ECE 697 Capstone Project: Automated MRI Prescription of livers

Mahalakshmi Sundaresan (msundaresan2@wisc.edu)

August 14, 2020

1 Introduction

Magnetic Resonance Imaging (MRI) is a preferred method of imaging livers to check for lesions as it produces comparable or superior results in comparison to Computerized Tomography (CT) scans [1]. MRIs tend to give better contrasts that can expose subtle variation in tissues and involves no ionizing radiation making it safe. It can be captured in any orientation and is not restricted to the axial plane like the CT [2]. However, MRI examinations take a longer duration. During an MRI examination, the patient goes through an MRI scanner or system with powerful magnetic fields. The applied radio waves interact with the protons present in the body tissue aligned by the magnetic field and produces a signal which is received by the MR scanner. Processing these signals result in images of the tissues which can be viewed in any orientation [3]. The MRI technologists or radiologists then manually select and draw a bounding box over the region of interest (liver) in the localizer images. These acquisitions are then used to quantify the fat content or lesions present in the region. This examination process typically takes around 15-45 minutes and might also extend over 60 minutes. The localizer images have a special (unusual) choice of imaging planes that includes a few slices obtained in the coronal plane, a few in the sagittal plane, and a few in the axial (transverse) plane. These 3-plane localizers are helpful for the technologist to identify structures such as the liver. The bounding box should cover:

- The liver in the S/I (head-to-feet) dimension
- The whole abdomen in the R/L (left-to-right) dimension
- The whole abdomen in the A/P (anterior-posterior) dimension

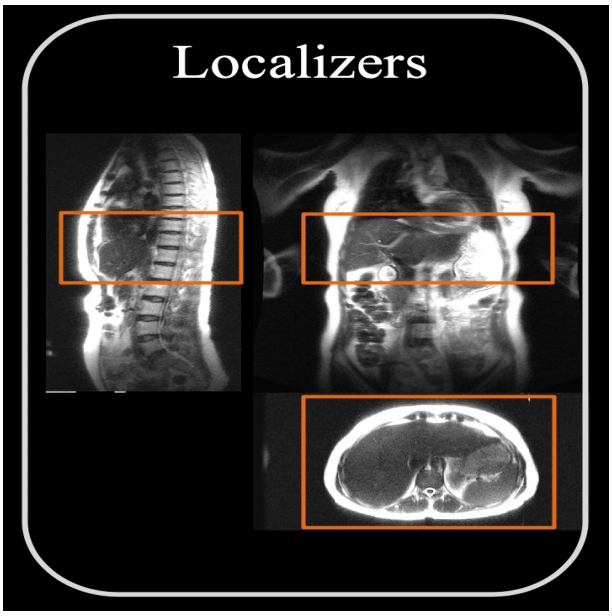


Figure 1: Localizer Images

Eliminating the need for manual intervention to define the bounding box using machine learning networks and computer vision algorithms will bring down the time consumed significantly with improved efficiency.

2 Problem Statement

In this project, we seek to implement the step of identifying the liver and bounding a box around it. This involves performing object detection and we would expect the neural network to output the coordinates of the bounding box. This can be performed by adopting YOLO (You Only Look Once) Algorithm which grids the input image and performs classification and localization algorithm in each cell of the grid. The output vector will indicate whether the object is present or not along with the coordinates of its bounding box. It is advantageous since it gives precise positions of the bounding box coordinates as they are computed relative to the grid cell. Intersection over Union (IoU) and Non-max suppression is done to avoid selecting overlapping boxes.

3 Literature Review

Object Recognition is a general term for all computer vision tasks that involve identifying objects in digital images. Object detection is a combination of image classification and object localization tasks. Region-based Convolutional Neural Networks, or R-CNNs, are a family of techniques to perform object detection which was developed by Ross Girshick, et al. in 2014 in the paper titled "Rich feature hierarchies for accurate object detection and semantic segmentation." [4]. It involves three modules, wherein the first module is used to generate and

extract category independent region proposals. The second module is used to extract features from each of the candidate region using a deep convolutional neural network and the third module is used for classifying the features as one of the known classes like a linear SVM classifier model [5].

