Notes - You Only Look Once (YOLO)

Monday, December 24, 2018 6:24 PM

What is YOLO?
A new approach to object detection. It treats object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.

What are the previous approaches before YOLO?
The approaches used previously have made use of the classifiers to perform detection.

What would the structure of YOLO would look like?
A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. This network is optimized end-to-end directly on detection performance. The bigger YOLO can process images at the rate of 45 frames per second. The other network which is Fast YOLO can achieve a speed of 155 frames per second with the mAP more than double times the previous approaches. YOLO reframes the object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. This algorithm looks at the image only once and predict the objects classes and locations.

What is the analogy between the YOLO and the real life functioning?
Humans glance at an image once and instantly know what objects are in the image, where they are, and how they interact. The YOLO algorithm also tries to implement the same.

What are the significant previous approaches?
1. Deformable Parts Models (DPM)
2. R-CNN

What is Deformable Parts Models (DPM) ?
It is an algorithm that is used before YOLO and it uses the sliding window approach where the classifier is run at evenly spaced locations over the entire image.

What is R-CNN?
They use region proposal methods which first generate the potential bounding boxes in an image and then run a classifier on those proposed boxes. After the classification, post-processing is used to refine the bounding box, eliminate duplicate detections and rescore the box based on other objects in the scene.

What are the advantages of YOLO?
Fast and accurate:
YOLO doesn't have a complex pipeline. With no batch processing, base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. The video can be processed in real-time with less than 25 milliseconds of latency. YOLO can also achieve more than twice the mean average precision of other real-time systems.

Global inference:
Unlike sliding window and region proposal methods, YOLO reasons globally about the image when making predictions. YOLO sees an entire image during the training and test time so it encodes contextual information about the classes as well their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for object because it is not able to see the larger context.

Generalizable representations:
After training on natural images, YOLO can also work on the art-work and out performs detection methods like DPM and R-CNN by a wide margin.

How the unified detection works in YOLO?
YOLO combines the detection and classification of objects in an image in a single network and this network can be trained end-to-end. It also reasons globally about the full image and all objects in the image.
YOLO divides the input image into a S x S grid. If the center of an object falls into a particular grid then that grid is responsible for detecting the object.
Each grid cell predicts 'B' bounding boxes and confidence scores for those boxes (to reflect confidence about the box containing an object and also the accuracy of the box dimensions enclosing the object). The confidence can be defined as follows:

$$\Pr(\text{Object}) * \hat{\text{IOU}}_{\text{pred}}^{\text{truth}}$$

If there is no object in that cell, then it has to be zero, else the confidence score has to be equal to intersection over union (IOU) between the predicted box and the ground truth.
Each bounding box has to make 5 predictions: x, y, w, h and confidence. x, y coordinates represent the center of the box relative to the bounds of the grid cell. The width (w) and height (h) are predicted relative to the whole image. The confidence represents the IOU between the predicted box and the ground truth box.
Each grid cell also predicts C Conditional probabilities, $Pr(Class_i|Object)$. One set of class probabilities are predicted regardless of number of boxes B.
During the test time the conditional class probabilities is multiplied with individual box confidence predictions to give class-specific confidence scores for each box. This score gives both the probability of that class appearing in the box and how well the predicted box fits the object.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

The following figure shows the approach for the unified detection in detail.

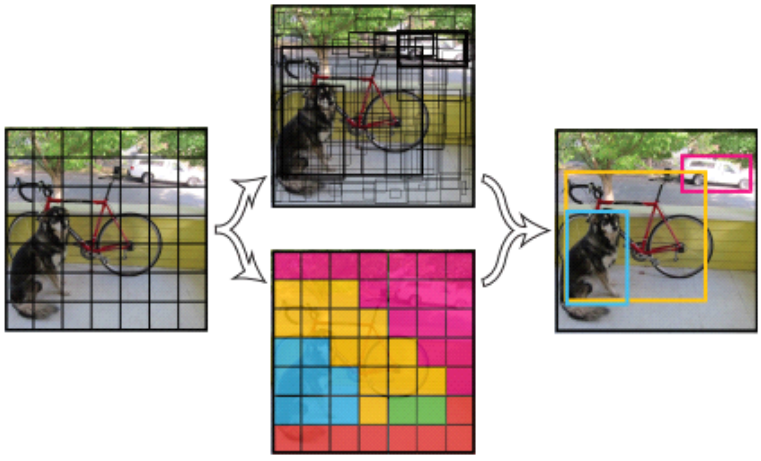

*Figure Description : The Model. Our system models detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities. These predictions are encoded as an S × S × (B * 5 + C) tensor.*

What is the design approach for unified detection in YOLO?
The unified detection model mentioned above was implemented using the convolutional neural network and evaluated on the PASCAL VOC detection dataset.
Initial convolutional layers extract features from the image while the fully connected layers predict the output probabilities and coordinates. The network architecture resembles the GoogLeNet model for image classification. The network has 24 convolutional layers followed by two fully connected layers. Instead of the inception modules used by GoogLeNet 1 x 1 reduction layers followed by 3 x 3 convolutional layers are used. The entire network is represented in the diagram below.

