Exploring You Only Look Once (YOLO) Model for Object Detection

Weixiao Cheng
Department of Civil and Environmental Engineering
University of Pittsburgh
Pittsburgh, USA
cheng_wx@pitt.edu

Mimi Xie
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, USA
mm.xie@pitt.edu

***Abstract*—Deep convolutional neural networks (CNN) appear to be a promising technology to be employed in object detection tasks. With deep CNN, the accuracy of classification has been greatly improved. R-CNN and YOLO are the two of the most famous object detection models based on deep CNN. Compared with R-CNN, YOLO model can support real-time object detection systems. In this project, we explore the power of deep neural networks for the problem of object detection. Specifically, we explore how to implement YOLO model for object detection. We design our own code for the preprocessing the input images. We feed the pre-trained weights into a Keras model. After this, we post-process the detection results that include a large amount of bounding boxes to find the most important object detection results. The experimental results show that YOLO model outperforms most existing object detection models in terms of fast inference. The prediction accuracy is about 70% (if the intersection over union between prediction and ground truth is larger than 0.5 and the classification is correct, then the prediction is correct), which is acceptable.**

***Index Terms*—CNN, YOLO, Object detection**

I. Introduction

Object detection is the process of finding objects such as faces, buildings, and vehicles in images or videos. Given an input image, an object detector is able to output a bounding box where one or multiple objects of interest exist. This technique has many existing applications such as face detection, counting objects, and image analysis, which can be further integrated into mobile devices, vehicles, and military equipments. In a word, this technique can significantly improve people's living quality and promote the development of technology. For human, object detection is simple. However, for a computer, an image is just a three dimensional array of numbers between 0 and 255. Therefore, it is really hard for a computer to directly know whether there is an object or in which region this object locates. Besides, the unknown number of objects and different forms and sizes that an object appears, increases the difficulty of object detection task for a computer.

Traditional object detection techniques usually take a long time and cannot be used for runtime applications. Object detection has three steps: first, selecting the candidate regions that may have an object; second, extracting the features from these regions; third, classifying objects with classifiers. Traditional object detectors employ a sliding window traverse the whole image in the first step. Since an object with different size and form may appear in any region of an image, different length and width need to be covered to consider all possible objects. As a result, this kind of exhaustive technique has an obvious drawback: the high time complexity. Because of too many windows, the following two steps for object detection are heavily influenced. To reduce the time complexity, some work propose to limit the number of sliding windows and only explore the windows that have a fixed length-width ratio . However, an object cannot be accurately located if its length-width is not included in the set sliding windows.

The advance of convolutional neural networks (CNN) brings new opportunity to the object detection field which are currently widely employed in the state-of-the-art object detectors. CNN takes advantages of the image structure. Unlike a regular neural network, the neurons of a CNN model are arranged in three dimensions including the height, width, and depth. The depth is with respect to the three channels of an image, RGB. CNN has a special structure which usually consists of three layers: convolutional layer, fully-connected layer, and pooling layer. As the most important layer, convolutional layer occupies more than 90% of the overall computation. This layer iteratively do element-wise multiplication between the filter (weights) and a region of the input feature map aiming for feature extraction. The pooling layer is usually connected after the convolutional layer to reduce the spatial dimensions and thus the computation overhead. Besides, it also helps control overfitting. The final fully-connected layer is used to calculate the classification score with respect to every possible class.

In this project, to evaluate the object detector, we train a large number of images with ground truth annotations in the form of bounding boxes (rectangles around the object of interest). Therefore, we use the images and annotations from the PASCAL VOC dataset [3]. The dataset is divided into training set and testing set. The training set is used to train the object detector while the testing set is used to test the performance.

II. Methods

We explored the performance of different convolutional neural network (CNN) based object detection algorithms. CNN or its variants are the most popular network architecture for image classification because of its higher performance and efficiency compared to fully connected neural network model [9]. The most important two kinds are R-CNN and YOLO based models. We compared these two models and found that R-CNN is still based on the transformation of sliding window for object localization. The sliding window is a conceptually simple but powerful algorithm [8], it can be considered as a rectangular region with a certain height and width that slides across an image. For each window, CNN model will be used to recognize whether it contains the object of interest. However, it takes a long time to apply the CNN model for each bounding box. Compared with this model, YOLO can support run-time object detection with high performance.