There were a few limitations to R-CNN owing to the fact that training was a multi-stage pipeline which involved preparation and operation of three separate models, it was expensive in space and time as there were many region proposals in every image and object detection was also slow [5]. This was further improved upon as fast R-CNN in 2015 [6] and faster R-CNN in 2016 [7] and 2017 [8].

You Only Look Once, or YOLO is a second family of techniques for object detection designed for speed and real-time use. It was first described in 2015 by Joseph Redmon et al., [9]. Its higher speed is attributed to the fact that this model passes the whole image to the neural network once for predicting the bounding boxes along with their class probabilities making it a single forward pass. After an update in 2016 as YOLOv2 [10] with an ability to predict 9000 classes (also called YOLO9000), Joseph Redmon and Ali Farhadi came up with YOLOv3 [11] in 2018.

YOLOv3 has 53 convolutional layers called Darknet-53. It splits the input image into SxS grid of cells. Each grid cell is used to predict the B bounding boxes and class probabilities of the objects with centers falling in the grid cell. The (5+C) attributes corresponding to each bounding box are center coordinates(bx, by) and shape (bh and bw) of the bounding box, one value corresponding to the confidence score, and C indicating the number of classes. The confidence score can range between 0 and 1 indicating how confident an object is in the box. It gives a 3-D tensor with the shape of [S, S, B *(5 + C)]. So, for the COCO dataset with the number of classes C=80 and B=3, the YOLOv3 outputs a tensor with the shape of [(13, 13, 3 * (5 + 80)] [12].

In addressing a similar problem statement for identifying cholelithiasis and classifying gallstones on CT images, Shanchen Pangand et al. used the YOLO v3-arch model. The results show that the accuracy of liver and gallbladder detection is comparatively high, with an average of more than 95 percent.[13]

However, this problem has not been worked on MRI images yet and this project is a novel attempt at working on this set of data.

4 Theory

The YOLOv3 network predicts 4 coordinates for each bounding box using , tx, ty, tw, th. Considering the cell is offset from the top left corner of the image by (cx, cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to: bx, by, bw and bh.

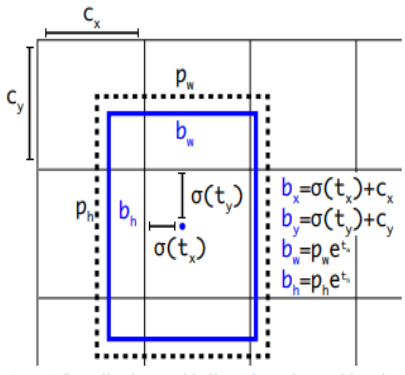


Figure 2: Bounding Box
Image source: https://pjreddie.com/media/files/papers/YOLOv3.pdf

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Figure 3: Bounding Box Coordinates
Image source: https://pjreddie.com/media/files/papers/YOLOv3.pdf

An improvement from earlier versions of YOLO in YOLOv3 is that it predicts an objectness score for each bounding box. This value will be 1 if the bounding box prior(anchor) intersects a ground truth object by more than any other bounding box prior (anchor). Also, YOLOv3 does not use softmaxing of classes. Softmaxing classes assume that classes are mutually exclusive. There could arise a failure in cases where there are two separate classes for person and women in a dataset. Thus, in YOLOv3, logistic regression is used to predict each class score and a threshold determines the label for the object in case of multiple labels.Classes with scores higher than this threshold are assigned to the box. [14]

The loss function of YOLO corresponds to the regression loss, followed by the confidence loss and then classification loss as seen in the image below.

