv2
batch norm?			✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4		65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

↙ Darknet 19

→ First, Net is trained for classification. Then, last conv layer is removed and 3×3 and 1×1 layers added. #filters = #outputs
For VOC, they predict 5 boxes w/ 5 coordinates each 200 classes per box, so 125 filters

Stronger

↳ They used WordNet to combine COCO dataset w/ ImageNet dataset.