YOLO9000: Better, Faster, Stronger

- [[📚 1612.08242.pdf]]
- **Basic Idea**:
  - Can we improve the performance of one-stage detector (YOLO)?
    - **Better**: In comparison to Fast R-CNN makes more localization error. Also has a lower recall (TPR) i.e., unable to identify boxes with classes. Can we improve recall as well as reduce localization error?
    - **Stronger**: Being able to detect 9000 object categories.
    - **Faster**
-
- **Stronger**
  - **Batch Normalization**:
    - 🟡 **P2** Batch normalization also helps regularize the model. With batch normalization we can remove dropout from the model without overfitting
-
  - **Fine-grained learning (High-Resolution Classifier)**:
    -
      - *Learning fine-grained features.*
        Image classification networks trained at 224×224 whereas the detection network trained at 448×448.
        🟡 **P4** Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image
        - *The classic problem of retaining the dense spatial information throughout the network which often gets lost in the classification networks due to max-pooling operation.*
    - 🟡 **P2** his means the network has to simultaneously switch to learning object detection and adjust to the new input resolution
    - **Solution:** 🟡 **P2** For YOLOv2 we first fine tune the classification network at the full 448 × 448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. We then fine tune the resulting network on detection.
  - **Fine-grained learning (Detecting Smaller objects)**:
    -
      - *Detection of smaller objects*
        🟡 **P4** YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds
        Also don't have any mechanism for multi-scale detection.
    - Feature-map output size for YOLO: 13×13 (sufficient for large objects)
    - 🟡 **P3** Faster R-CNN and SSD both run their proposal networks at various feature maps in the network to get a range of resolutions.
    - **Solution:** Take feature map from previous layer ($26 \times 26 \times 512$) and resize it into $13 \times 13 \times 2048$ and concatenate with the original feature map of the last layer channel-wise. 🟡 **P4** Our detector runs on top of this expanded feature map so that it has access to fine grained features.
-

- **Fine-grained learning (Multi-scale training)**:
  - **Objective**: 🟡 **P4** YOLOv2 to be robust to running on images of different sizes so we train this into the model
  - **Solution**: 🟡 **P4** Instead of fixing the input image size we change the network every few iterations. Every 10 batches our network randomly chooses a new image dimension size. Since our model downsamples by a factor of 32, we pull from the following multiples of 32: {320, 352, ..., 608}. Thus the smallest option is 320 × 320 and the largest is 608 × 608. We resize the network to that dimension and continue training.

- **Anchor Boxes**:
  - Use hand-picked anchor boxes (priors) like Faster R-CNN for predicting bounding boxes (offsets wrt to these anchor boxes instead of predicting the coordinates directly). 🟡 **P2** Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn. This is because convolutions have local information, therefore knowing the coordinates wrt to whole image would be difficult.
  - 🟡 **P2** We also shrink the network to operate on 416 input images instead of 448×448. We do this because we want an odd number of locations in our feature map so there is a single center cell. Objects, especially large objects, tend to occupy the center of the image so it's good to have a single location right at the center to predict these objects instead of four locations that are all nearby.
  - *YOLO limitation*: 🟡 **P4** each grid cell only predicts two boxes and can only have one class.
    🟡 **P2** When we move to anchor boxes we also decouple the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box. Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object.
  - 🟡 **P2** Without anchor boxes our intermediate model gets 69.5 mAP with a recall of 81%. With anchor boxes our model gets 69.2 mAP with a recall of 88%.

- **Data-driven anchor box (Auto-anchor):**
  - 🟡 **P2** Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automat-🟡 **P3** ically find good priors. If we use standard k-means with Euclidean distance larger boxes generate more error than smaller boxes. However, what we really want are priors that lead to good IOU scores, which is independent of the size of the box. Thus for our distance metric we use

$$\mathrm{d}(\text{box}, \text{centroid}) = 1 - \mathrm{IOU}(\text{box}, \text{centroid})$$

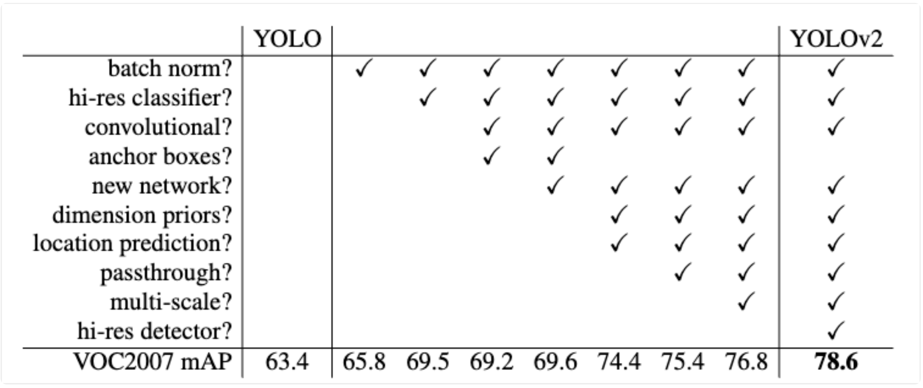
  - 🟡 **P3** We choose k = 5 as a good tradeoff between model complexity and high recall. The cluster centroids are significantly different than hand-picked anchor boxes. There are fewer short, wide boxes and more tall, thin boxes. In order to achieve 100% recall on the training dataset, we can set k to be equal to the number of all possible bounding boxes in the training dataset.
  - *Efficacy of auto-anchor*: Measured by calculating average IoU between the training set bounding boxes and the anchor boxes. 🟡 **P3** At only 5 priors the centroids perform similarly to 9 anchor boxes with an average IOU of 61.0 compared to 60.9. Therefore, with 5 (instead of 9) anchor boxes we can save number of parameters.

- *Downside of auto-anchor*: Lack of generalization. The downside of learning anchor boxes is that it may not transfer well to other datasets.

- **Direct Location Prediction:**
  - The network training with anchor boxes is highly unstable during the early iterations due to unconstrained prediction of centre of the bounding box. P3 This formulation is unconstrained so any anchor box can end up at any point in the image, regardless of what location predicted the box. With random initialization the model takes a long time to stabilize to predicting sensible offsets.
  - P3 Instead of predicting offsets we follow the approach of YOLO and predict location coordinates relative to the location of the grid cell. This bounds the ground truth to fall between 0 and 1. We use a logistic activation to constrain the network's predictions to fall in this range. This is because the centre of the object has to lie within the grid cell.
  - P3

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

- **Results**
  - P5



	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**

Unlinked References