YOLO

August 5, 2020

1 "Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016."— CVPR 2016, OpenCV People's Choice Award

This report is about You Only Look Once (YOLO) a real-time object detection system, which may not be the most accurate object detector system, but probably is the one of the most popular object detectors for real-time engineering applications. At this date, the paper has 10327 citations. In this report I will talk about how YOLO works and what makes it so popular in computer vision and robotics community.

1.1 Why is YOLO important? Why did it get accepted? Why is it so popular?

1. This paper purposes an simpler alternative to sliding-window or region proposal methods for object detection, which is faster to optimize. Unlike them, it uses entire image during the training which lowers the amount of false positives.
2. Authors claim that other fast approaches can be optimized for special purposes (pedestrians etc.) thanks to small variability, while YOLO is highly generalizable, and can deal with new domains or unexpected inputs.
3. The authors proved their claims by comparing YOLO to R-CNN and DPM. In addition to prove their point about generalizability, they applied YOLO to artwork.
4. Speed and generalizability are very important for engineering. Cameras are used as an affordable and sufficiently accurate alternative to lidars. Therefore, many systems are vision based. Many methods are intended for a real-time use with limited processing resources(limited processing power, limited memory etc.) on portable systems. If YOLO's detection error can be compensated by a system, then the developers will prefer it over a more accurate CNN due to YOLO's fast speed. Especially, a robot will rarely need a pixel-wise accuracy from an object detection system for localization, mapping, navigation tasks. It might only need that high accuracy for manipulation. This is why YOLO is important for robotics society and is popular among robotic applications.

1.2 What is YOLO? How does it work?

YOLO defines the object detection as a single regression problem, which outputs detected object bounding box coordinates and class probabilities. It uses features from the entire image to predict all

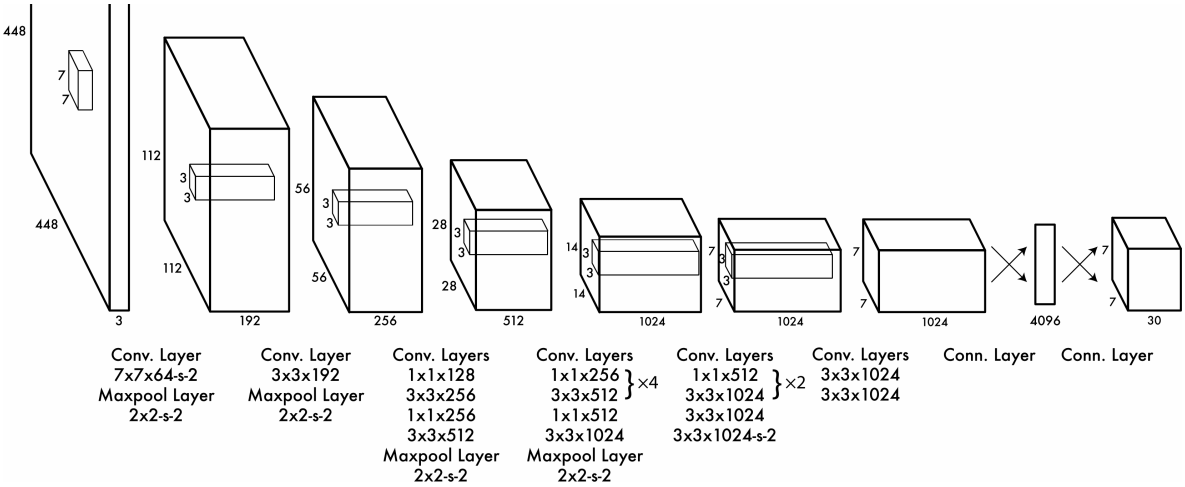

Figure 1: The architecture (from the paper)

bounding boxes across all classes simultaneously.

1. An input image is divided in to $S \times S$ grid.
2. Each grid predicts B bounding boxes and confidence score for bounding boxes for objects whose center falls in the grid.
3. The confidence score is calculated.

$$Pr(Object) * IntersectionOverUnion_{prediction}^{groundtruth} \tag{1}$$

   If there is any any object in the grid cell, then the confidence should be zero.

4. Each bounding box has 5 predictions: (x, y) center of the bounding box relative to the grid, (w, h) width and height relative to whole image, confidence for IOU between predicted box and the ground truth box.
5. Each grid also predicts C conditional class probabilities for class prediction given the object as

$$Pr(Class_i|Object) \tag{2}$$

6. At the test time box probability 1 and conditional class probability 2 are multiplied to give confidence score both for probability of the class and how well the box fits the object.

$$\begin{aligned} Pr(Class_i|Object) * Pr(Object) * IntersectionOverUnion_{prediction}^{groundtruth} = \\ Pr(Class_i) * IntersectionOverUnion_{prediction}^{groundtruth} \end{aligned} \tag{3}$$

7. Predictions are encoded as a tensor as

$$S \times S \times (B * 5) + C \tag{4}$$

8. As it can be seen at the figure 1 the detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 layers are for reduction of the feature space.

