# CAP6610 Project Report: Supervised object detection with YOLOv3

Yuhao Shi
shiyuhao@ufl.edu
CAP6610 - Machine Learning
University of Florida

April 24, 2022

**Abstract**

The main focus of this project is to implement a Convolutonal Neural Ntework (CNN) machine learning model that detects the presence, location, and class of one or more objects given an image. These neural network algorithms performs two tasks, which is to classify the objects of the image and locate the identified objects by plotting a box around them. This paper aims to view object detection as a supervised problem by using a pre-trained model on labeled data and YOLOv3 algorithm

## 1 Introduction

Object detection is a task in computer vision technnique that allows computers to detect, identify, and locate visual objects of certain classes within a bounding box from a given image or video. Combined with identification and localization, object detection can be used to accruately count objects and track their location. A human can be able to recognize a cat sitting on bed, or a laptop placed on desk, the goal of object detection is also to achieve this kind of intelligence by solving one of the fundamental problems in computer vision: What objects are Where?

The problem consists of three methods and steps in order to achieve the goal of object detection, which include object classification (what categories are the objects belong to), object localization (where are the objects located at), and object recognition (determine the presence of objects). Below shows a general example of the result of object detection:

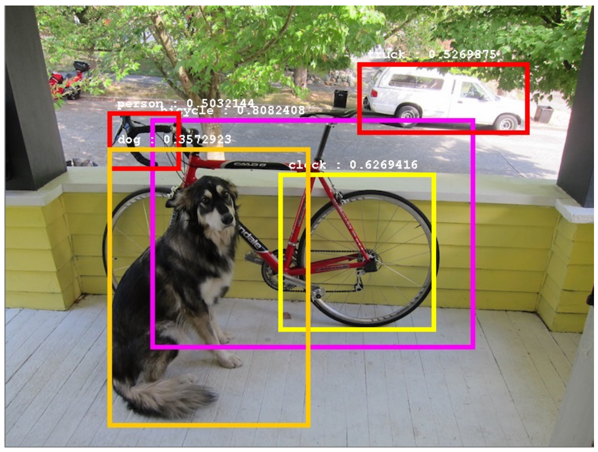


*Figure 1 shows an example of how objects are recoginzed and located. (Image taken from blog by Alexey Zinoviev).*

Object detection is a supervised problem, which the models are trained on labeled examples and data, and each trainning dataset needs to be accompanied with additional data that stores classes and boundaries of the objects within the images. This approach allows to train models with high accuracy. For the CNN algiorithm (YOLOv3) that I will be using in this project, a pre-trained model and a pre-trained model weight needs to be used in order to implement and perform object detection. These models and datasets are open-sourced tools that were given by YOLOv3 developers

### 1.1 Motivation

The main motivation for choosing object detection as my final project topic of this class is because I want to gain practical experiences in the field of computer vision combined with machine learning. There are numerous use cases of object detections being used in a wide range of industries and their products, some well-known examples include security cameras for tracking, face recognition, and object detections employed by the systems of self-driving cars and robots. Some of the concepts and topics that I have learned over the course such as classifications, model optimization, and regularization have helped me when researching the field of object detection. This project and topic provides me with practical experimentations and experiences of machine learning being used in the research domain and industry.

## 2 Problem Statement

The object detection problem is implemented using convolution neural network for classifications. Locating object's spatial location is resolved by using YOLOv3 algorithm. A pre-trained data model trained on COCO datasets along with its pre-trained weight data was provided by the developers of this algorithm. The model was trained on labeled datasets, and it's needed in order to perform object detection in this project (furthur discussed in following sections). YOLO algorithm employs convolutional neural networks to detect objects in real time, which only require a single forward propagation via a neural network. This suggests that the prediction can be done in one run, and class probabilities and bounding boxes can be predicted in the same time. Therefore, the problem can be formulated based on convolution opreation and forward propagation.

