A

Project Report

On

“*Image Objects Detection By YOLO*”

Submitted By

Harshad Dilipkumar Ahale (2016BTEEN00001)

Shreyas Anil Koshti (2016BTEEN00031)

Deepak Maruti Phadtare (2016BTEEN00062)

Under the guidance of

Dr. S. D. Ruikar

Department of Electronics Engineering

WALCHAND COLLEGE OF ENGINEERING, SANGLI

(An Autonomous Institute)

2019-2020

DEPARTMENT OF ELECTRONICS ENGINEERING

WALCHAND COLLEGE OF ENG1INEERING, SANGLI

(An Autonomous Institute)

CERTIFICATE

This is to certify that the Project-I Seminar Report entitled

Image Objects Detection By YOLO

Submitted by

Harshad Dilipkumar Ahale (2016BTEEN00001)

Shreyas Anil Koshti (2016BTEEN00031)

Deepak Maruti Phadtare (2016BTEEN00062)

in partial fulfilment for the award of the Degree of
Bachelor of Technology
in
Electronics Engineering

is a record of students own work carried out by them under our supervision and guidance during the first semester of academic year 2019-2020.

Date: 14/06/2020
Place: Sangli.

Dr. S. D. Ruikar	**Dr. B. G. Patil**
(Project Guide)	**(HOD)**

ACKNOWLEDGEMENT

We are pleased to present this dissertation report entitled “Image Objects Detection by YOLO” to our college, as part of academic activity. We express our sincere gratitude towards our guide Dr. S. D. Ruikar Sir and Head of Department Dr. B. G. Patil for their constant help, encouragement and inspiration throughout the project work. Without their invaluable guidance, this work would never have been a successful one.

We express a deep sense of gratitude towards all the Asst. Professors of Electronics Department as they provided well support and guidance during the project. Finally, we wish gratefulness to all those people who contributed to our project directly and indirectly.

Abstract:

The computer vision industry is projected to hit [$48.6 billion]() by 2022, with applications in industries ranging from video collaboration and medicine to education and security. Computer vision provides machines with the ability to see and visually sense the world around them, similar to how humans use their own eyes. It pertains to the automatic extraction, analysis and understanding of useful information from one or several images ([BMVA.org]()). Object detection is one form of computer vision that's gaining momentum in both the enterprise and consumer-facing tech communities.

So we were looking for algorithm that will detect object present in given digital image. There are several algorithms which can perform that operation. After lots of comparison and stats we decided to use YOLO (You only look once).YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we 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. Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork. We implemented the YOLO in MATLAB.

Content:

Chapter 1

Introduction:

Object detection is a computer technology related to computer vision and image processing that detects and defines objects such as humans, buildings and cars from digital images and videos (MATLAB). This technology has the power to classify just one or several objects within a digital image at once. Object detection has been around for years, but is becoming more apparent across a range of industries now more than ever before

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems. Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image.

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [Fig1]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately. We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. This system, you only look once (YOLO) at an image to predict what objects are present and where they are. By this way we can reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using you only look once (YOLO) at an image to predict what objects are present and where they are. YOLO is refreshingly simple: see Figure 1. A single

Convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

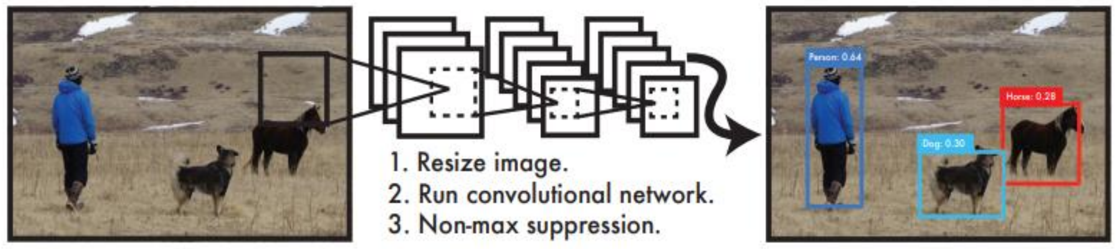


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Chapter 2

Present Scenario:

Object detection has always been an interesting problem in the field of deep learning. It’s primarily performed by employing convolutional neural networks (CNNs), and specifically region-based CNNs (R-CNNs).

R-CNNs (Region based CNNs)

Instead of working on a massive number of regions, or going through each and every pixel and localized part of the image to search for the presence of an object, the R-CNN algorithm proposes a bunch of boxes within an input image and checks to see if any of these boxes contain any object. R-CNN then uses selective search to extract these boxes from an image.

Issues with R-CNNs:

Training an R-CNN model is expensive and slow because:

- We must extract 2,000 regions for each image based on selective search
- We also have to extract features using a CNN for every image region. Suppose we have N images—the number of CNN features will be N*2,000
- These processes combine to make R-CNN very slow. It takes around 40–50 seconds to make predictions for each new image, which essentially makes the model cumbersome and practically impossible to build when faced with a gigantic dataset.
  So a better alternative is YOLO algorithm.

Chapter 3

Literature Survey:

In search of finding the best algorithm for object detection, we read some research papers like :

1) You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi University of Washington, Allen Institute for AI, Facebook AI Research
2) Introduction to basic object detection algorithms written by Nikita Sharma

After studying both of these papers we came up with following conclusion. YOLO's architecture is based on CNNs using anchor boxes and is proven to be the go-to object detection technique for a wide range of problems like detection of vehicles on the road for traffic monitoring, detection of humans in an image etc. YOLO employs an F-CNN (fully convolutional neural network). It passes an input image (n x n) once through the F-CNN and outputs a prediction (m x m).