9. The authors also implemented a faster version named FastYOLO, which has the exact same parameters except it has a smaller size network (9 layers instead of 24).

10. The convolutional layers were pretrained for a week on the ImageNet 1000-class competition set on 20 convolutional layers followed with a max pooling and a fully connected layer. The authors achieved %88 accuracy on the Image Net 2012 validation set.

11. The authors added four convolutional layer and two fully connected layers with randomly initialized weights to improve the performance. They also increased the input size from 224×244 to 448×448 for detection. The bounding box width and height are normalized to fall between 0 and 1.

12. Activation function is leaky rectified linear activation function.

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{5}$$

13. The authors optimized the sum squared error, but had trouble because it ended up weighting localization error with classification error and made confidence error of cells sink to zero due to existence of many grid cells without a bounding box. Therefore, to deal with this problem, they increased the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects by using two parameters set λcoord= 5 and λnoobj= .5.

14. To decrease the impact of the difference between a large bounding box error and the small bounding box error, they worked with the square root of the width and height of the bounding boxes.

15. YOLO predicts multiple bounding boxes for per grid cell. Each grid has one predictor for a class based on the highest IOU.

16. In the following loss function, the classification error and bounding box coordinate errors are penalized. $\mathrm{l}_i^{\mathrm{obj}}$ means if the object appears in cell box i and $\mathrm{l}_{ij}^{\mathrm{obj}}$ means the j the bounding box predictor at cell i.

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{6}$$

17. YOLO was trained about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012 with a batch size of 64, a momentum of 0.9 and a decay of 0.0005. Our learning rate was 10^{-2} for 75 epochs, then 10^{-3} for 30 epochs, and finally 10^{-4} for 30 epochs.

1.3 Problems

1. One grid cell only can predict two boxes and can only have one class. Therefore it struggles with grouped small objects.
2. YOLO learns to predict bounding boxes from data, therefore it struggles to generalize to objects in unusual aspect ratios.
3. There is an earlier mentioned loss function problem for treating same to error in small bounding boxes and large bounding boxes. A small error in a small box has a much greater effect on IOU.
4. The authors state that the main source of error is incorrect localizations.

2 "Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017"—CVPR 2017, Best Paper Honorable Mention

2.1 What is there to improve? What is new? Why is it accepted?

1. YOLO has high localization error compared to Fast R-CNN, and low recall compared to region based methods. The authors aimed to fix these issues while maintaining the accuracy and speed. They aim to achieve these with YOLOv2.
2. They propose a real-time framework YOLO9000 for detecting of more than 9000 object categories by jointly optimizing detection and classification using hierarchical tree representation for class labeling.

2.2 How?

2.2.1 YOLOv2

1. Batch normalization was added on all of the convolutional layers in YOLO. This achieved more than 2% improvement in mAP and helped removing dropout from the model without overfitting.
2. The authors fine tuned the classification network at the 448×448 resolution for 10 epochs on ImageNet instead of 224×224 training input .Then they fine tuned the network on detection. This high resolution classification network achieved a 4% mAP increase.
3. The authors replaced the fully connected layers on top of the convolutional feature extractor with anchor boxes to predict bounding boxes. They eliminate one pooling layer to make the output higher resolution, and made the network to operate on 416 input images instead of 448×448 to have an odd number of locations in feature map. This was important because especially large objects tend to occupy the center of the image so they wanted a single location right at the center to predict these objects. In the end, YOLO's convolutional layers downsample the image by a factor of 32, and had a feature map of 13×13. Using anchor boxes decoupled the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box. The objectness prediction still predicts the IOU and conditional probability same

as YOLO. Without anchor boxes the intermediate model achieves 69.5 mAP with 81% recall, whereas anchor boxes achieves 69.2 mAP with 88% recall. Thus, this change helped recall.

4. The authors wanted to set good priors for setting dimensions for network to learn to predict good box sizes (therefore higher IOU score). Therefore, they used k-means clustering on the training set bounding boxes to find good priors. Since using standard k-means with Euclidean distance makes larger boxes generate more error than smaller boxes, instead they use

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid}) \tag{7}$$

5. Using anchor boxes also led to an instability of bounding box location prediction at the beginning. Therefore it had to be bounded. Thus, they predict location coordinates relative to the location of the grid cell as they did in YOLO. They made the groundtruth to fall between 0 and . We take a different approach, simply adding a passthrough layer that brings features from an earlier layer at 26×26 resolution.