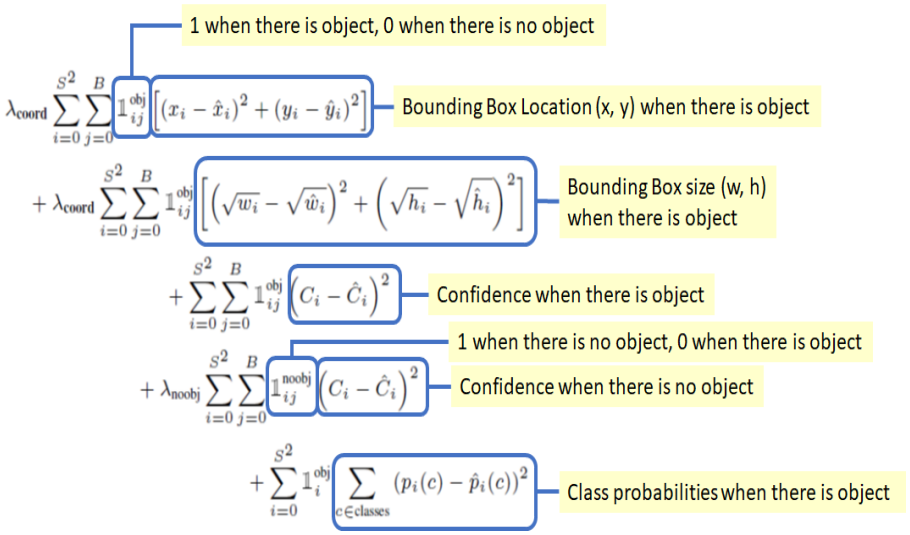


Figure 4: Loss function
Image source: https://pjreddie.com/media/files/papers/yolo$_1$.*pdf*

Evaluation Metrics:
Precision: It is a measure of how accurate the prediction is based on the percentage of correct predictions. It is the ratio of true positives over the sum of true positives and false positives.
Recall: It indicates how good all the positives are found. It is calculated as the ratio of true positives over the sum of true positives and false negatives.

Average Precision[15]: In general, Intersection over Union (IoU) is calculated to determine if the object detection was done right. IoU(A,B)=A$\cap B/A \cup B$,

Where set A represents the proposed object pixels and set B represents the true object pixels.

Generally, IoU > 0.5 means that it was a hit, otherwise it was a fail. For each class, there is:

- "True Positive (TP for class c): a proposal was made for class c and there actually was an object of class c." [15]
- "False Positive (FP(c)): a proposal was made for class c, but there is no object of class c."[15]

Average Precision is $TP(c)/(TP(c) + FP(c))$ and
mAP (mean average precision) is $1/|classes|*(c\in$classes TP(c) / (TP(c)+FP(c)))

5 Method

I implemented this project in three phases. Initially, I used the pre-trained weights available online to implement YOLOv3 on the COCO2014 dataset. It had 80 classes. In the second phase, I trained the algorithm for a custom dataset with two classes of man and cycle. Finally, I trained the algorithm on the liver dataset. I used Google Colab for its implementation. The hardware specification provided in GOOGLE COLAB environment is [16]:

- GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM (most important, it is compatible with CUDA)
- CPU: 1xsingle core hyperthreaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)
- RAM: 12.6 GB Available
- Disk: 33 GB Available

As you can see here, the workflow of any machine learning algorithm is to have a problem statement that we saw earlier. Then to collect data. For this project, I received 29 original MRI images from the Radiology department in the DICOM format which is the standard format for medical images and in short for Digital Imaging and Communications in Medicine. I converted the DICOM images to jpeg and then performed data augmentation to get closer to an initial of 100 images. I used rotation by 45,90 and 180 and increased the training dataset to 75 images.

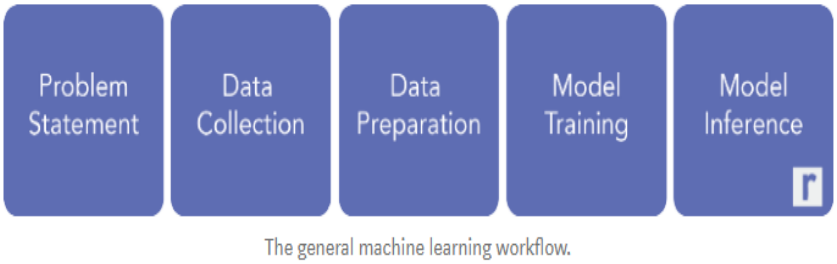


Figure 5: Machine Learning Workflow

The following steps will guide you through the implementation:[17]

5.1 Problem Statement

Locate and draw a bounding box over the region having liver in MRI images using YOLO v3. Since the MRI images are Protected Health Information, I have not uploaded MRI images despite them being de-identified in my git repository. A similar approach has been adopted to perform object detection for man and cycle with two custom classes. The image data and files related to this dataset are included in the git repository. `https://github.com/MahaSundaresan/CustomYolov3_MRI/tree/master`

5.2 Data Collection

The liver images were shared by the Radiology department at UW Madison. The image dataset with man and cycle was shared as a part of an assignment in CS766 Computer Vision. Augmentation of the available image dataset by performing rotation by 45 degrees, 90 degrees, 180 degrees, and addition of Gaussian noise is done to get at least 100 images to begin the training. Refer the

data_augmentation.ipynb `https://github.com/MahaSundaresan/CustomYolov3_MRI/tree/master` to implement this step.

