Real Time Object Detection and Logo Detection with You Only Look Once

Deepika Konreddy, Eric Jiang, Harshitha Krishna, Priscilla Roy Imandi, Teja Salapu
University of Houston, COSC 6373: Computer Vision, Spring 2018

Abstract— **Since the application of convolutional neural networks to object detection systems, improvements have been made to the speed and computational cost of object detection. In this project, we focus on the effectiveness of You Only Look Once (YOLO) to detect a variety of object classes in real-time using pre-trained classes from the Pascal VOC Dataset and comparing against custom-trained classes. In addition, we demonstrate the ability for YOLO to learn from logo images from the Flickr-Logo 27 Dataset and detect logos in real-time. Boasting an accuracy of over *90%* in both testing conditions, we validate YOLO's claim on being an accurate real-time object detection.**
.

1.INTRODUCTION

The human visual system can detect real time objects at a glance, by visually understanding features and objects from light input and converting them into ideas. Vision requires recognizing objects presented in a wide range of orientations and accurately interpreting geospatial cues.

Computer vision aims to simulate human perception, by not only processing a voluminous amount of image data but further identifying specific objects in a given environment. A key problem in computer vision consists of classifying an image into one of many categories and finding the location of a single object within an image, the combination of which outlines the topic of object detection.

Developing real time object detection techniques has a significant impact in real world applications. Object detection techniques are widely used in security systems to identify threat objects from surveillance video, and in medical applications to detect tumors from CT scan images [1]. In this project, we are interested in a multi-class object detection system as well as logo detection. The ability for a computer vision system to detect objects of various classes in real time is a critical element of autonomous driving in detecting obstacles and making decisions accordingly. We will analyze the performance of the YOLO detector under different training datasets of multiple classes. In addition, we will train YOLO to detect logos, which has a range of applications from contextual ad placement to vehicle logo for traffic-control systems.

2.RELATED WORK

The classical approaches to object detection involved using keypoint-based detectors and descriptors, HAAR, Local Feature-Based Recognition, bag of words, etc. However, since the ImageNet Large-Scale Visual Recognition Challenge in 2012, deep learning has been the de facto approach for object detection, significantly improving upon the performance of the aforementioned classical approaches. The three main methods currently used for object detection are Regional Convoluted Neural Networks (R-CNN), Single Shot MultiBox Detector (SSD), and You Only Look Once (YOLO).

The first application of R-CNN by R. Girshick, et al. [2] aimed to identify where the main objects are in the image by using selective search to generate proposals for bounding boxes, extracting CNN features from each bounding box for classification, and running the box through a linear regression model to tighten coordinates of the box. Future iterations of R-CNN include Fast R-CNN, Faster R-CNN, and Mask-CNN. *Improvements were made in the processing speed and accuracy by sharing the forward passes of CNN across subregions as well as joining the CNN, classifier and bounding box regressor into a single model.*

SSD by W. Liu, et al. [3] offered a higher performing object detection method using sets of auxiliary convolutional layers along with a class-agnostic bounding box proposal system. Requiring an input image with ground truth boxes assigned, the convolutional network produces a fixed-size collection of bounding boxes and scores for the presence of object class instances.

3.PROBLEM

Compared to other regional proposal classification networks, which perform detection on various regions and make multiple predictions per region, rendering a complex pipeline too expensive and slow for real time object detection, YOLO unifies the recognition tasks by simultaneously predicting multiple bounding boxes and class probabilities for those boxes. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and

test time so it implicitly encodes contextual class and appearance information. Furthermore, YOLO learns a generalizable representation of objects, enabling it to be applied to new domains and with unexpected inputs. Overall, the YOLO architecture enables end-to-end training and real time speeds while maintaining high average precision.

Therefore, we have selected YOLO by Red J. Redmon et al. [4] as the model for implementing our multi-class detection system.

4..METHODOLOGY

4.1 Yolo Implementation

4.1.1 Creation of Dataset

Selected the classes which we want the algorithm to detect and then collected images of the classes which we want to detect.We labelled the images, by drawing the bounding boxes around the interested classes. This can be done using [8].This would generate an Annotation file(xml) which contains details regarding the image and the bounding boxes within the image.

