Object Detection with YOLO on Artwork Dataset

Yihui He*
Computer Science Department, Xi'an Jiaotong University
heyihui@stu.xjtu.edu.cn†

Abstract

I design a small object detection network, which is simplified from YOLO(You Only Look Once[15]) network. YOLO is a fast and elegant network that can extract meta features, predict bounding boxes and assign scores to bounding boxes. Compared with RCNN, it doesn't have complex pipline, which is easier for me to implement. Start from a ImageNet pretrained model, I train my YOLO on PASCAL VOC2007 training dataset. And validate my YOLO on PASCAL VOC2007 validation dataset. Finally, I evaluate my YOLO on an artwork dataset(Picasso dataset). With the best parameters, I got 40% precision and 35% recall.

1. Introduction & Related Work

For object detection task in the past, Exhaustive search [6, 13, 17], segmentation based approaches [5, 7], and Region Convolutional Network based approaches[11, 10, 16] have been widely used.

Exhaustive search approachs attempts to search for all object proposals in an image. The major drawback is computationally expensive. And thus requires constraints to prune the search space and reduce the number of considered locations, at the cost of completeness. In [8], Felzenszwalb *et al*. performed an exhaustive search using HOG features and a linear SVM. This resulted in excellent object detection performance. To better guide the exhaustive search, in [14], Lampert *et al*. used an appearance model, combined with a branch and bound technique, to remove scale, aspect ratio, and grid constraints from the previous exhaustive search formulation. However, their method still requires evaluation on over 100,000 proposals [2].

Segmentation approaches attempts to construct object proposals from the underlying pixels. In [12], the authors segment objects based on a parts model. Each part is described by appearance and shape features. In [3], Arbelaez *et al*. use a contour detector which has been shown to provide excellent results. Although segmentation methods perform well, they are often computationally expensive.

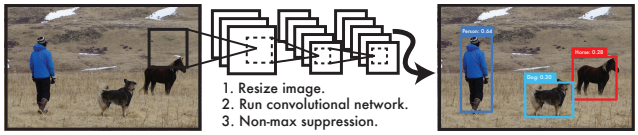

Figure 1: **The YOLO Detection System.** (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Region Convolutional Network approaches attempt to use additional neural network to propose bounding boxes. Girshick *et al*. use convolutional network features to train a pipeline consisting of a classifier and regression step, called R-CNN [11]. R-CNN, fast R-CNN and faster R-CNN are faster and more accurate compared with previous methods. However, their training pipline is complex and still not efficient enough.

Single Convolutional Network approach, namely YOLO, is a fast and accurate detection system. They frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

In this work, I implement and investigate YOLO.

2. Feature Extraction & Classification Scheme in One Network: YOLO

In this section, I'll briefly introduce YOLO algoithm that I employed. For more details, you can read the original paper[15]. The pipline of standard YOLO is show in figure1.

*Yihui is a CS sophomore, with Xi'an Jiaotong University. This work is his course project for 2016 Spring CS281B Advanced Computer Vision taught by Prof. Yuan-Fang Wang, when he was an international exchange student at University of California, Santa Barbara.

†My UCSB email is yihui@umail.ucsb.edu. Perm Number: X289793

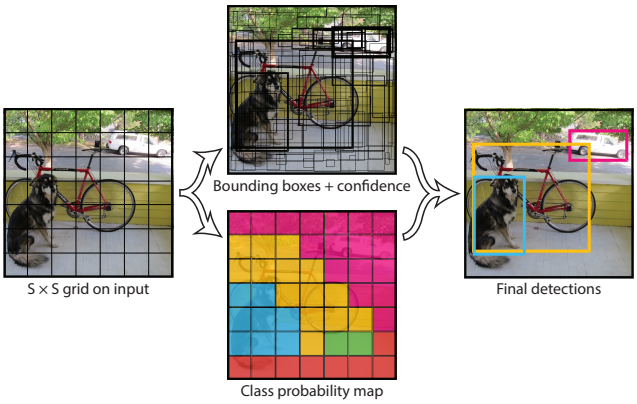

Figure 2: **The Model.** It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

YOLO unify the separate components of object detection into a single neural network. It uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously.

First, YOLO divides the input image into an $S \times S$ 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 boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally, confidence is $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) represent the center of the box.(w, h) represents size of a bounding box. Confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$.

At test time, the conditional class probabilities times the box confidence gives class-specific confidence scores for each box, which encode both the probability of that class appearing in the box and how well the predicted box fits the object.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

In practice, I use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Network output is a $7 \times 7 \times 30$ tensor. The architecture of YOLO is similar with ordinary convolutional neural network(Like AlexNet), only difference is object function. So I'll not discuss the architecture in details, which is shown in figure3.

3. Technical Details

In this section, I'll discuss technical details in my project.