Thus, the architecture is splitting the input image in an (m x m) grid, and for each grid, generates 2 bounding boxes and class probabilities for those bounding boxes.

Bounding boxes contain potential objects and the class probabilities that correspond to the object's class.

Comparison to Other Detection Systems:

Object detection is a core problem in computer vision. Detection pipelines generally start by extracting a set of robust features from input image. Then, classifiers or localizers are used to identify objects in the feature space. These classifiers or localizers are run either in sliding window fashion over the whole image or on some subset of regions in the image. We compare the YOLO detection system to several top detection frameworks, highlighting key similarities and differences.

Deformable parts models:

Deformable parts models (DPM) use a sliding window approach to object detection. DPM uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with a single convolutional neural network. The network performs feature extraction, bounding box prediction, nonmaximal suppression, and contextual reasoning all concurrently. Instead of static features, the network trains the features in-line and optimizes them for the detection task. Our unified architecture leads to a faster, more accurate model than DPM.

R-CNN:

R-CNN and its variants use region proposals instead of sliding windows to find objects in images. Selective Search generates potential bounding boxes, a convolutional network extracts features, an SVM scores the boxes, a linear model adjusts the bounding boxes, and non-max suppression eliminates duplicate detections. Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time.

YOLO shares some similarities with R-CNN. Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. YOLO system also proposes far fewer bounding boxes, only 98 per image compared to about 2000 from Selective Search. Finally, our system combines these individual components into a single, jointly optimized model

Other Fast Detectors:

Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search . While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance. Many research efforts focus on speeding up the DPM pipeline. They speed up HOG computation, use cascades, and push computation to GPUs. However, only 30Hz DPM actually runs in real-time. Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design. Detectors for single classes like faces or people can be highly optimized since they have to deal with much less variation. YOLO is a general purpose detector that learns to detect a variety of objects simultaneously.

Deep MultiBox:

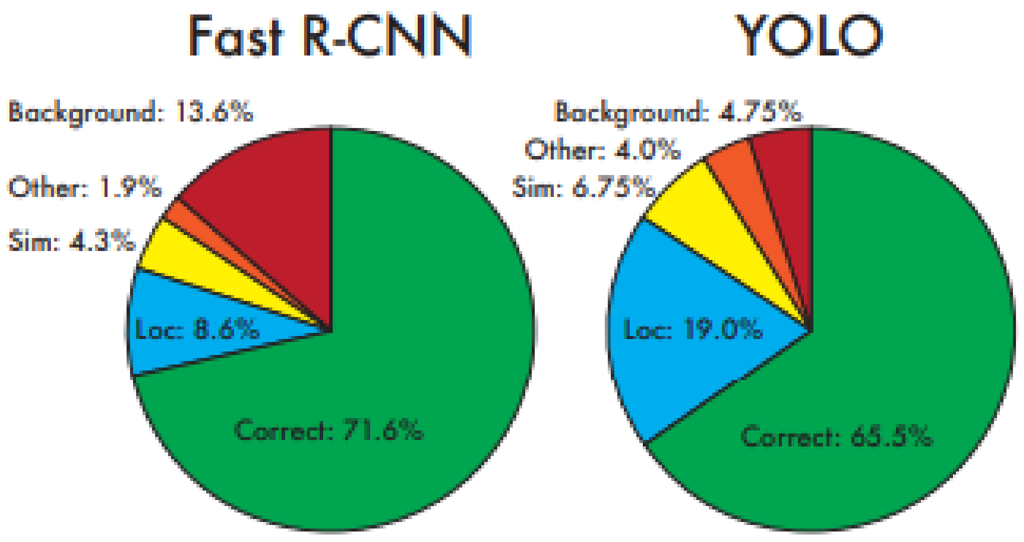
Unlike R-CNN, **Szegedy et al.** train a convolutional neural network to predict regions of interest [8] instead of using Selective Search. MultiBox can also perform single object detection by replacing the confidence prediction with a single class prediction. However, MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. Both YOLO and MultiBox use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection system.

OverFeat:

Sermanet et al train a convolutional neural network to perform localization and adapt that localizer to perform detection. OverFeat efficiently performs sliding window detection but it is still a disjoint system. OverFeat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

Comparison to Other Real-Time Systems:

Many research efforts in object detection focus on making standard detection pipelines fast. However, only **Sadeghi et al**. actually produce a detection system that runs in real-time (30 frames per second or better). We compare YOLO to their GPU implementation of DPM which runs either at 30Hz or 100Hz. While the other efforts don’t reach the real-time milestone we also compare their relative mAP and speed to examine the accuracy-performance trade-offs available in object detection systems. Fast YOLO is the fastest object detection method on PASCAL; as far as we know, it is the fastest extant object detector. With 52.7% mAP, it is more than twice as accurate as prior work on real-time detection. YOLO pushes mAP to 63.4% while still maintaining real-time performance.



Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Chapter 4

Software Requirement of project:

MATLAB (*matrix laboratory*) is a multi-paradigm numerical computing environment and proprietary programming language developed by Math-Works. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

Current Stable release: R2020a / March 19, 2020

This Project uses following **MATLAB** features:

1) Computer Vision Toolbox
2) Deep Learning Toolbox
3) Deep Learning Toolbox Model for ResNet-50 Network
4) Image Processing Toolbox

Chapter 5

YOLO Concept Brief Explanation:

1. Unified Detection

Unify the separate components of object detection into a single neural network. Proposed network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real time speeds while maintaining high average precision.