A. Basic Idea

As the name implied, YOLO model [1] only looks at the whole image once and makes predictions of the object classes and the corresponding bounding boxes simultaneously. Specifically, the image first will be divided into $S \times S$ grid, if the center of the object lies in a grid, then that grid will be used to detect the bounding box of the object. for each grid, the model will predict 5 anchor boxes, each anchor box containing a vector of 85 values, i.e., the multiplication of probability of containing an object and the intersection over union (IOU) between ground truth and predicted bounding box, the box parameters, and the probability of each class, as shown in Fig. 1. The right image in Fig. 1 shows concrete examples of one anchor box in each grid cell.

B. Model Architecture

To transform the original image into $S \times S$ grid (S = 13 for the YOLO model cited in this paper), a deep CNN was employed. The model architecture is shown in Fig 2. For image of arbitrary size (i.e., width height 3), it was first preprocessed into shape of $416 \times 416 \times 3$, which is the required for the YOLO model input. After a deep CNN (Table 1), the original image pixels were encoded to bounding box parameters and class probabilities information that are included in a matrix of shape $13 \times 13 \times 5 \times 85$. Finally, the loss function was defined so that the model can be trained using stochastic gradient descent.

C. Loss Function

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$
$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$
$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2$$
$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2$$
$$+\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

```
Layer (type)                    Output Shape
================================================
input_1 (InputLayer)            (416, 416, 3)
conv2d_1 (Conv2D)               (416, 416, 32)
max_pooling2d_1 (MaxPooling2D)  (208, 208, 32)
conv2d_2 (Conv2D)               (208, 208, 64)
max_pooling2d_2 (MaxPooling2D)  (104, 104, 64)
conv2d_3 (Conv2D)               (104, 104, 128
conv2d_4 (Conv2D)               (104, 104, 64)
conv2d_5 (Conv2D)               (104, 104, 128
max_pooling2d_3 (MaxPooling2D)  (52, 52, 128)
conv2d_6 (Conv2D)               (52, 52, 256)
conv2d_7 (Conv2D)               (52, 52, 128)
conv2d_8 (Conv2D)               (52, 52, 256)
max_pooling2d_4 (MaxPooling2D)  (26, 26, 256)
conv2d_9 (Conv2D)               (26, 26, 512)
conv2d_10 (Conv2D)              (26, 26, 256)
conv2d_11 (Conv2D)              (26, 26, 512)
conv2d_12 (Conv2D)              (26, 26, 256)
conv2d_13 (Conv2D)              (26, 26, 512)
max_pooling2d_5 (MaxPooling2D)  (13, 13, 512)
conv2d_14 (Conv2D)              (13, 13, 1024)
conv2d_15 (Conv2D)              (13, 13, 512)
conv2d_16 (Conv2D)              (13, 13, 1024)
conv2d_17 (Conv2D)              (13, 13, 512)
conv2d_18 (Conv2D)              (13, 13, 1024)
conv2d_19 (Conv2D)              (13, 13, 1024)
conv2d_20 (Conv2D)              (13, 13, 1024)
conv2d_21 (Conv2D)              (26, 26, 64)
conv2d_22 (Conv2D)              (13, 13, 1024)
conv2d_23 (Conv2D)              (13, 13, 425)
================================================
```

TABLE I: CNN architecture for YOLO model

The sum of square error loss function was used and shown above. 1_{ij}^{obj} means that if there is an object in i_{th} grid cell (S2) and jth bounding box (B), its value is 1, otherwise, its value is 0; the same meaning applies for l_{ij}^{noobj} except that when there is no object its value is 1, otherwise, its value is 0. λ means that how much we want to penalize the corresponding error term. As in the original paper [1], $\lambda_{coord} = 5$ indicates that we want the bounding box parameters (i.e., center (x, y), width and height) to be more accurate, while $\lambda_{noobj} = 0.5$ means we care less about the loss for boxes containing no objects. In the second error term, the square root operation on width and height means that we want to penalize more for small boxes than large boxes. C_i represents the multiplication of probability of containing an object and the IOU between ground truth and predicted bounding box for i_{th} grid cell; and $p_i(c)$ is the probability of each class in i_{th} grid cell.