3.1. Features

The convolutional neural layers before output are used for generate features. Since the original YOLO described in previous section is a large network. I design a smaller network for this project, which is similar with YOLO. My architecture is shown below(FC: Full-connected, Conv:convolution, pool:max-pooling, Drop:Dropout)

```
Warp image > 448 x 448 x 3 image
Conv 448x448x3 image, 16 filters
pool 448x448x16 image, 2 size, 2 stride
Conv 224x224x16 image, 32 filters
pool 224x224x32 image, 2 size, 2 stride
Conv 112x112x32 image, 64 filters
pool 112x112x64 image, 2 size, 2 stride
Conv 56x56x64 image, 128 filters
pool 56x56x128 image, 2 size, 2 stride
Conv 28x28x128 image, 256 filters
pool 28x28x256 image, 2 size, 2 stride
Conv 14x14x256 image, 512 filters
pool 14x14x512 image, 2 size, 2 stride
Conv 7x7x512 image, 1024 filters
Conv 7x7x1024 image, 1024 filters
Conv 7x7x1024 image, 1024 filters
FC 50176 inputs, 256 outputs
FC 256 inputs, 4096 outputs
Drop 4096 inputs, 0.5 probability
FC 4096 inputs, 1470 outputs
```

There's no explicit features in YOLO,since features are extracted by neural network automatically. Actually, the last convolutional layer can be regard as feature map(7x7x1024).

3.2. Classification Scheme

As I've described in the previous section, YOLO is a unified neural network for multi-objects detection. Specifically, YOLO converts detection task into a regression problem. In this regression problem, the position of bounding boxes and confidence for each class are our regression object Y. The raw image is X.

3.3. Dataset Selection & Partition

I select two small dataset: PASCAL VOC2007(400MB, 9000 images) and Picasso Artwork Dataset[9](218 paintings).

Pretraining For objects detection tasks, researchers usually initalize their network with pretrained models on ImageNet. I initalize my neural network with caffe pretrained

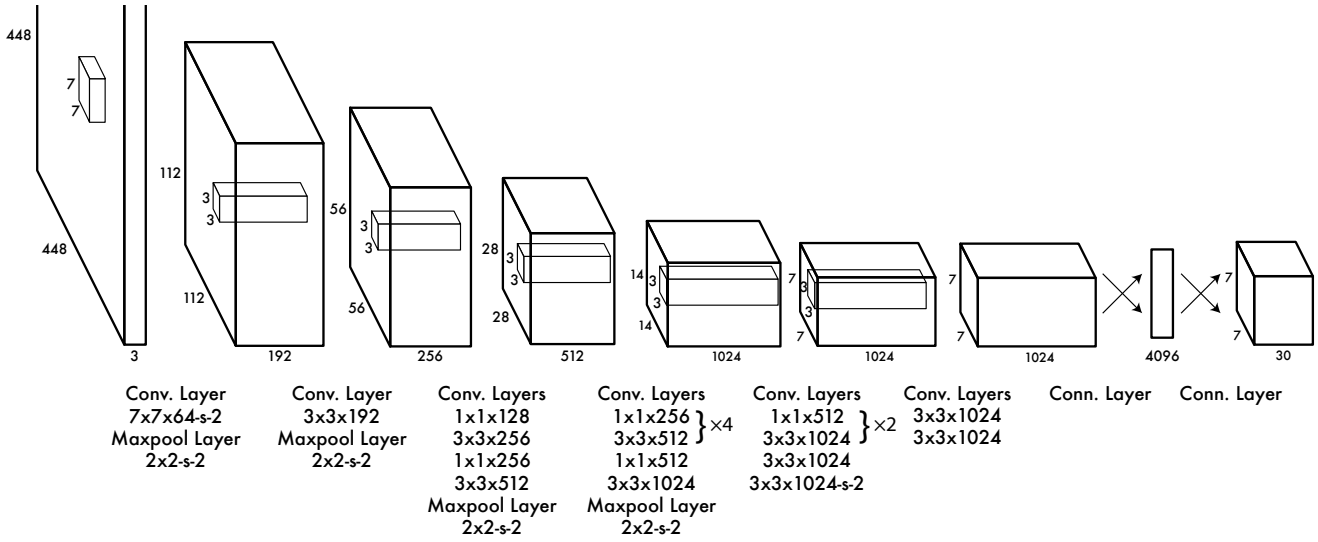

Figure 3: **The Architecture.** The convolutional layers are pretrained on the ImageNet classification task at half the resolution (224 224 input image) and then double the resolution for detection. Note that since this network is too large, I design a simplified network which is described in section 3.1.

model[1]. So that for training part, I only need to perform fine-tuning on PASCAL VOC2007.