System divides the input image into an S × S grid. If the centre 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 we define confidence as Pr(Object) ∗ IOUtruthpred . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

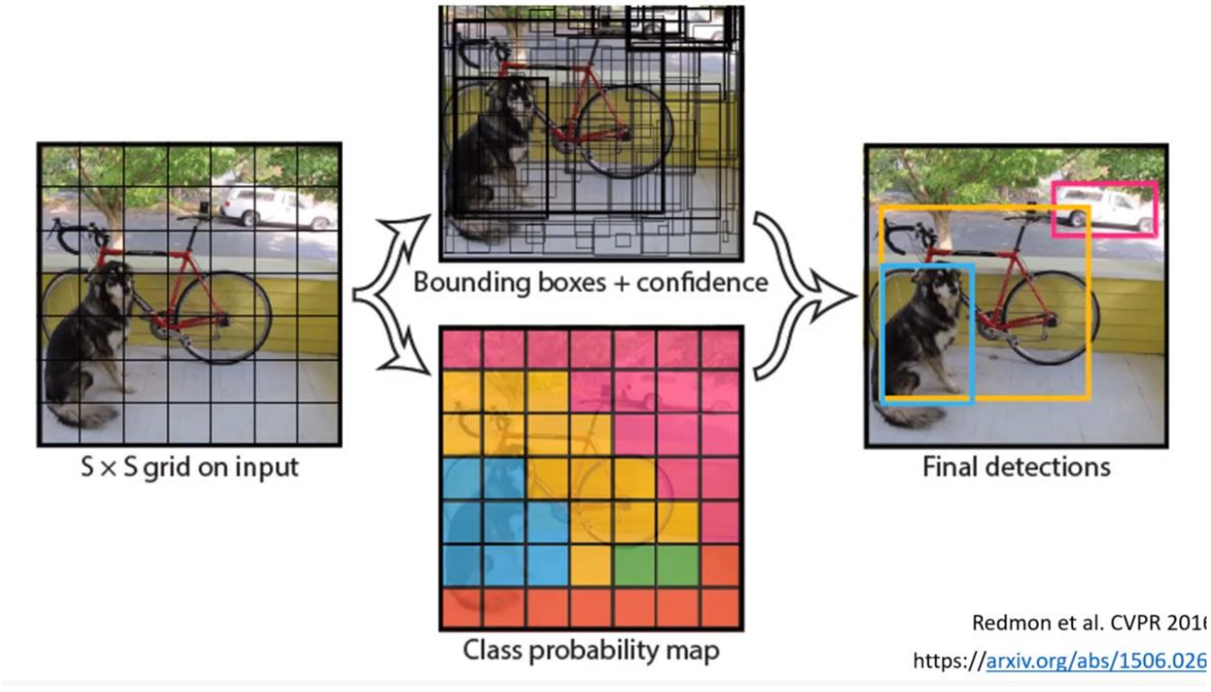
Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the centre of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, Pr(ClassiObject). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

Pr(Classi|Object) ∗ Pr(Object) ∗ IOUtruthpred = Pr(Classi) ∗ IOUtruthpred (1)

Which gives us class specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.



System models detection as a regression problem. It divides the image into an S × 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.

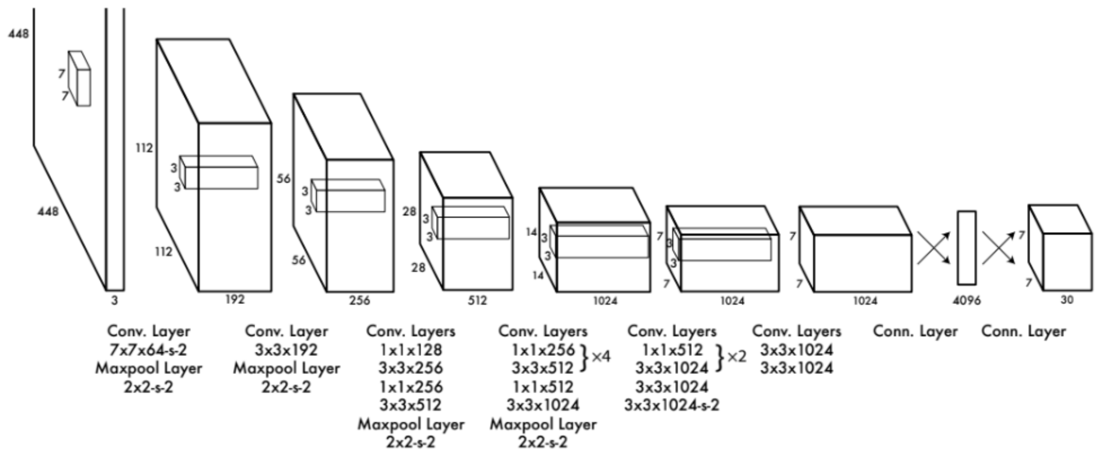
For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a 7 × 7 × 30 tensor.

2. Network Design

Implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. Our network architecture is inspired by the GoogLeNet model for image classification. Our network has 24 convolutional layers followed by 2 fully connected layers.

Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure

Also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.



The Architecture, detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pre-train the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

3. Training

Pre-trained our convolutional layers on the ImageNet 1000-class competition dataset. For pre-training we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo.

Then convert the model to perform detection. Renetal show that adding both convolutional and connected layers to pre-trained networks can improve performance. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448.

Final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parameterize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

Use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

$$\varphi(x) = \left(\begin{array}{ll} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{array} \right)$$

(2) Optimize sum-squared error in the output of our model. Use of sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λcoord and λnoobj to accomplish this. We set λcoord = 5 and λnoobj = .5.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall

Recall.

During training we optimize the following, multi-part loss function:

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

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

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

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

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

3) Where $\mathbb{I}_i^{obj}$ denotes if object appears in cell i and $\mathbb{I}_{i\,j}^{obj}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012. When testing on 2012 we also include the VOC 2007 test data for training. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.