D. Post-processing

As seen in Fig. 3, each grid cell will predict 5 bounding box vectors, which result in many overlap boxes for one image sample. To achieve the clean results in the right-most image (Fig. 3), non-max suppression technique was employed. Specifically, the confidence of each bounding box (i.e., P(Object)$*$ IOU (truth, prediction)) was calculated and filtered through a threshold (e.g., 0.4). Then for the filtered boxes, the one with highest confidence score was selected; then the IOU was calculated between the selected box and all the other boxes. If the IOU is larger than a threshold (e.g., 0.5), then that box will be removed. The above process is iterated until the expected number of bounding boxes are reached. The non-max suppression procedure can be easily implemented with build-in function non_max_suppression in TensorFlow.

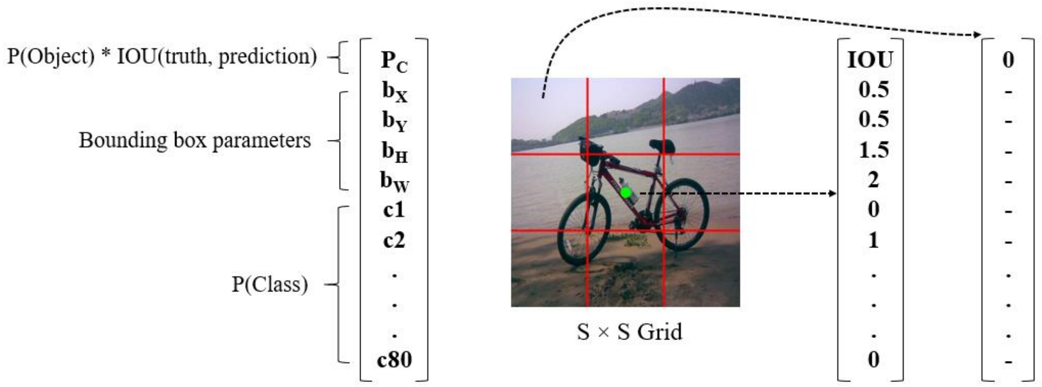

Fig. 1: Illustration of YOLO model

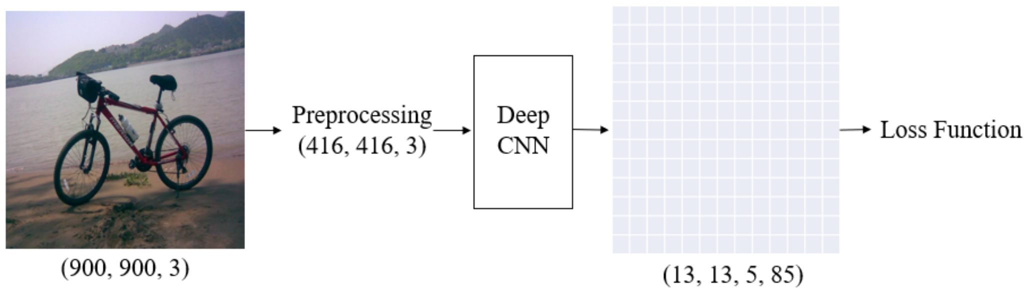

Fig. 2: YOLO model architecture

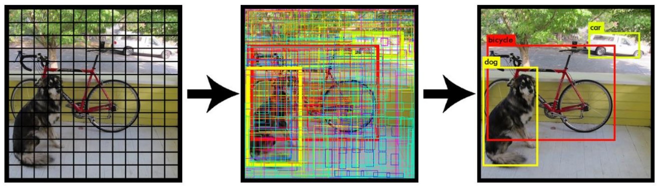

Fig. 3: YOLO model workflow from Redmon et al.s paper [1]

E. Dataset

The YOLO model was trained using Microsoft Common Objects in Context (COCO) dataset [2], which is a large dataset that contains 80 object categories and over 1.5 million labeled object instances (with bounding box and segmentation labeling). The 80 object classes are provided in coco_classes.txt file in model_data folder. Compared to other datasets such as ImageNet [3], PASCAL VOC 2012 [4], and SUN [5], the images in COCO have richer contextual information, which make it more suitable for real-world application.

