DEEP LEARNING LAB(EEE4423-01) Week 7 - You Only Look Once: Unified, Real-Time Object Detection

Hyeon Si Eun[1], 2019142214
[1] Department of Electric and Electronic Engineering, Yonsei University

Introduction

The paper "You Only Look Once: Unified, Real-Time Object Detection"[4] presents a real-time object detection system that is both fast and accurate, by framing the task as a regression problem to predict bounding boxes and class probabilities directly from full images in a single forward pass of a convolutional neural network (CNN).

Existing detection models redefine classifiers and use them as detectors. Classification means looking at an image and determining whether it is a dog or a cat. However, object detection determines where the dog is located and where the cat is located within an image. Therefore, object detection needs to determine location information as well as classification. Existing object detection models include DPM[2] and R-CNN[1].

Deformable Parts Models (DPM) are models that perform object detection through a sliding window method across the entire image. R-CNN uses a method called region proposals to generate bounding boxes in images. Classification is performed by applying a classifier to the proposed bounding box. After classification, they do post-processing to adjust the bounding boxes, remove redundant detections, and rescore the boxes based on the objects. Because of this complexity, R-CNN is slow. It is also difficult to optimize because each procedure must be trained independently.

Thus, the authors improved the procedure by viewing object detection as a single regression problem. This is a regression problem that redefined the series of steps from image pixels to bounding box locations (coordinates) and class probabilities. With this system, YOLO quickly finds what objects are in an image and where they are in a single pipeline. It is named YOLO (you only look once) because you can detect an object by looking at the image only once.

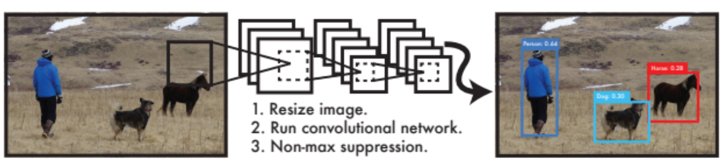

Figure 1: Processing images with YOLO is simple and straightforward.

YOLO is simple. Let's look at Figure 1. A single convolutional network computes multiple bounding boxes and their class probabilities simultaneously. YOLO optimizes detection performance immediately by learning the entire image. This unified model of YOLO has several advantages over traditional object detection models.

First, YOLO is incredibly fast. Because YOLO turns the traditional complex object detection process into a single regression problem. This eliminates the need for complex pipelines like traditional object detection models. You can easily detect objects by simply feeding new images to the YOLO neural network during the test phase. YOLO's base network processes 45 frames per second without batch processing on the Titan X GPU. The fast version of YOLO (Fast YOLO) processes 150 frames per second. This means that the video can be processed in real time. (It can be processed with a latency of less than 25 milliseconds) Moreover, YOLO has more than twice the mean average precision (mAP) of other real-time object detection models.

Second, YOLO looks at the entire image when making predictions. Unlike sliding window or region proposal methods, YOLO looks at the entire image during training and testing phases. Thus, it learns and processes not only information about the shape of the class, but also information about its surroundings. On the other hand, Fast R-CNN, which is the best performing object detection model prior to YOLO, cannot process surrounding information. So, if there is a speckle or noise in the background without any object, it recognizes it as an object. This is called background error. Because YOLO processes the entire image, the background error is much smaller than that of Fast R-CNN. (approximately 1/2)

Third, YOLO learns the general parts of an object. Because it learns the general part, when it learns natural images and tests them with pictorial images, YOLO's performance is far superior to DPM or R-CNN. Therefore, compared to other models, YOLO is more robust to new images not seen in the training phase. This means higher detection accuracy.

However, YOLO has the disadvantage of slightly lower accuracy compared to state-of-the-art (SOTA) object detection models. It has the advantage of being able to detect objects quickly, but the accuracy is somewhat lower. Detection accuracy is poor, especially for small objects. Speed and accuracy are trade-offs. All of YOLO's code is open source, and pretrained models are also available for download.

Unified Detection

YOLO is a model that integrates the individual elements of object detection into a single neural network. YOLO uses features across images to predict each bounding box. This design of YOLO enables end-to-end learning and real-time object detection while maintaining high accuracy. YOLO divides the input images into an S x S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those bounding boxes. The confidence score indicates how confident we are that the bounding box contains the object, and how accurate the predicted bounding box is. The confidence score is defined as $\Pr(\text{object}) * IOU_{pred}^{truth}$.