4.1.2 Random Weight Initialization

Random weights are initialized. This file is in the form of <filename>.weights which is taken from [9]. We updated labels.txt with the names of all the "interested classes".Updated the configuration file with the value of filter [5*(5 + Number of Classes)] and the number of classes.

We start the training the model using the following command :

In CPU:

```
flow --model cfg/tiny-yolo-voc-20c.cfg --load bin/tiny-yolo-voc.weights --train --annotation train/Annotations --dataset train/Images
```

Here -- model is the configuration file, -- weights are the initial random weights, --annotation denotes the xml files – dataset would indicate the Images.

In GPU:

We just add an additional parameter –gpu which indicates the fraction of GPU being used.

For every few iterations, checkpoints are saved, which can be used to verify the training process.

4.2 Loss Function

YOLO's loss function must simultaneously solve the object detection and object classification tasks. This function simultaneously penalizes incorrect object detections as well as considers what the best possible classification would be. We employ the stochastic gradient descent optimization method offered by TensorFlow[6] with the Adam optimizer [7] to minimize the cost function.

For VOC Dataset and Logo Dataset we got the following loss function values.

Metric	Logo Dataset	VOC Dataset
Loss Value	0.9	3.01

Figure 1. Loss Function Value

5..DATASET AND EVALUATION METRICS

5.1 Dataset

5.1.1 Real time Object Detection

For real time object detection, we have selected images from PASCAL VOC 2007[10] database to comprise our detection system. It consists of images as well as annotations for the task of object detection.The images correspond to 20 classes of Person, Bird, Cat, Cow, Dog, horse, sheep, aeroplane etc. The results of this system are tested on real world instances of images and real time video.

5.1.2 Logo Detection

For the task of Logo Detection, we use the FlickrLogos-27 [5] dataset. It consists of real-world images collected from Flickr depicting company logos in various circumstances. The annotations were in the form of a text file hence we generated/created the ground truth xml files for the training dataset and testing dataset.

5.2 Evaluation Metrics

We have used three evaluation metrics namely Accuracy, Precision, Recall.

5.2.1 Accuracy

Accuracy = (correctly predicted class / total testing class) × 100%.The accuracy can be defined as the percentage of correctly classified instances (TP + TN)/(TP + TN + FP + FN). where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives, respectively.

5.2.2 Precision

Precision is the fraction of relevant instances among the retrieved instances. Precision = TP/TP+FP

5.2.3 Recall

Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

Recall = TP/TP+FN.

5.2.4 F-Statistic

When measuring how well you're doing, it's often useful to have a single number to describe your performance. We could define that number to be, for instance, the mean of your precision and your recall. This is exactly what the F statistic is:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

6. EXPERIMENTAL RESULTS

YOLO is a fast, accurate object detector, making it ideal for computer vision applications. We have trained our model on two custom datasets. The results are as follows:

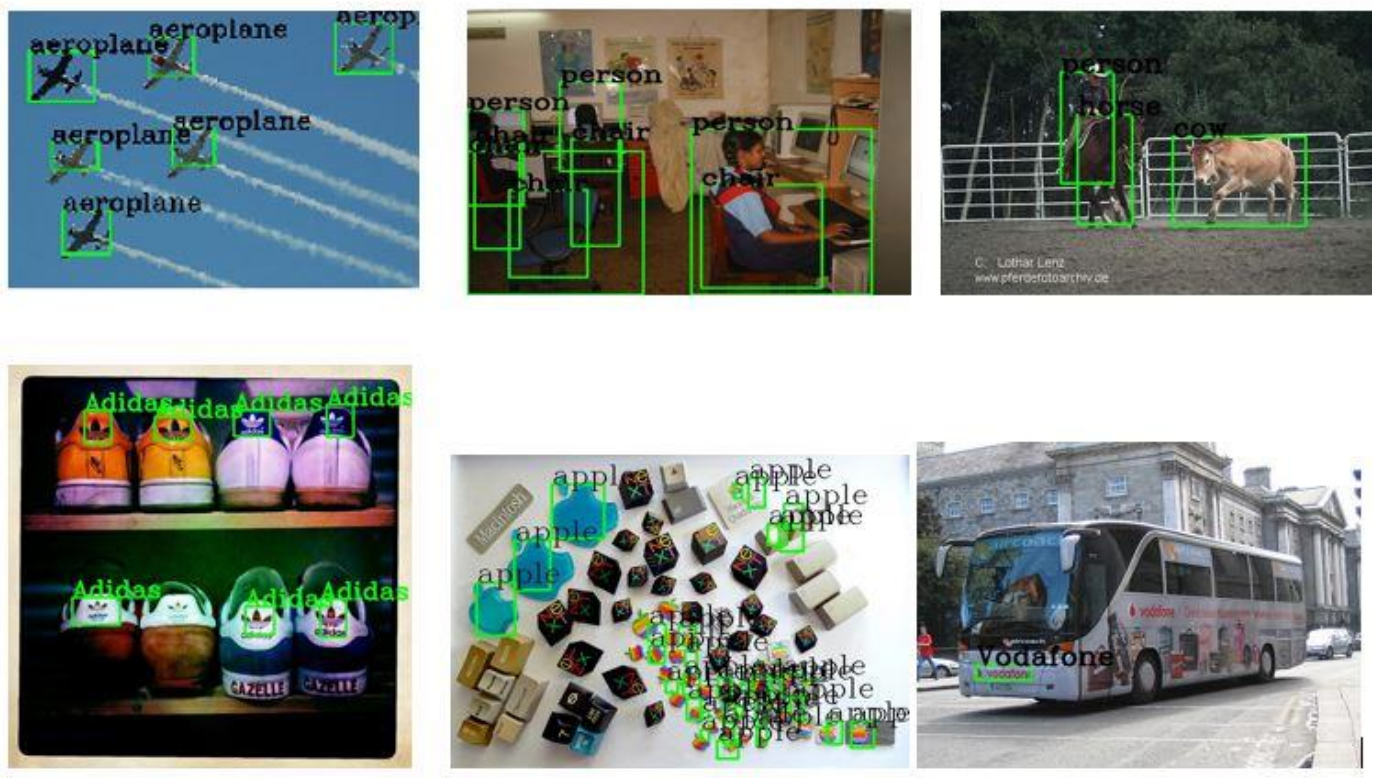

Figure 2: Qualitative Results. YOLO running on sample images of logo and natural images. It is evident that the detections are good except for few images where they have been detected with wrong labels.

Image	TP	TN	FN	FP	Accuracy	Precision	Recall	F- Measure
1	7	1	0	0	87.5	100	100	100
2	1	1	0	0	50	100	100	100
3	1	1	0	0	50	100	100	100
4	10	7	0	8	40	55	100	70.96
5	4	1	0	0	80	100	100	100
6	7	4	0	1	58.3	87.5	100	93.3
7	2	1	0	0	66.6	100	100	100
8	4	1	0	0	80	100	100	100
9	1	1	0	3	20	25	100	40
10	2	4	0	1	28.5	66.6	100	79.95

Table 1: YOLO EVALUATION METRIC ANALYSIS. The evaluation metrics for all the images tested on the real time object detection and logo detection model.

Image(1-10) included in Appendix.

6.1 Real time Object Detection

Operating System: Ubuntu 16.04v
RAM: 30GB
CPU's: 8
GPU: 8GB
Network : PaperSpace

We performed object detection on a video at 67 frames/second. During the detection, every frame is treated as an image and the model performs detection on each frame.

7.LIMITATIONS OF YOLO

Each grid cell in YOLO predicts two boxes and can have only one unique class. This is a spatial constraint on bounding box predictions which makes the model struggle to detect objects which are very small and appear in groups.

8.CONCLUSION AND FUTURE WORK

We trained and tested YOLO's real-time detection framework on multiple object categories by jointly optimizing detection and classification. We found that YOLO indeed offers a significant balance between speed and accuracy. By creating our custom datasets in both the multiple class learning and specifically the logo detection, we subsequently demonstrated YOLO's capability to be applied to any object and situation.