Our learning rate schedule is as follows: For the first epochs we slowly raise the learning rate from 10^{-3} to 10^{-2}.

If we start at a high learning rate our model often diverges due to unstable gradients. We continue training with 10−2 for 75 epochs, then 10^{-3} for 30 epochs, and finally 10^{-4} for 30 epochs.

To avoid over fitting we use dropout and extensive data augmentation. A dropout layer with rate = 5 after the first connected layer prevents co-adaptation between layers. For data augmentation we introduce random scaling and translations of up to 20% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

3. Inference

Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.

The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2-3% in mAP.

Chapter 6

Proposed Work:

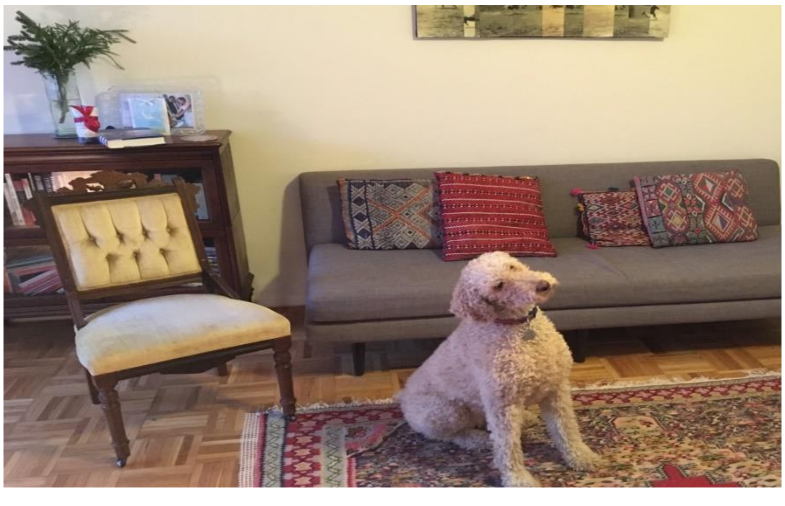
1. Downloading YOLONET and modify for regression

The YOLO network that is available from Math-works requires modification before it can be used for object detection. The network you will download contains final layers for a classification algorithm; a classification layer and a softmax layer. However, YOLO is actually structured as a CNN regression algorithm. The output should be an array or vector of numbers between 0 and 1 which encode probabilities and bounding box information for objects detected in an image rather than a series of 1 and 0's.

The last two layers need to be replaced with a single regression layer. In addition, for concordance with the original YOLO papers (see above), the last leaky ReLu transfer function needs to be replaced with a standard (non-leaky?) ReLu.The following code downloads the network, modifies the layers, and saves the resulting modified network to the current folder in MATLAB.

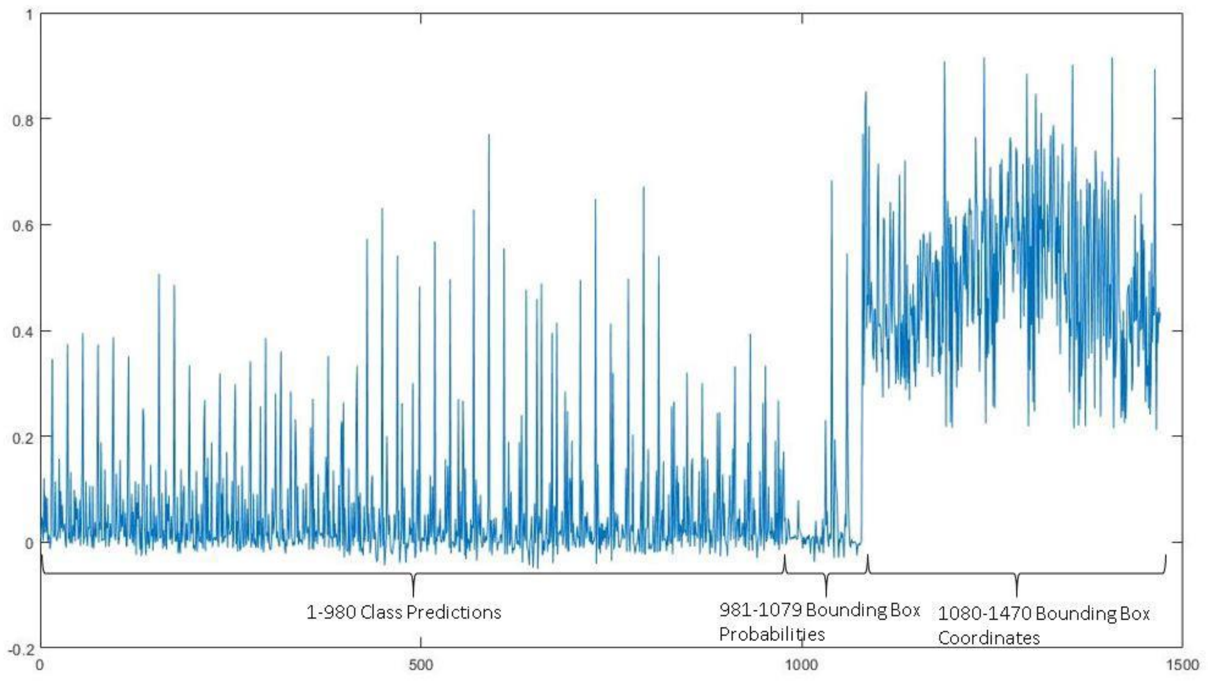
2. Running an image through the network and examine the output vector

To test my implementation of YOLO, We summoned the heights of my visual art abilities and took a snapshot that contained four objects that YOLO has been trained on — a chair, dog, potted plant, and sofa. Here is test image:



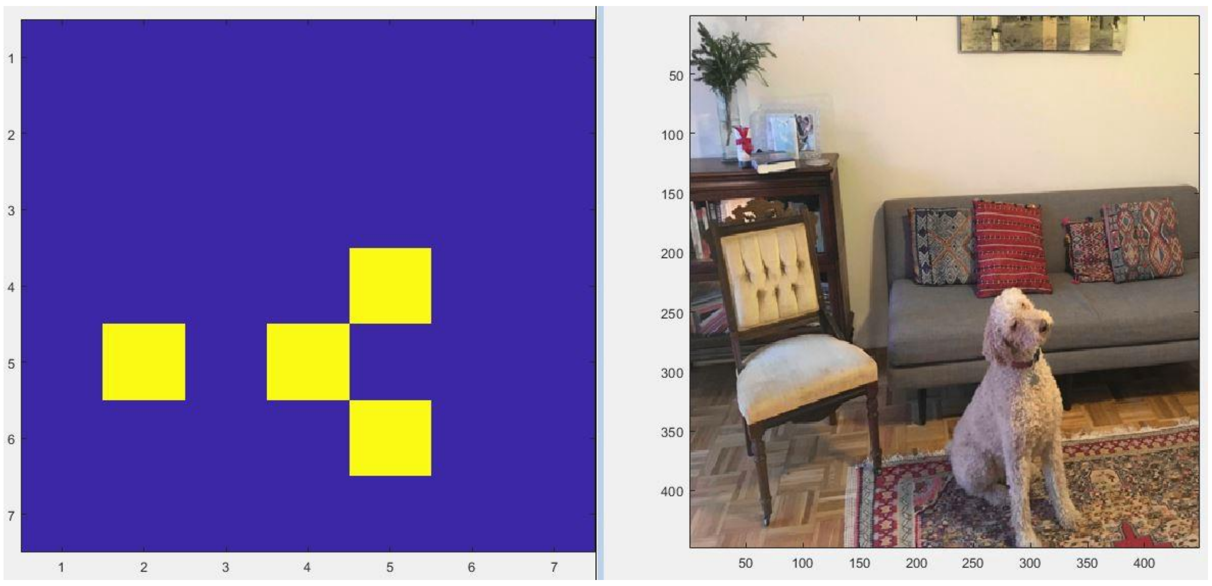
Because YOLO is a regression network, we will use the function *predict* rather than *classify*. Image pixels need to be scaled to [0,1] and images need to be resized to 448x448 pixels. If your image is gray scale (i.e. single channel) it needs to be expanded to 3 channels. The following code pre-processes an image (you will need to supply your own image in the MATLAB current folder), applies the regression network to it, and plots the resulting 1x1470 output vector. In this section of code, we also define a probability threshold for a cell containing an object (0.2 seems to work well) and an intersection over union threshold for non-max suppression, both of which we will use later.

Let's look at the output vector of the YOLO algorithm. You can reverse engineer the output by noting that indices 1–980 peak approximately every 20 positions corresponding to the 20 classes of objects YOLO has been trained on, indices 981–1079 contain a few peaks corresponding to the probabilities of handful of recognizable objects (4) in my test image, and indices 1080–1470 are fairly stochastic which is what you would expect from the bounding box coordinates.



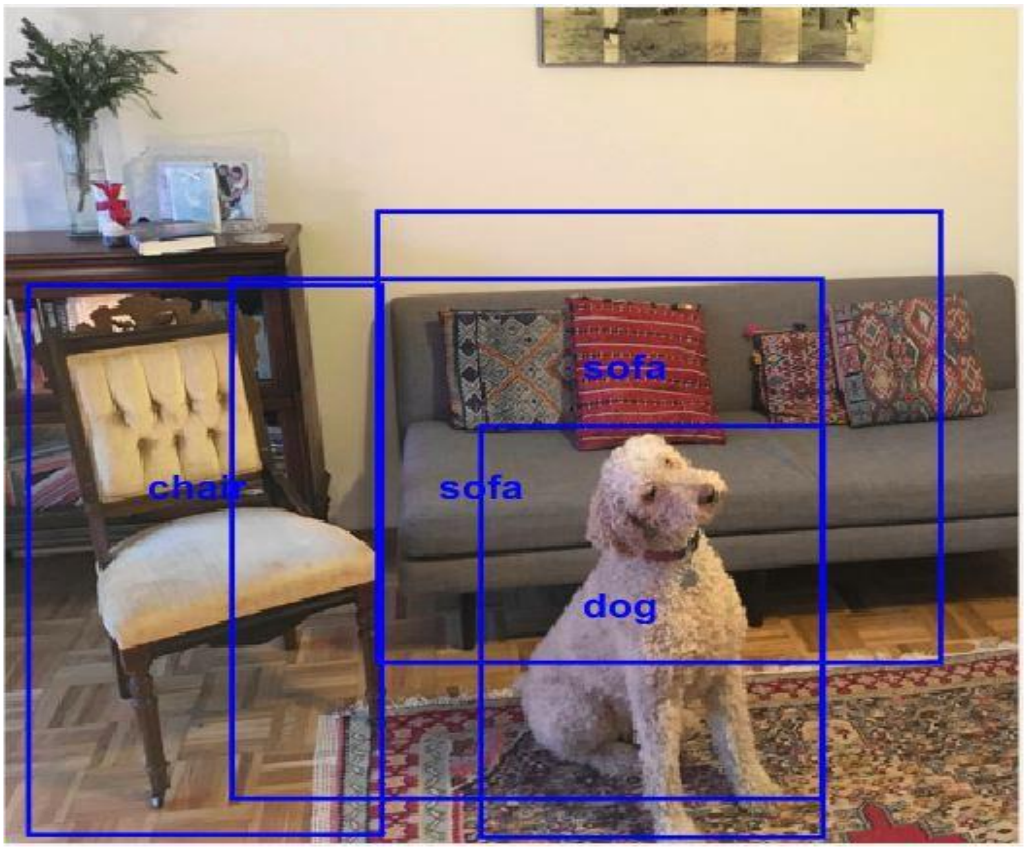
3. Reshape the output vector and find cells containing objects