IOU (Intersection Over Union) is the ratio of the intersection area to the union area of the actual and predicted bounding boxes of an object. That is, IOU = (intersection of actual bounding box and predicted bounding box) / (union of actual bounding box and predicted bounding box). If there is no object in the grid cell, Pr(Obejct)=0. Therefore, the confidence score is also 0. If we predict that there is definitely an object in the grid cell, Pr(Object)= 1. Therefore, if the confidence score equals the IOU, it is the most ideal score. Each bounding box consists of 5 predictions. These are x, y, w, h, and confidence. The (x, y) coordinate pair represents the relative position within the grid cell of the center of the bouning box. It is not an absolute position, but a relative position within a grid cell, so it has a value between 0 and 1. If (x, y), the center of the bounding box, is exactly at the center of the grid cell, then (x, y)=(0.5, 0.5). The (w, h) pair represents the relative width and relative height of the bounding box. At this time, (w, h) indicates the width and height of the bounding box as relative values when the width and height of the entire image are 1. Therefore, (w, h) also has a value between 0 and 1. Finally, confidence is the same as the confidence score discussed earlier. And each grid cell predicts C(conditional class probabilities), $\Pr(\text{Class}_i|\text{Object})$. This is the conditional probability of what class an object is, given that the object is inside a grid cell. Regardless of how many bounding boxes are in the grid cell, only one class probability value is obtained for one grid cell. We said that one grid cell predicts B bounding boxes. Regardless of the number of B's, only one class is predicted in one grid cell. In the test phase, the conditional class probability (C) is multiplied by the confidence score of each bounding box, which is called the class-specific confidence score for each bounding box. The class-specific confidence score can be calculated as follows.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * IOU_{pred}^{truth} = \Pr(\text{Class}_i) * IOU_{pred}^{truth} \tag{1}$$

These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

YOLO system model treats object detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These

This is an extended abstract. The full paper is available at the this paper predictions are encoded as an S × S × (B * 5 + C) tensor. The YOLO researchers experimented with the Pascal VOC, an image recognition international competition dataset. They set S=7, B=2, and Pascal VOC has a total of 20 labeled classes, so C=20. If S=7 then the input image is divided into a 7 x 7 grid. B=2 means we want to predict 2 bounding boxes in one grid cell. When we do this, we create an S x S x (B * 5 + C) tensor. So the dimension of the final prediction tensor is (7 x 7 x 30).

Network Design

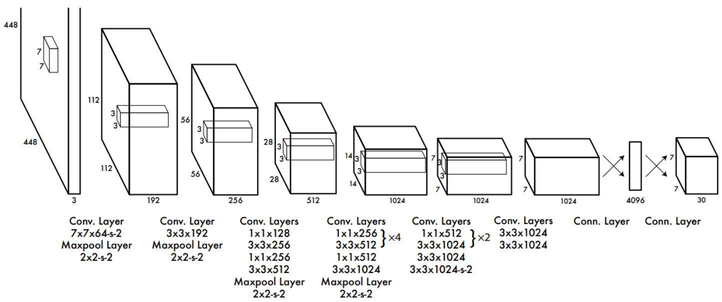

Figure 2: The Architecture

As mentioned earlier, this YOLO model is designed as a CNN structure. YOLO researchers performed modeling on the Pascal VOC dataset. The front end of this CNN is a convolutional layer, followed by a fully-connected layer. A convolutional layer extracts features from an image, and a fully connected layer predicts class probabilities and bounding box coordinates. YOLO's neural network structure is derived from GoogLeNet, which is used for image classification. YOLO consists of a total of 24 convolutional layers and 2 fully connected layers. Instead of GoogLeNet's inception structure, YOLO uses a combination of 1 x 1 reduction layers and 3 x 3 convolutional layers. The overall structure of the YOLO model is shown in Figure 3. The final output of this model is 7 x 7 x 30 prediction tensors. There is also Fast YOLO which uses fewer convolutional layers (9 instead of 24) and filters than YOLO for faster object recognition. It differs only in size, and all other parameters used for training and testing are the same as YOLO.

Training

First, we pretrained YOLO's convolutional layer with the ImageNet dataset with 1,000 classes. This pretrained model recorded an accuracy of 88 percents on the ImageNet 2012 validation dataset. The YOLO researchers used the Darknet framework[3] for all this training and inference.