The results from our project demonstrate that custom detection systems can be effectively trained using the YOLO algorithm for many domains. While there is an abundance of image data, labeled data can be expensive. Future improvements can be made to improve the training using weak labels or even semi-supervised learning methods to increase the robustness of YOLO.

.

APPENDIX:

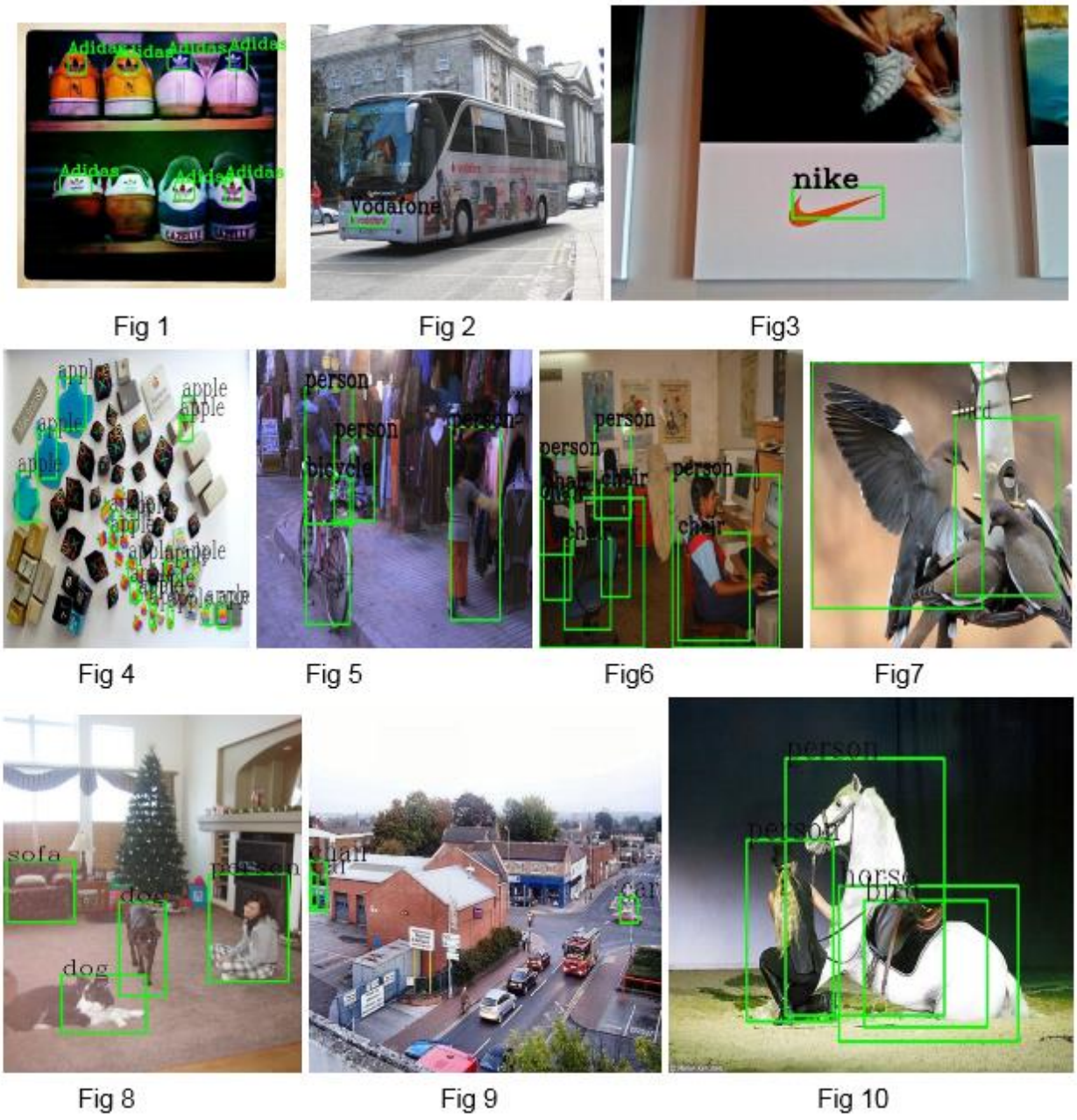

Figure 3: Images used for testing the evaluation metrics.