Now that we have a key to the output of the YOLO algorithm, we can begin the process of converting that 1x1470 output vector into bounding boxes overlaid onto our test image. As a first step, we decided to convert the 1x1470 vector into a 7x7x30 array where the first two dimensions correspond to the 7x7 cells that YOLO divides the image into, and the last dimension contains the 30 numbers defining probabilities and bounding boxes for each cell. After we do this, we can make a simple plot showing cells that are likely to contain objects. Comparing the plot below to the original test image, we see that the algorithm appears to be detecting the chair, dog, and sofa, but seems to be missing the potted plant.



4. Plotting Bounding Boxes and Class labels

Now that we have a nice 7x7x30 output array, we can plot bounding boxes and class labels over the original image. We will now going to make use of the probe-Thresh variable from earlier in the code. You made need to tweak this value for optimal results, but We've found that 0.2 seems to work well for a variety of images. We won't say much about the following code except that it's a bit tedious and makes my head hurt.

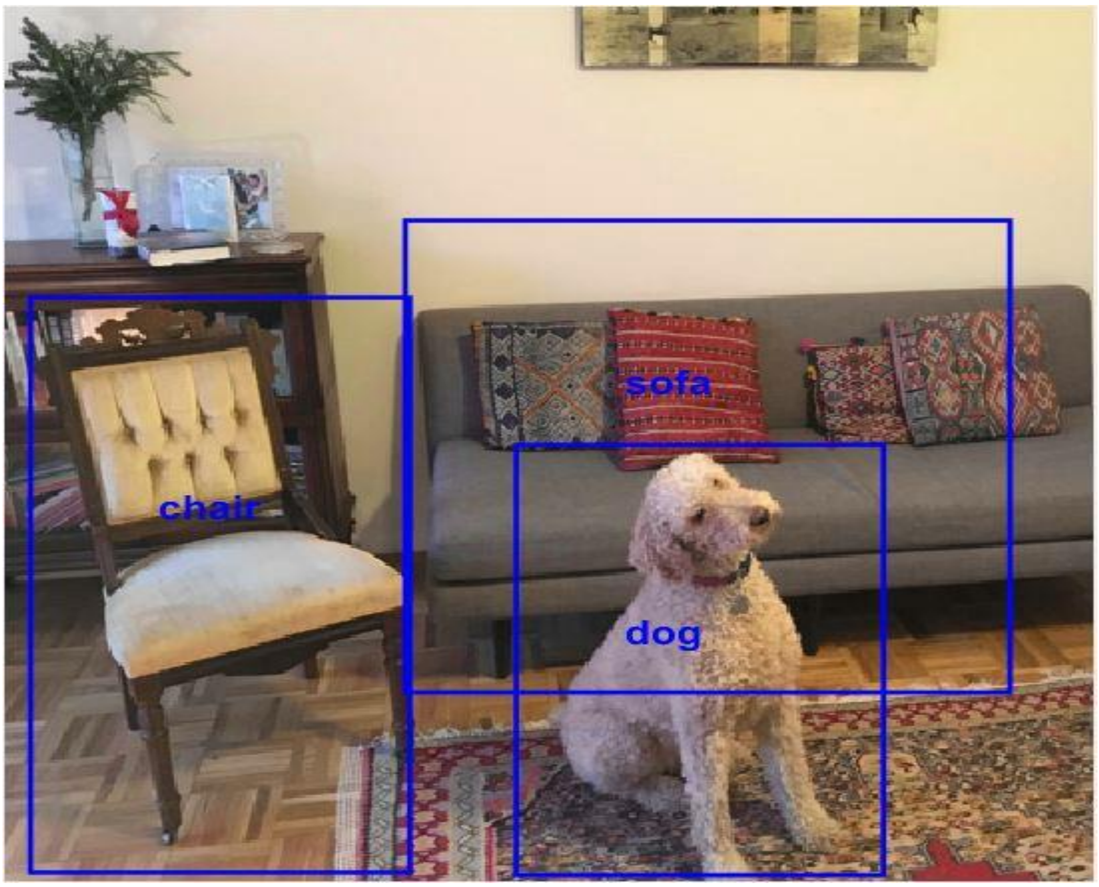


5. Non-max suppression

As you can see above, there are two immediate problems with our test image: 1) The sofa is marked twice, and 2) the potted plant is not marked. The chair and Stella the Dog are classified perfectly. We will correct the first problem (too many sofas) using non-max suppression and the second problem we will ignore. In reality, the object in the upper left isn't technically a potted plant, it is a few evergreen boughs in a vase of water which we call a Manhattan Christmas Tree.