III. Results and Discussions

A. Pre-trained Weights

Because the training of YOLO model is extremely expensive, we employed a pre-trained model for this project. The pre-trained weights (i.e., yolov2.weights) were downloaded from the YOLO website (https://pjreddie.com/darknet/yolov2/). The original YOLO paper did not provide detailed description of the model architecture, to reduce debug process, we used the YAD2K (Yet Another Darknet to Keras, https://github.com/allanzelener/YAD2K) tool to convert the pre-trained weights to a Keras model. The Keras model was then stored in yolov2.h5 file in model_data folder.

B. Inference Validation

Although we used the pre-trained model directly to simplify our training process, applying the model to do inference task is still very challenging. Especially, post-processing the CNN model output is not trivial. According to the YOLO paper [6], for the output of CNN (Fig. 2), the first five values in each bounding box vector (corresponding to, t_x, t_y, t_w, and t_h in Fig. 4) are first applied a either sigmoid or exponential function so that their values lie in (0, 1) or larger than 1. Then these values were added to coordinates of the grid cell (c_x, c_y) or multiplied by the anchor box parameters (p_w, p_h, both were obtained using k-mean algorithm based on the COCO training dataset). The above process will convert the CNN model output layer to bounding box parameters that are suitable for further filtering and non-max suppression. Finally, the predicted bounding boxes were parsed and drawn with Python Image Library (Fig. 5). For inference validation, we selected the image used in the original YOLO paper (Fig. 3) as an example to perform the inference task. As shown in Fig. 5, we achieved the same results compared with the original paper [1].

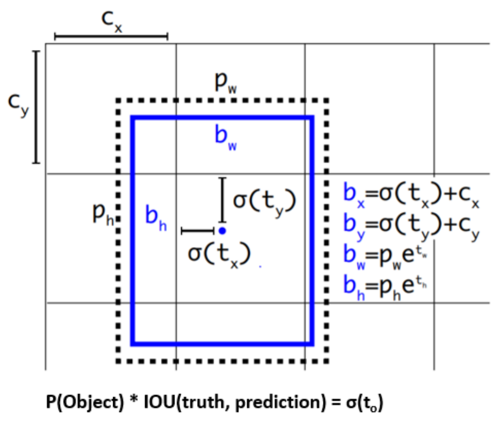

Fig. 4: Bounding box parameters [6]

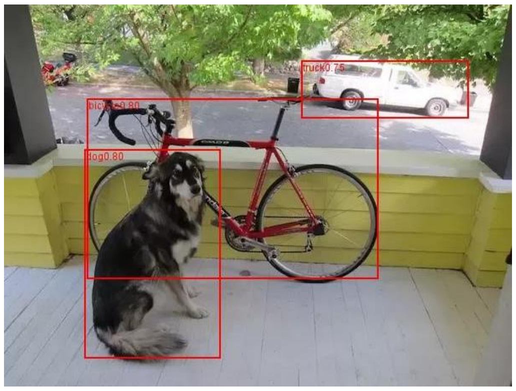

Fig. 5: Inference validation example

C. Car Detection

We used the YOLO model for some real-world application C car detection to test its performance. Above are the detection results, all pictures were taken on the street in Pittsburgh and all the results are saved in images folder. The prediction accuracy is estimated to be 70% (21 / 30), based on the definition that if a predicted box has IOU with ground truth larger than 0.5 and the predicted class is correct, then the prediction is considered as correct. As shown in Fig. 6, when the object present in the picture is small, it is hard for YOLO to make accurate predictions. For example, in the first picture, bicycles were misclassified into bench, and in other picture, some cars and buses far away from the camera were even not detected by the model. In addition, in some cases (e.g., the traffic light in Fig. 6 (2)), the predicted bounding box have a low IOU (less than 0.5) with the ground truth. Although those results show that the prediction accuracy is relatively low, those results are generally acceptable, especially given the fast run time.

D. Model Limitation

Although YOLO model is very fast (40 frame per second), it has some accuracy issue, especially when compared to some other algorithms such as faster R-CNN [7]. The mean average precision (mAP) of YOLO for COCO training dataset is only around 48%, and this was calculated based on the definition