ImageNet is a dataset for classification. Therefore, we need to replace the pretrained classification model with an object detection model. After the 20 pretrained convolutional layers, the researchers improved the performance by adding 4 convolutional layers and 2 fully combined layers. When adding 4 convolutional layers and 2 fully combined layers, we initialized the weights of these layers randomly. In addition, the resolution of image information must be high for object detection. Therefore, we increased the resolution of the input image from 224 x 224 to 448 x 448. A linear activation function was applied to the last layer of the YOLO network, and leaky ReLU was applied to all other layers. In ReLU, all values below 0 are 0, whereas in leaky ReLU, even values below 0 have small negative values.

YOLO's loss is based on sum-squared error (SSE). So they have to optimize the sum-squared error (SSE) of the final output. The reason they used SSE is because SSE is easy to optimize. however it does not perfectly align with our goal of maximizing average precision. Losses in YOLO include localization loss, which is how well you predicted the location of a bounding box, and classification loss, which is how well you predicted a class. It is not a good idea to train with the same weight for localization loss and classification loss. However, the way they optimize SSE treats the weights of these two losses equally. There's another problem, most of the grid cells in the image don't have objects. This is because the background area is larger than the foreground area. The confidence score is 0 if there are no objects in the grid cell. Therefore, we have no choice but to learn so that the confidence socre of most grid cells is 0. This causes model imbalance.

To improve this, they increased the weight of loss for bounding box coordinates where objects exist, and decreased the weight for confidence loss of bounding boxes where objects do not exist. It means that the weight of the localization loss is increased among the two losses (localization loss, classification loss), and the weight of the confidence loss of grid cells with objects is increased rather than the confidence loss of grid cells without objects. They used two parameters for this, λ_{coord}=5 and λ_{noobj}=0.5.

SSE has another problem. SSE computes the loss with equal weights for both large and small bounding boxes. However, a small bounding box is more sensitive to small position changes than a large bounding box. A bounding box enclosing a large object still wraps the large object well when moved slightly, but a bounding box enclosing a small object escapes the small object when moved slightly. To improve this, they take the square root of the bounding box's width and height. This is because taking the square root for width and height reduces the increase rate as width and height increase, which has the effect of reducing the weight for loss.

YOLO predicts multiple bounding boxes per grid cell. In the training phase, one bounding box predictor should be responsible for one object. That is, each object must be matched with one bounding box. Therefore, only one of the multiple bounding boxes must be selected. For this, among several predicted bounding boxes, the one with the largest IOU with the ground-truth bounding box that encloses the real object is selected. The largest IOU with the ground-truth bounding box equates to the best wrapping of the object. A bounding box predictor trained in this way is good at predicting specific sizes, aspect ratios, and classes of objects.

The loss function used in the training phase is as follows:

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

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

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

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

Inference

Similar to training, network evaluation is only required once for detection on the test image. YOLO requires only a single network evaluation, so the test time is very fast. The grid design applies spatial deiversity to predict the Bbox. Only one box is predicted in which grid cell the object is in. However, some large objects or objects near the boundaries of multiple cells are localized by multiple cells.

Limitations of YOLO

YOLO predicts two bounding boxes per grid cell. And only one object can be detected per grid cell. This causes spatial constraints. Spatial restriction means 'one grid cell detects only one object, so if two or more objects are attached to one grid cell, it cannot be detected well'. One grid cell detects only one object, but if there are many objects, there are some objects that cannot be detected. And since the YOLO model learns to predict bounding boxes from data, it will struggle when it encounters a new aspect ratio that it didn't learn during training. Finally, the YOLO model has the disadvantage of equally weighting losses in large and small bounding boxes. A small change in the position of a large bounding box has relatively little effect on performance, whereas a small change in the position of a small bounding box can have a large effect on performance. This is because the small bounding box has a more severe IOU change with position change compared to the large bounding box. YOLO's main source of error is incorrect localizations.

Comparison to Other Detection Systems and Experiments

The authors conducted several comparative experiments with other detection systems. The authors demonstrated the effectiveness of YOLO on the PASCAL VOC and COCO datasets, achieving state-of-the-art results in terms of both speed and accuracy. YOLO was able to process images at a speed of 45 frames per second (fps) on a Titan X GPU, making it suitable for real-time applications. If you want to see detailed experimental results, we recommend viewing the full paper.

Conclusion

YOLO introduced a new approach to object detection that combines high accuracy and real-time performance. YOLO also detects objects well for new images that were not seen in the training phase. YOLO makes it a widely adopted algorithm in computer vision applications.

[1] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446, 2015.

[3] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.