Non-max suppression will remove a bounding box if:

1. It's intersection over union (IOU) with a bounding box B is above a defined threshold
2. Bounding boxes A and B contain the same class of object
3. Bounding box A has a lower probability of containing an object than box B

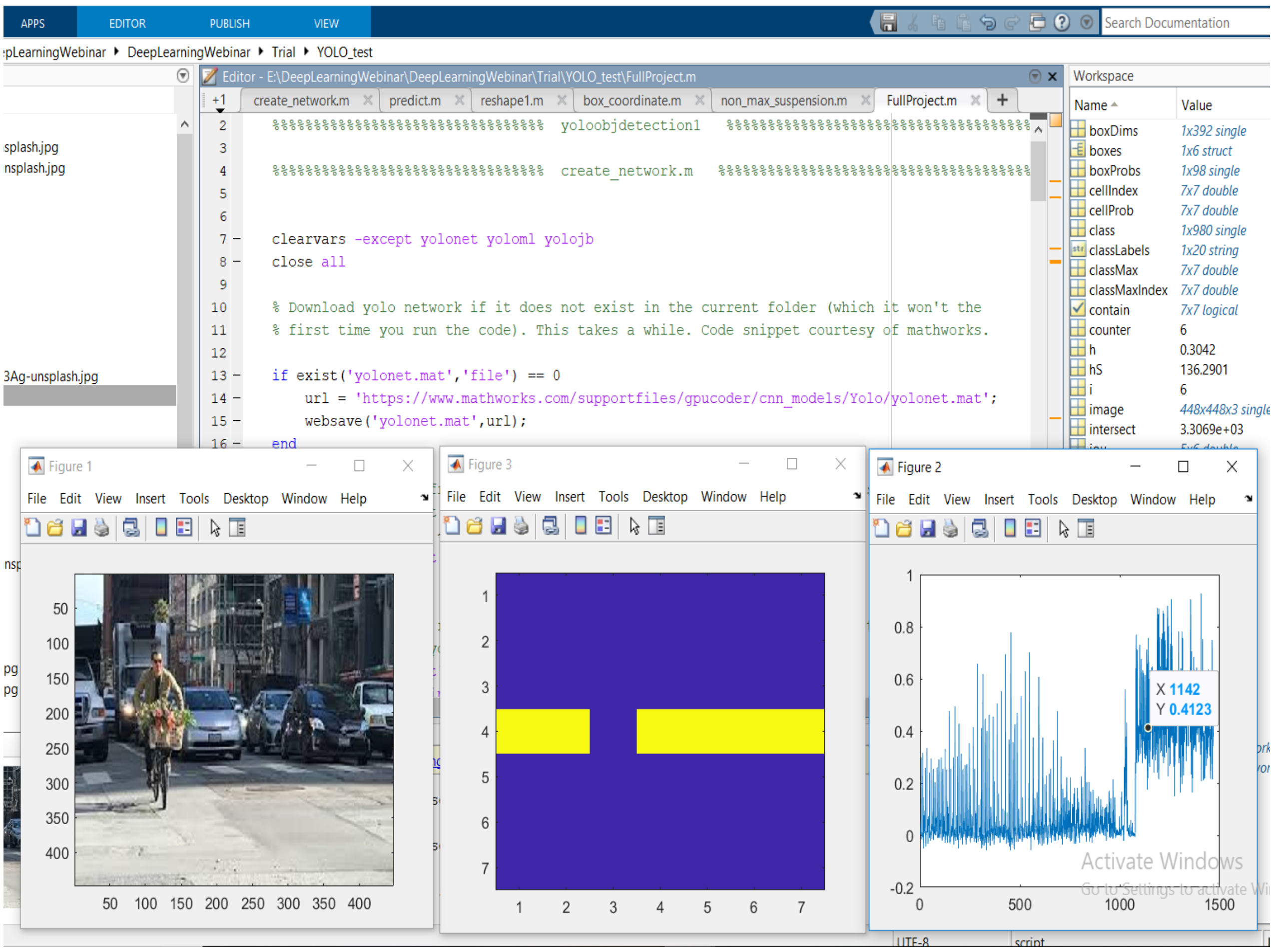


Chapter 7

Results:

Snapshot of project Result :

1)



Editor - E:\DeepLearningWebinar\DeepLearningWebinar\Trial\YOLO_test\FullProject.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  yoloobjdetection1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  create_network.m   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clearvars -except yolonet yoloml yolojb
close all

% Download yolo network if it does not exist in the current folder (which it won't the
% first time you run the code). This takes a while. Code snippet courtesy of mathworks.