Forward propagation is to calculate and store intermediate variables from input layer to output layer. The convolution output from input layer $a$ is convolved with weight $W$, where $W$ is the weight parameter of the hidden layer, and intermediate variable $z$ is calculated for current output layer by:

$$z = Wa$$

The activation function $\phi$ for all layers is a linear function where $\phi(x) = x$ if the $x$ coordinates of bounding box is greater than 0, and $\phi(x) = 0.1x$ iif otherwise. Then the intermediate variable is run through the activation function $\phi$, and we can get the hidden activation vector of length $h$:

$$h = \phi * z$$

$h$ ia also the intermediate variable, then assume another weight parameter $W^1$, we can get the next output layer variable $y$:

$$y = W^1 * h$$

The loss term $L$ for a single data example can be calculated with loss function $l$ and example label $e$ :

$$L = l(y, e)$$

The loss function of the YOLO algorithm is given by its developers, where the algorithm learns about the four parameters that it predicts - width, height, center of bounding box, and value $c$ representing the class of an object (discussed in sections below) :

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

*Figure 2 shows the loss function given by the developer of YOLO algorithm (Image taken from the paper by Joseph Redmon).*

The regularization term $t$, given the hyperparameter $\lambda$ can be calculated by :

$$t = (\lambda/2)(\|W\|^2 + \|W^1\|^2)$$

The model's regularized loss would be

$$t = (\lambda/2)(\|W\|^2 + \|W^1\|^2)$$

## 3 Algorithms used for object detection problem

### 3.1 YOLOv3 algorithm

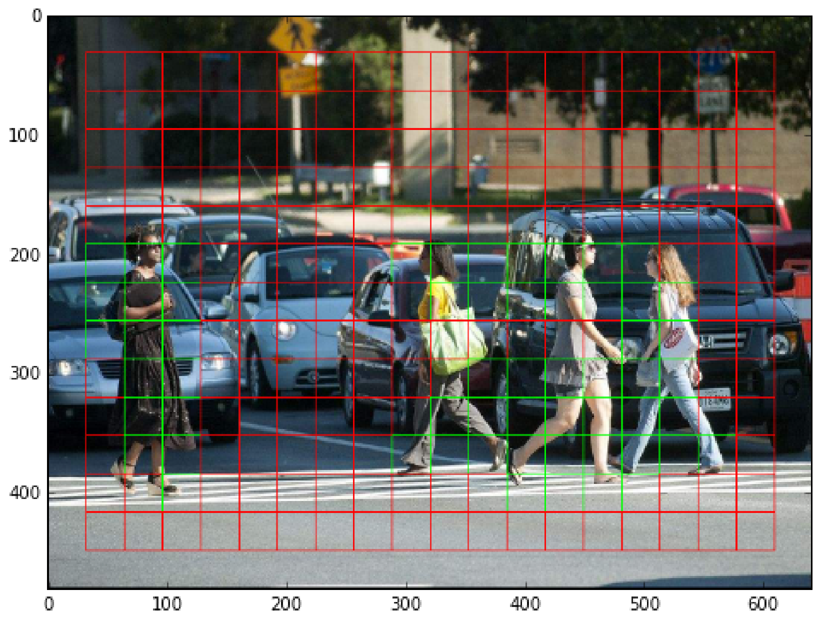
YOLO (You Only Look Once) is a Convolutional Neural Network that achieve state-of-the-art results with a single end-to-end model that perform object detection in real time. Since its release back in 2015 by researchers at the University of Washington, there has been five generations of the algorithm with each one improves its detection speed and accruacy over the last version. YOLO has become one of the most popular algorithm detection methods due to its speed and accuracy over any other algorithms, and it can also be used to demonstrate in real time videos and camera feed input. In this project, the problem will be implemented using version 3 of YOLO (YOLOv3), and pre-trained model weight data file trained on COCO datasets is required in order to perform object detection on test image files.

### 3.2 How YOLO works

YOLO algorithm uses three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union

.
The images were first divided into grids with equal dimension, as shown in example image below. Each cell will detect objects within them.