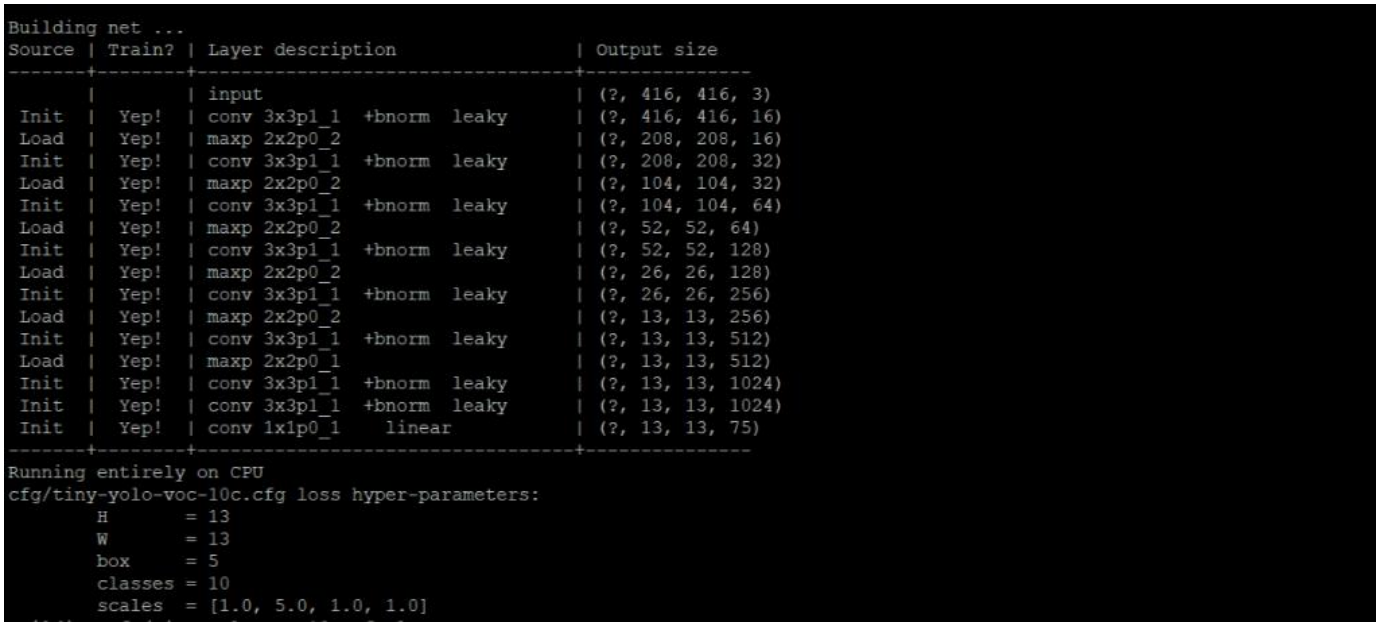

Figure 4: Training in GPU

REFERENCES

[1] D. Mandal et al., "Human visual system inspired object detection and recognition", Technologies for Practical Robot Applications (TePRA) 2012 IEEE International Conference on, pp. 145-150, 2012.

[2] Ross B. Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik:
Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 38(1): 142-158 (2016)

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, "SSD: Single shot multibox detector", 2015.

[4] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. arXiv preprint arXiv:1612.08242, 2016. 2

[5]Flickr-Logo27dataset-
http://image.ntua.gr/iva/datasets/flickr_logos/

[6]Mart´ın Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software availablefromtensorflow.org.2015. URL:http://
tensorflow.org/.

[7] LiweiWang,YanZhang,andJufuFeng.“OntheEuclidean distance of images”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 27.8 (2005), pp. 1334–1339.

[9]https://github.com/leetenki/YOLOtiny_v2_chainer/blob/master/tiny-yolo-voc.weights

[10]http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar

[8] https://github.com/tzutalin/labelImg