if exist('yolonet.mat','file') == 0
    url = 'https://www.mathworks.com/supportfiles/gpucoder/cnn_models/Yolo/yolonet.mat';
    websave('yolonet.mat',url);
end
```

Workspace

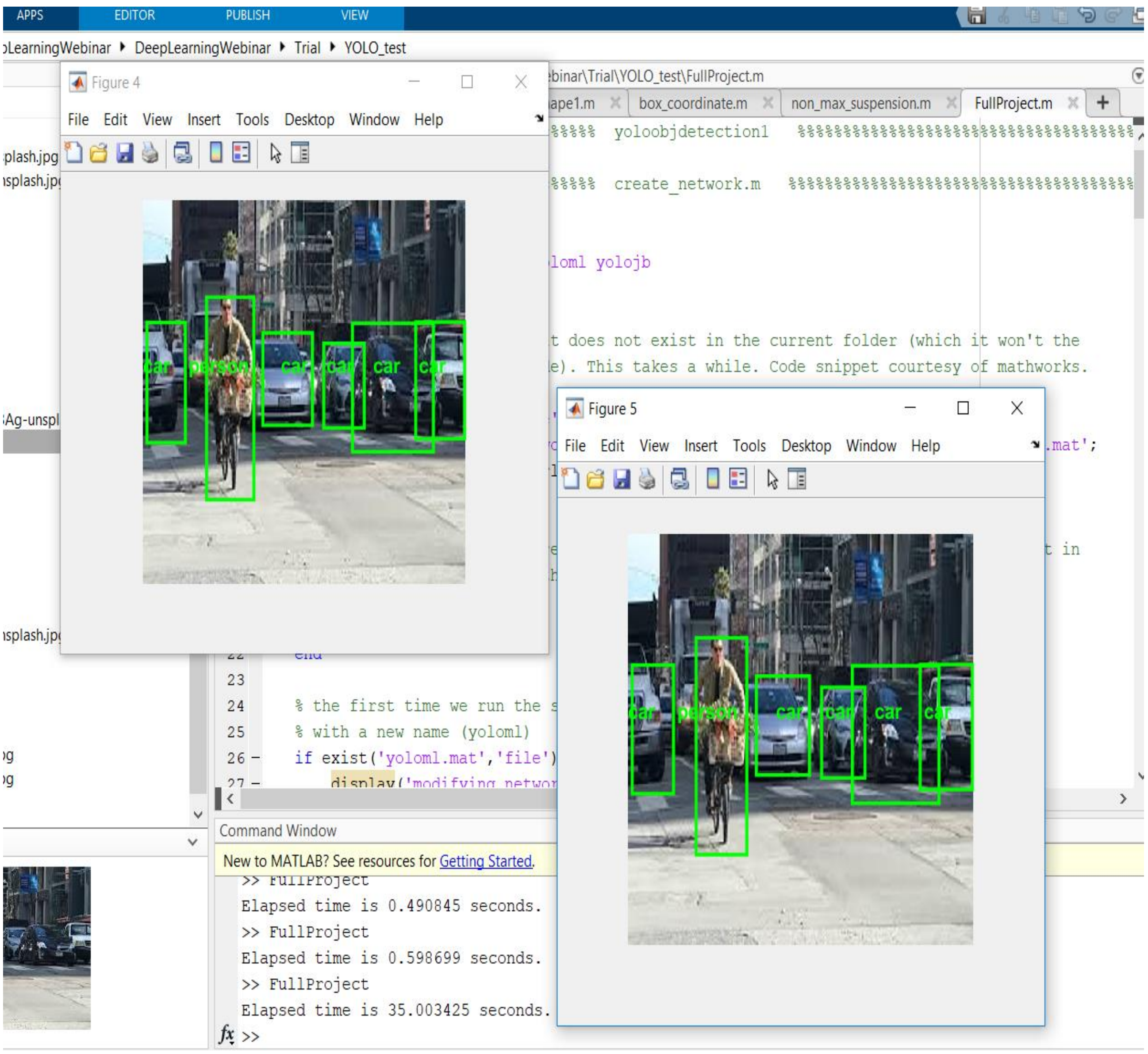
Name	Value
boxDims	1x392 single
boxes	1x6 struct
boxProbs	1x98 single
cellIndex	7x7 double
cellProb	7x7 double
class	1x980 single
classLabels	1x20 string
classMax	7x7 double
classMaxIndex	7x7 double
contain	7x7 logical
counter	6
h	0.3042
hS	136.2901
i	6
image	448x448x3 single
intersect	3.3069e+03

1) Figure 1- Input Image.

2) Figure 2- Output Vector.

3) Figure 3- Reshape the output vector and find cells containing objects.

2)


APPS
EDITOR
PUBLISH
VIEW
LearningWebinar ▸ DeepLearningWebinar ▸ Trial ▸ YOLO_test
binar\Trial\YOLO_test\FullProject.m
ape1.m
box_coordinate.m
non_max_suspension.m
FullProject.m
Figure 4
Figure 5
File Edit View Insert Tools Desktop Window Help
car
person
car
car
car
car
Command Window
New to MATLAB? See resources for Getting Started.

```
%%%%%  yoloobjdetection1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%  create_network.m   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

loml yolojb

t does not exist in the current folder (which it won't the
e). This takes a while. Code snippet courtesy of mathworks.

.mat';

t in

    end

    % the first time we run the s
    % with a new name (yoloml)
    if exist('yoloml.mat','file')
        display('modifying networ
```

```
>> FullProject
Elapsed time is 0.490845 seconds.
>> FullProject
Elapsed time is 0.598699 seconds.
>> FullProject
Elapsed time is 35.003425 seconds.
fx >>
```

4) **Figure 4-** Plotting Bounding Boxes and Class labels

5) Figure 5 – Non-Max suppression.

Chapter 8

Merits of YOLO:

1) YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline.
2) We simply run our neural network on a new image at test time to predict detections.
3) Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency.
4) Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems.

Demerits of YOLO:

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image.

Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

Chapter 9

Conclusion:

We introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

Fast YOLO is the fastest general-purpose object detector in the literature and YOLO pushes the state-of-the-art in real-time object detection. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection.

Chapter 10

Future Scope:

1) Right now the object detection is performed on pre-captured image, And it works well and also it is fast compare to other techniques.

2) Improvement can be done by applying this technique to real time video capturing and objects detection.

3) This technique will be applied in many application like Auto-driving Cars, counting similar object from group of objects, counting no of peoples from the group of peoples etc.

Chapter 11

References:

- [1] You Only Look Once: Unified, Real-Time Object Detection
  Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
  -University of Washington, Allen Institute for AI, Facebook AI Research

- https://in.mathworks.com/help/vision/examples/object-detection-using-yolo-v2-deep-learning.html

- [2] Real-Time Object Detection with Yolo Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam
  - International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019

- [3] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In InternationalConference on Computer Vision (ICCV), 2009. 8

- [4] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, J.Yagnik, et al. Fast, accurate detection of 100,000 object classes on a single machine. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1814–1821. IEEE, 2013.

- [5] S. Gidaris and N. Komodakis. Object detection via a multiregion & semantic segmentation-aware CNN model. CoRR, abs/1505.01749, 2015.
- [6] 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.