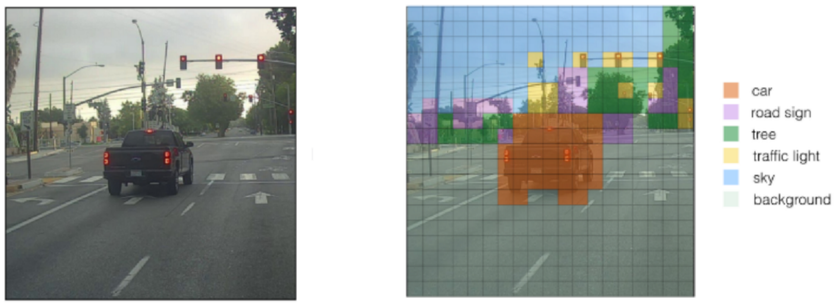
.
YOLO aims to predict a class of an object and the bounding box for object's location. Each bounding box contains four parameter attributes:

- Width
- Height
- x,y coordiniates of the center of bounding box
- Class representing the object

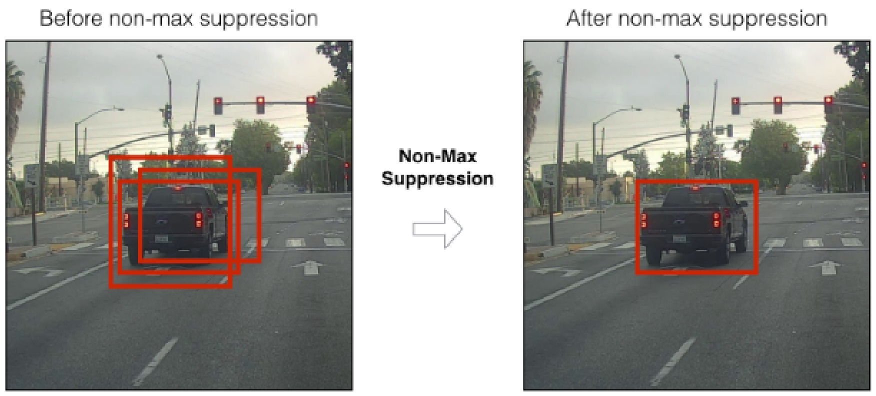
During one pass of forward propagation (discussed above), the algorithm calculates the probability that the cell contains a class:

$$score_{c,i} = p_c * c_i$$

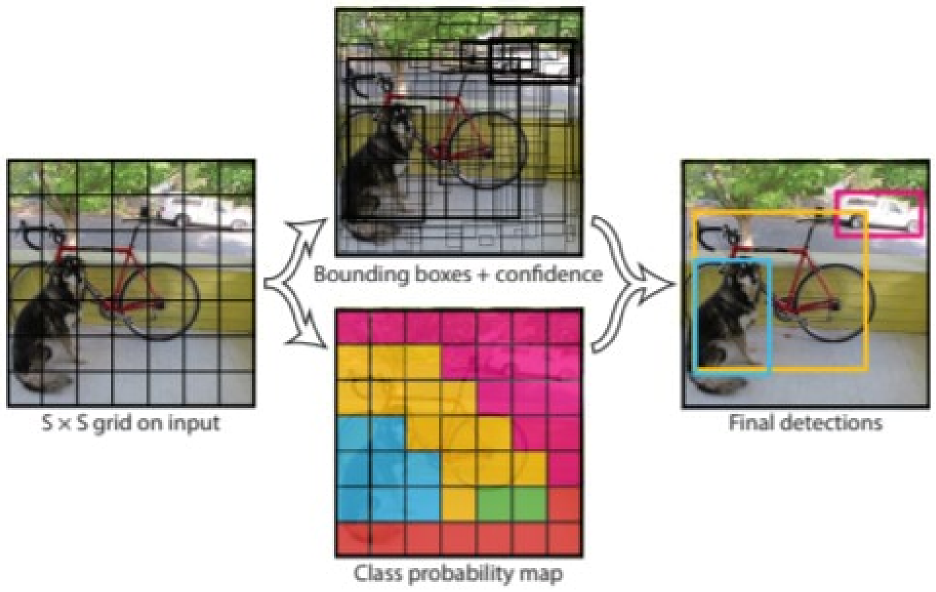
where $p_c$ is the probability that there is an object in the bounding box, $c_i$is the class for that object. The class with maximum probability score is chosen and empolyed to that specific cell. An example after calculating and classifying all the grid cells will look like the image below.



After the probability for all cells are calculated, non-max suppression technique is used to remove all unnecessary boxes. In order to acheive this, intersection over union can be utilized to remove bounding boxes that are not the same as the actual box. The following image shows that after applying this technique, all the unnecessary bounding boxes will be removed.