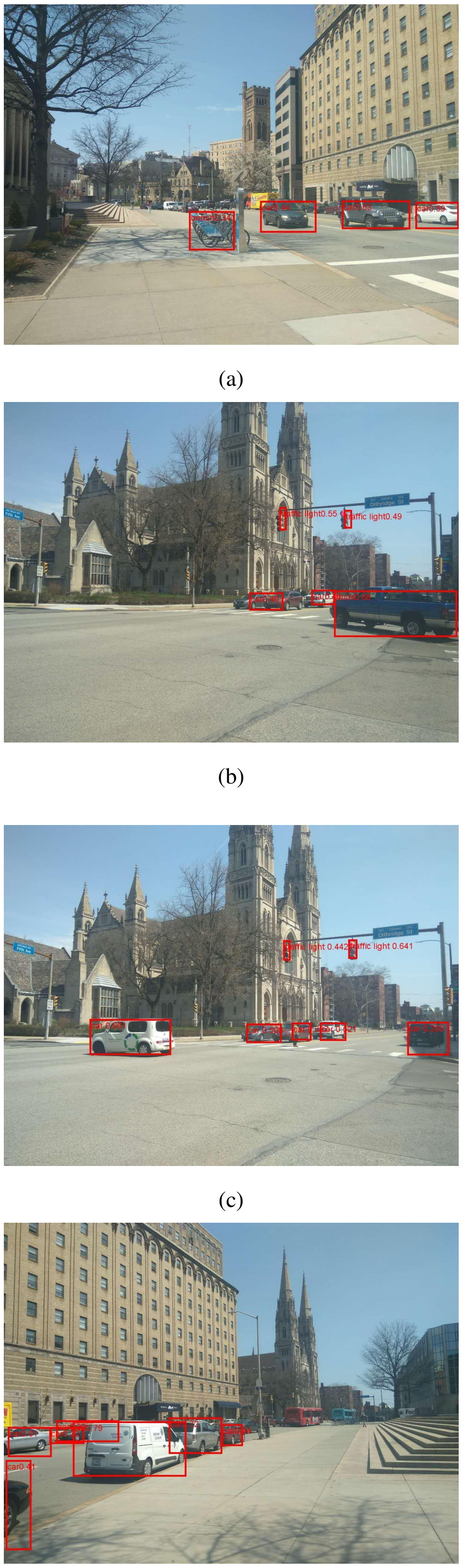

Fig. 6: Car detection results

that if a predicted box has IOU with ground truth larger than 0.5, then the prediction is considered as true positive. The reason for this, as seen from our car detection results, could be that the YOLO predicts the bounding box from the training dataset directly, so it has problems with generalizing to unseen object instances (e.g., the bicycles in Fig. 6).

IV. Conclusion

In this project, we explored CNN based detection models and how to implement YOLO model for object detection. YOLO is a very fast algorithm that can be used for real-time task. Although the prediction accuracy is relatively low, the results are still acceptable in general. In the future, we may want to try to use faster R-CNN on the same test dataset and make a comparison between those two state-of-the-art real-time algorithms.

V. Team Member Contribution

Both two team members did the research of CNN based object detection technology and studied different CNN models especially for object detection. Besides, both two members work on the final report with equal contribution. Specifically, Weixiao is responsible for the implementing the YOLO model , evaluating the results, and post-processing the resulting bounding boxes. Mimi is responsible for proposing the idea of using CNN for object detection, analyzing different CNN models used for this project, and coding the part of preprocessing the images.

Both authors reviewed the manuscript before submission.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ”You only look once: Unified, real-time object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779-788.

[2] T.-Y. Lin et al., “Microsoft coco: Common objects in context,” in European conference on computer vision, 2014, pp. 740-755: Springer.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ”Imagenet: A large-scale hierarchical image database,” in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 2009, pp. 248-255: IEEE.

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, ”The pascal visual object classes (voc) challenge,” International journal of computer vision, vol. 88, no. 2, pp. 303-338, 2010.

[5] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, ”Sun database: Large-scale scene recognition from abbey to zoo,” in Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, 2010, pp. 3485-3492: IEEE.

[6] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” arXiv preprint, 2017.

[7] S. Ren, K. He, R. Girshick, and J. Sun, ”Faster r-cnn: Towards real-time object detection with region proposal networks,” in Advances in neural information processing systems, 2015, pp. 91-99.

[8] Rowley, Henry A., Shumeet Baluja, and Takeo Kanade. ”Neural network-based face detection.” IEEE Transactions on pattern analysis and machine intelligence 20, no. 1 (1998): 23-38.

[9] LeCun, Yann, Leon Bottou, Yoshua Bengio, and Patrick Haffner. ”Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86, no. 11 (1998): 2278-2324.