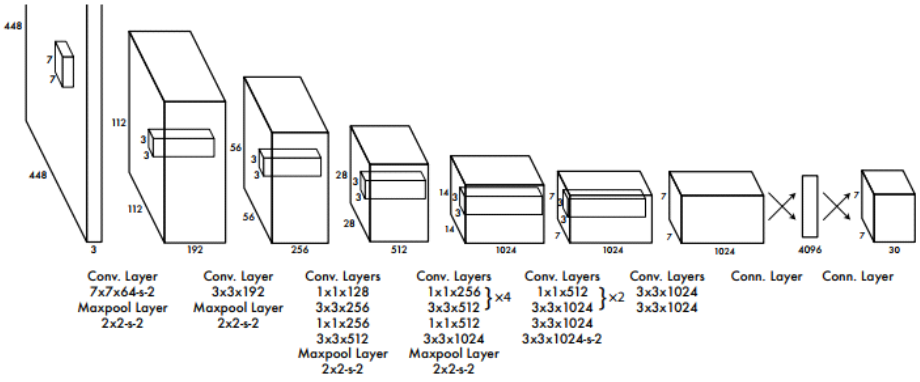

Figure Description: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

The paper also proposes a faster version of YOLO in which the number of convolutional layers are 9 instead of 24 and fewer filters in those layers. But all the training and testing parameters are same between both the versions.
For evaluating YOLO on PASCAL VOC Dataset following assumptions are considered for the network dimensions.
S = 7, B = 2, C = 20 (PASCAL VOC has 20 labelled classes). Hence the final prediction layer is a 7 × 7 × 30 tensor.

How the YOLO network is trained?
The convolutional layers are pretrained on the ImageNet 1000-class competition dataset. The network is trained for approximately a week and achieved a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, which is comparable to the GoogLeNet model. This pretrained network use the first 20 convolutional layers followed by an average pooling layer and a fully connected layer. Then this pretrained model is converted to perform detection (Because of the fact that adding convolutional layers and connected layers to pretrained models can improve performance). So four convolutional layers and two fully connected layers are added and their weights are initialized randomly. In order to provide the detection network with fine grained information, the input resolution of the network is increased from 224 x 224 to 448 x 448.
The final layer predicts both class probabilities and bounding box coordinates. The bounding box width and height are normalized by the width and height of the image so that they will be between 0 and 1. Then the x and y coordinates for the bounding box are considered to be offsets of a particular grid cell location and they will also take the value between 0 and 1.
Linear activation functions are used for the final layer and all other layers use the leaky rectified linear activations as mentioned below.

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

What is the optimization approach used and the error metric to apply the optimization?
The optimization will be for sum-squared error in the output of the model. *Sum-squared error is used because it is easy to optimize but it does not perfectly align with the goal of maximizing average precision*.
However this approach weighs the localization error equally with the classification error and it is not a good one. Also many grids in every image won't contain any objects and the confidence scores of these cells will be pushed towards zero, often overpowering the gradient from cells that do contain objects. This result in model instability and cause the training to diverge early on.
In order to avoid this, the loss from the boundary box coordinate predictions have to be increased and the loss from the confidence predictions for the boxes that do not contain objects have to be decreased.
As a result two parameter have to be used one is λ_{coord} and other is λ_{noobj}, then the values of them are set as λ_{coord}= 5 and λ_{noobj}=0.5.
Also the sum squared error weighs equally the errors in large and small boxes. But the error metric has to consider the fact that small deviations in large boxes matter less than the small deviations in the small boxes. In order to address this the square root of the bounding box width and height has to predicted instead of the width and height directly.
YOLO predicts multiple bounding boxes per grid cell. But during training only one bounding box predictor has to be responsible for the prediction. The particular predictor is selected based on the highest current IOU with the ground truth. It leads to the specialization between bounding box predictors. So each predictor gets better at predicting certain sizes, aspect ratios or classes of object, improving overall recall.

What is the loss function of the YOLO algorithm?
The following multi-part loss function has to be optimized.

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

The following expression denotes if the object is present in cell *i*.

$$\mathbb{1}_{i}^{\text{obj}}$$

The following expression denotes that *j*th bounding box predictor in cell *i* is responsible for the prediction.

$$\mathbb{1}_{ij}^{\text{obj}}$$

This loss function only penalizes the classification error if an object is present in that grid cell. It also only penalizes bounding box coordinate error if that predictor is responsible for the ground truth box (i.e. The highest IOU of any predictor in that cell).

What are the inferences that can be obtained from YOLO algorithm?

Same as training, the detection predictions for a test image only requires one network evaluation. On PASCAL VOC, the network predicts 98 bounding boxes and class probabilities of each box. YOLO is extremely fast since it requires a single network evaluation unlike classifier based methods.

The grid design provides spatial diversity in the bounding box predictions. So it is often clear which grid cell an object will fall into and the network only predicts one box for each object. In some cases some large objects or objects near the border of multiple cells can be localized by multiple cells. Non-maximal suppression is used to avoid these cases. This non-maximal suppression adds 2-3% in mAP.

What are the limitations of YOLO?

YOLO imposes strong spatial constraints on bounding box predictions, since each grid cell can only predict two boxes and can have only one class. This further limits the number of nearby objects that the model can predict. The model also struggles with the small objects that appear in groups such as flock of birds.

Since the model learns to predict from data, it struggles to generalize to objects in new or unusual aspect ratios and configurations. Relatively coarse features are used for the predictions since the architecture has multiple down sampling layers from the input image. When the algorithm is trained on a loss function that approximates detection performance, the errors are treated the same in small bounding boxes versus the large bounding boxes. Small error in the large boxes won't affect much but Small error in small box has a much greater effect on IOU, and this is the main source of incorrect localizations in YOLO algorithm.