The flow of the whole algorithm is the combination of these three techniques, a flow chart of YOLO is provided by Joseph Redmon in their paper:



## 4 Implementation and result

### 4.1 Pre-trained model

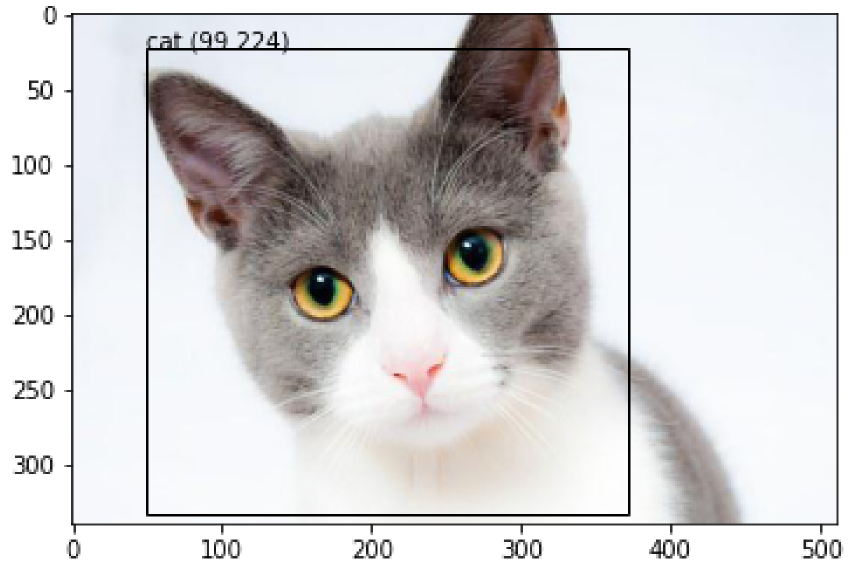
As mentioned above, this project will be using a pre-trained weight file provided by the developers of YOLO. And since a model for YOLO algorithm would be relatively sophisticated to code from scratch, and the developers of the algorithm has already given the implmentations of the model, most people would use some open-sourced library tools to implement object detection. One of the library model implementation is written by Huynh Ngoc Anh, and the implementation for creating the model was taken directly from the functions in that library.

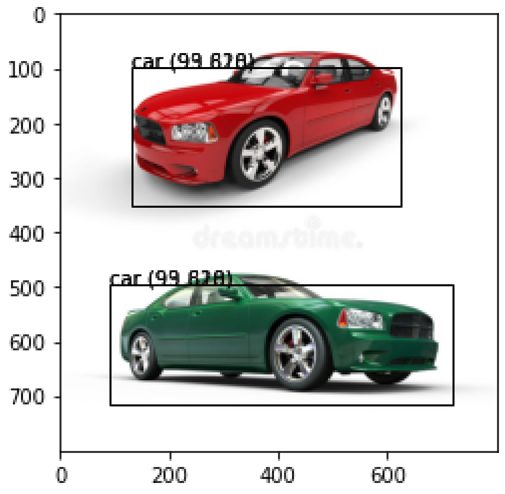
Once the model created based on the pre-trained weight file was saved, we can apply the YOLOv3 algorithm and its techniques discussed in the section above.

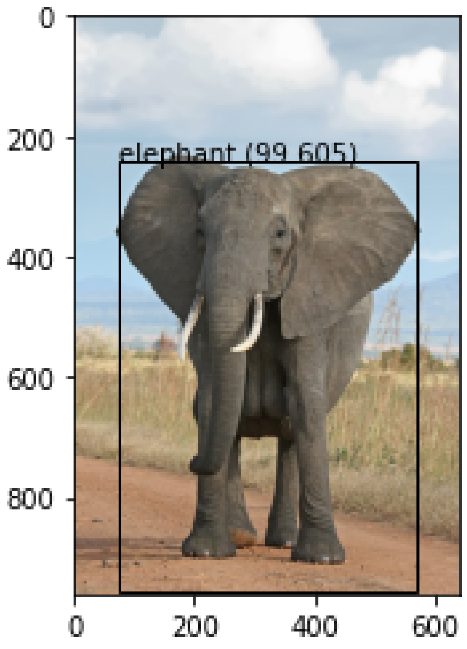
One problem with the algorithm implementation is that intersection over union technique was not successfully implemented, so the results would show the correct result alongside with mutiple unnecessary results unremoved. However, we can still get the only correct bounding box result by selecting the ones with highest probability value, and tests show that the correct boxes will always get a probability value around 99 percent. Another unsolved problem seems to be related to *matplotlib* module, as the plotting right after we get the four parameter value would cause the kernel to crash. Therefore, the current solution is to implement the plotting in another seperate terminal and plot the parameter results we get from the algorithm manually.