Training & Validation Since PASCAL VOC2007 is not large(9000 images) and it is also a commonly used dataset for object detection tasks, I choose it for training and testing. I use 6000 images from PASCAL VOC2007 as training dataset, and the rest 3000 images from PASCAL VOC2007 as validation dataset.

Testing I choose Picasso Artwork Dataset as testing dataset for 3 reasons. (i) Artwork dataset is interesting and new for me. (ii) There's many algoithms tested on it. (iii) Since artwork style is different from real world, it can tell whether a classifier have good generalization to other domains.

3.4. Cross-Validation

I've mentioned in previous subsection that I have 6000 images for training and 3000 images for testing. So I performed a 3-fold cross-validation. One also needed to mentioned is that, I also performed exponential moving average, which also helps improving performance and preventing overfitting. This is a special technique for neural network. It is formalized as follow:

$$W_t = \alpha \cdot W_t + (1 - \alpha) \cdot W_{t-1}$$

Here, W is the weights of my model, $\alpha$is the exponential decay rate. So that the final model is an ensemble of previous iterations. I'll show in experiments section that these two techniques both help improve performance.

[1]https://github.com/BVLC/caffe/wiki/Model-Zoo

4. Experiments

I implement my YOLO with TensorFlow[1] in python. Training, validation, and testing are performed on my laptop Thinkpad T440p.

4.1. Training & Cross-Validation

First, I perform 3-fold cross-validation on PASCAL VOC2007 as described in previous section. For each training time, I also perform exponential moving average weights. To show the effect of cross-validation, I compare 3 standalone models with the combined model. It turns out that the combined model have **3%** mAP improvement, which have 43% mAP.

To show the effect of exponential moving average weights, I train a model without exponential moving average weights as control group. Then I compare it with the model above. It turns out that Without exponential moving average weights, mAP drops **1%**.

After training, I compared my model with other public algoithms on Pascal VOC2007, which is shown in table1.

It seems obvious that my YOLO implementation has lower mAP compared with the original paper. That makes sense because my implementation has fewer layers and fewer filters compared with the original paper. As for frames per second, since the benchmark results got by other algorithms are tested on GPU. My result is from my laptop CPU. So it is not comparable. However, you can still see that the FPS is higher that Fast R-CNN. We can safely claim that YOLO has a good trade-off between speed and performance compared with other state-of-the-art algorithms.

Real-Time Detectors	Train	mAP	FPS
30Hz DPM[18]	2007	26.1	30
YOLO	2007+2012	63.4	**45**
Fast R-CNN[10]	2007+2012	70.0	0.5
Faster R-CNN[16]	2007+2012	**73.2**	7
Mine(simple YOLO)	2007	43	0.77(CPU)

Table 1: **Detection systems comparision PASCAL VOC.** Comparing the performance and speed of detectors. YOLO have good trade-off between speed and performance. That's also a reason that I employ YOLO in this project.

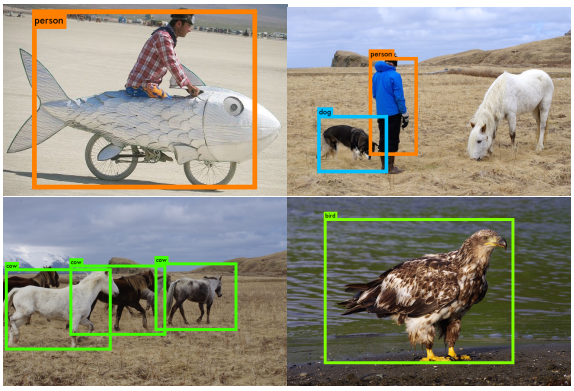

Figure 4: Some interesting results in real world

I include some interesting detection results in figure4. Some are successful, some failed.

4.2. Testing

After training on PASCAL VOC2007, I start testing on Picasso Artwork Dataset. Since it is a small dataset with only 218 images, it only takes me 3 minites to go through the full dataset, even on my CPU. Some successful and failed examples are shown in figure5.

4.2.1 Tuning Threshold: Precision Recall Trade-off

Threshold of object bounding boxes confidence is the most important parameter for me to tune. I've mentioned in second section that confidence threshold is defined as IoU(intersection over union) between predicted box and the ground truth. a confidence threshold of 50% means that we'll accept proposals that believes their bounding boxes have more that 50% overlap with real object. Increase confidence threshold will lead to less bounding box proposals for each image. Decrease confidence threshold will result in more bounding boxes. It's like the threshold for SIFT, which controls number of feature points. I've tested 5 value for confidence threshold(100%, 75%, 50%, 25%, 0%) and compare my result with other public algoithms, which is