6. One problem was detection of small objects. They added a pass through layer which concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, so that the $26 \times 26 \times 512$ feature map turns into a $13 \times 13 \times 2048$ feature map, and can be concatenated with the original features. This means that the detector will on this map. This change resulted in 1% performance increase.

7. The authors wanted YOLO to be robust to different image sizes and be able to predict well on diferent input sizes, so they change the network every few iterations. Every 10 batches the network randomly chooses new image dimensions of factor 32 (downsample factor) wiyth the smallest option 320×320 and the largest as 608×608. After resizing the training continues. YOLOv2 runs faster on smaller images, but has less accuracy. Thus, the user has a tradeoff between accuracy and speed while choosing an input size.

8. After these changes YOLOv2 has 78.6 mAP on VOC 2007 while operating above real-time speeds.

2.2.2 Faster Classification Model

1. The authors proposed a new classification model to be used as the base of YOLOv2 as it is seen in Fig 2 with 19 convolutional layers and 5 maxpooling layers to surpass YOLO and VGG-16 based models. Darknet requires 5.58 billion operations to process an image, and achieves 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet.

2.2.3 Jointly Training and Classifying

1. The authors a proposed a mechanism for jointly training on classification and detection data by using images labelled for detection to learn detection-specific information and how to classify common objects, which uses images with only class labels to expand the number of categories it can detect.

2. Images from both detection and classification datasets are mixed for training with the idea that if the network sees an image labelled for detection it can backpropagate based on the full YOLOv2 loss function or if it sees a classification image, it can only backpropagate loss from the classification related parts.

3. However, mixing different datasets turned out to be problematic due to the labelling style. Detection datasets have only common objects and general labels like "dog", whereas classification datasets have specific labels such as "Norfolk terrier". In order to solve this issue the authors proposed a hierarchical model.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	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		$2 \times 2/2$	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			

Figure 2: Darknet (from the paper)

4. ImageNet labels are pulled from WordNet, a language database in form of a directed graph that structures concepts and how they relate.
5. The authors wanted to simplify this graph problem by building a hierarchical tree named WordTree from the concepts in ImageNet. To achieve that they examined the visual nouns in ImageNet andtheir paths through the WordNet graph to the root node object. They allowed tree to branch as little as possible.
6. To perform classification with WordTree, conditional probabilities are predicted at every node. The authors shared an example as F

$$\begin{aligned} Pr(\text{Norfolk terrier}) = Pr(\text{Norfolk terrier}|\text{terrier}) \\ *Pr(\text{terrier}|\text{hunting dog}) \\ *\ldots* \\ *Pr(\text{mammal}|Pr(\text{animal}) \\ *Pr(\text{animal}|\text{physical object}) \end{aligned}$$

   and a representative image as Fig 3
7. Using this joint training, YOLO9000 learns to find objects in images using the detection data in COCO and, to classify objects using data from ImageNet. For detection, ImageNet shares only 44 object categories with COCO. In authors evaluation YOLO9000 achieved 19.7 mAP overall with 16.0 mAP on the 156 new object classes that it has never seen any labelled detection data for.

3 Conclusion

YOLO and its version are important to computer vision community for introducing and answering to new challenges regarding fast classification task and to robotics society for being a finally sufficiently accurate and fast detector for real-time tasks. I believe we will continue seeing YOLO's name cited in many future works. Currently it is pronounced often with Jetson Nano. I believe soon applications

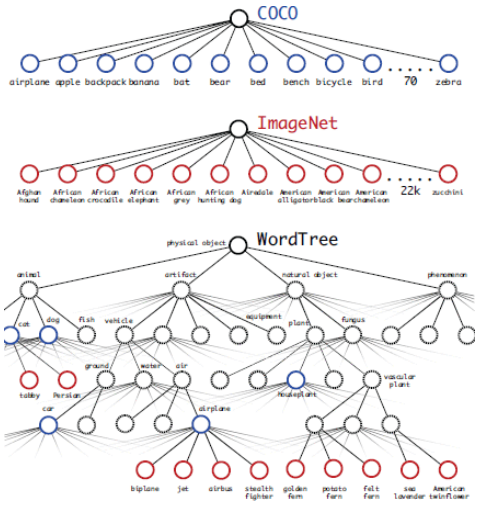

Figure 3: Combination (from the paper)

running on Jetson Nano will be using YOLO as a default detector as the speed is sufficient and the code is available.