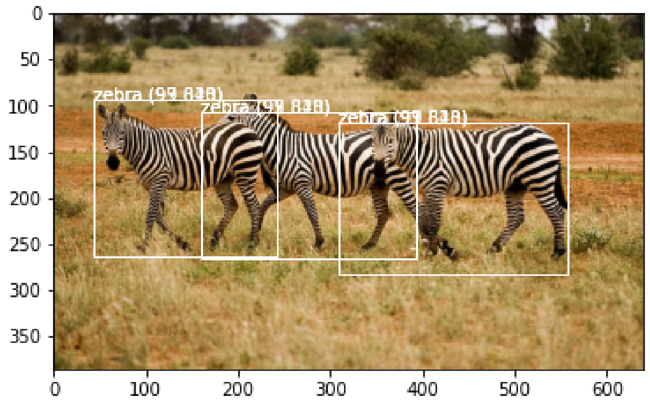
### 4.2 Results and plot

There are a list of class labels used to train the model (specified in the algorithm), and by inputing any random image within the range of this class list, we will be able to get the final result of object detection. Some tests are performed by inputing random image found online, and all of them shows correct bounding box locations, object classes, and their probability value.









As shown in some example tests results above, the final detection recognized the objects' presence, located their coordinates with bounding boxes, and the classes of the objects. Each bounding box shows a high value of probability that the object belongs to the class, which is almost 100 percent confidence level. Furthur tests with random images that are within the range of label list shows similar results, this proves the model and the YOLOv3 algorithms accuracy and performance.

## 5 Conclusion

Implementation and results shows that YOLOv3 produces high accuracy, precision, and efficient results for object detection. It possess the capability to detect one or more objects in any random images with almost 100 percent confidence level.

The model of the project, however, is used directly from libararies developed by others who worked on YOLO algorithm. The model is a pre-trained model trained on COCO datasets. It gives high level of accuracy and performance, but further research and implementation on other training datasets or even our own datasets needs to be done to prove the effectiveness of YOLO algorithm and the model.

As mentioned before, one major problem of the implementation is the lack of intersection over union technique, this causes the results to give extra multiple unneccessary bounding boxes. Even though the correct results can still be outputed every time, plotting needs to be done manually by updating the results from the algorithm each time. Therefore, this problem needs to be addressed and fixed in future research.

In this project folder: implementation is provided. ”object detection” contains the YOLO algorithm, ”object model” contains the pre-trained data model, ”obj plot zebra” is the implementation for plotting the results taken from ”object detection” , ”yolov3.weights” is the weight data file provided by developers, and ”model.h5” is the model data file generated.

## 6 Paper and source references for the project

- **You Only Look Once: Unified, Real-Time Object Detection** by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi at University of Washington. Retrieved from *https://arxiv.org/pdf/1506.02640.pdf*
- **YOLOv3: An Incremental Improvement** by Joseph Redmon and Ali Farhadi at University of Washington. Retrieved from *https://arxiv.org/pdf/1804.02767.pdf*
- **Object Detection with Deep Learning: A Review** by Zhong-Qiu Zhao, Peng Zheng, Shoutao Xu, and Xindong Wu. Retrieved from *https://arxiv.org/pdf/1807.05511.pdf*
- Pre-trained model weight is given by YOLO developers, YOLO website: *https://pjreddie.com/darknet/yolo/*
- COCO dataset and object labels: *https://cocodataset.org/*, *https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/*
- Example images during the ”algorithm” section were referenced from blogs related to the topic: *https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4*
- Pre-trained models were used by open-sourced libaray written by Huynh Ngoc Anh. Some functions were utilized: *https://github.com/experiencor/keras-yolo3*