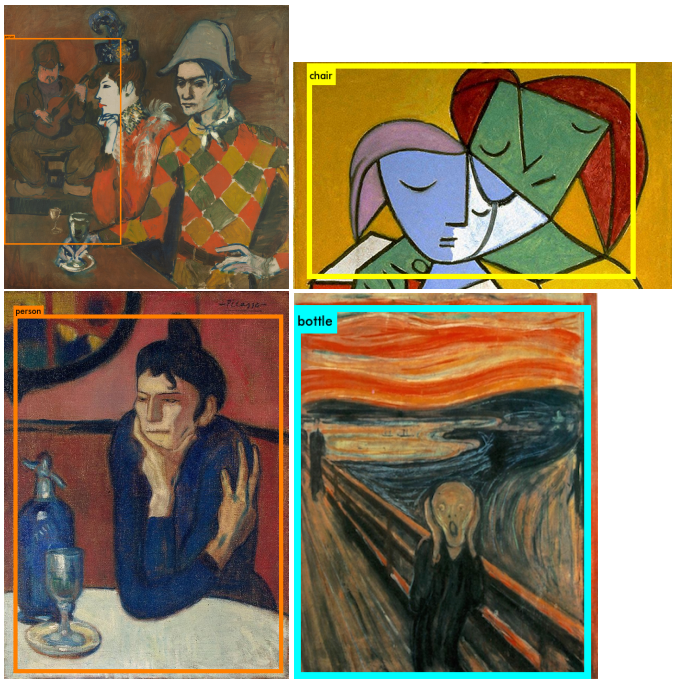

Figure 5: The two on the left are successful examples. The two on the right are failed examples.

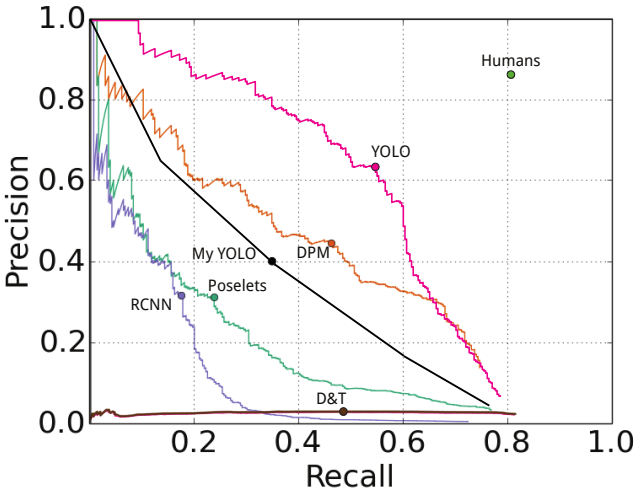

Figure 6: Picasso Dataset precision-recall curves

shown in figure6. Human perfomance on Picasso Artwork Dataset is the dot in right up corner.

4.2.2 Comparision with Other Models

After considering the trade-off between precision and recall, I select confidence threshold of 50% as my final model, which has precision of 40% and recall of 35% . I compared my YOLO with other algoithms in table2.

	AP	Best F1
YOLO	53.3	0.590
R-CNN	10.4	0.225
DPM	37.8	0.458
Poselets[4]	17.8	0.271
D&T[6]	1.9	0.051
Mine	34	0.373

Table 2: AP and F˙1 score evaluation of different algorithms on Picasso Dataset

From this table, you can easily see that, (i) current state-of-the-art is still far away from human perfromance, which also means that current algoithms do not have good generalization on non-real image. (ii) YOLO have a better generalization than other algoithms. (iii) The result I got is reasonable, since my model is a simplified YOLO.

5. Conclusion & Lessons Learned

I learned a lot from through this project. Here I draw some conclusions and show some lessons I learned.

- I extensively read papers on objects detection tasks. Get the general idea of exhaustive search approaches, segmentation approaches and convolutional neural network approach.
- Important parameters like threshold need to be tuned very carefully. Precision and recall trade-off need to be decide.
- Cross-Validation is a good way for most models to further improve their accuracy. However, need some time to implement myself, since there's no cross-validation module in most machine learning packages.
- Objects detection is a computation expensive problem. It takes long time to train, validate and test, if I want to get good perfomance.
- Though many computer vision tasks like image classification have surpass human performance. However, there's still a lot of work to be done on objects detection task.

Finally, thanks to this project, it helps me gain a deep and broad view on objects detection task!

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*. 3

[2] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010. 1

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011. 1

[4] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009. 5

[5] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*. IEEE, 2010. 1

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 5

[7] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*. 2010. 1

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 1

[9] S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In *Computer Vision-ECCV 2014 Workshops*, pages 101–116. Springer, 2014. 2

[10] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1, 4

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[12] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *CVPR*, 2009. 1

[13] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *CVPR*. IEEE, 2009. 1

[14] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 2009. 1

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 1

[16] 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*, pages 91–99, 2015. 1, 4

[17] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004. 1

[18] J. Yan, Z. Lei, L. Wen, and S. Li. The fastest deformable part model for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2504, 2014. 4