Note: Download the Walking person folder and run the notebook. Else use your image dataset folder and change the folder path in the notebook (Line 42 and 73) before execution. You can generate as many images as you want by changing the number on Line 44 (Set as 25 at present).

5.3 Data Preparation

- Step 1: The data should be labeled in the Darknet format. I used LabelImg `https://github.com/tzutalin/labelImg` to annotate the data. In the Darknet format, for each image file (.jpg) there will be a corresponding label file(.txt). This also means that there is no label file for an image with no object in it. The label file has the following specifications:

  1. Each row corresponds to one object.
  2. The first number corresponds to the class. The class numbers are zero-indexed (starts from 0). That is, 0 corresponds to man and 1 corresponds to cycle.
  3. The following 4 numbers are the normalized box coordinates ranging between 0 and 1. It indicates x_center, y_center, width, and height. (If the boxes are in pixels, then x_center and width are divided by image width and y_center and height are divided by image height).

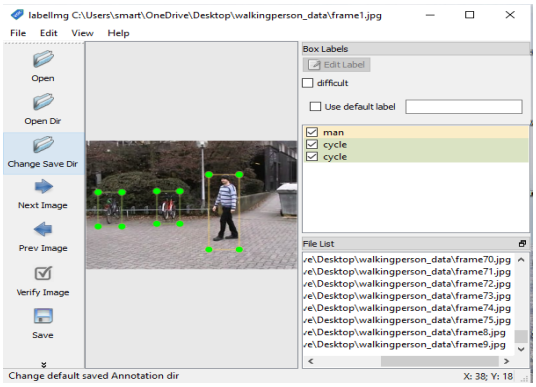


Figure 6: Working on LabelImg

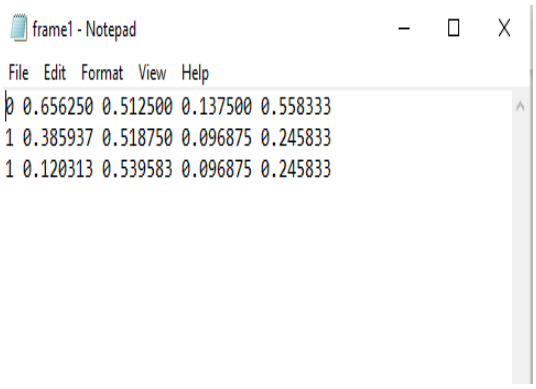


Figure 7: frame1.txt

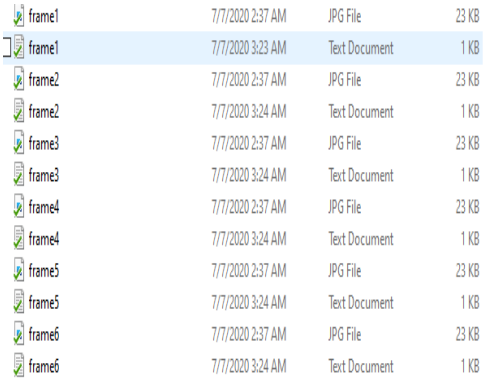


Figure 8: Corresponding .jpg and .txt files

- Step 2: Creating train.txt and test.txt files. Each row in these files contain the path to the image.

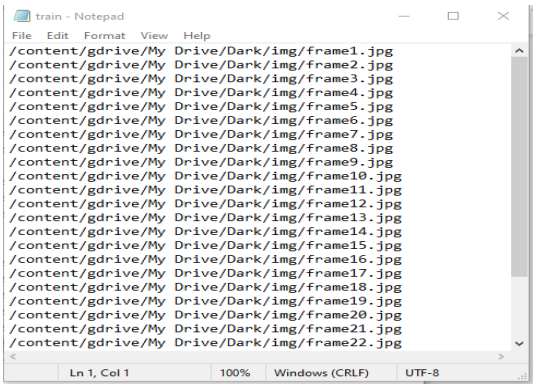


Figure 9: train.txt

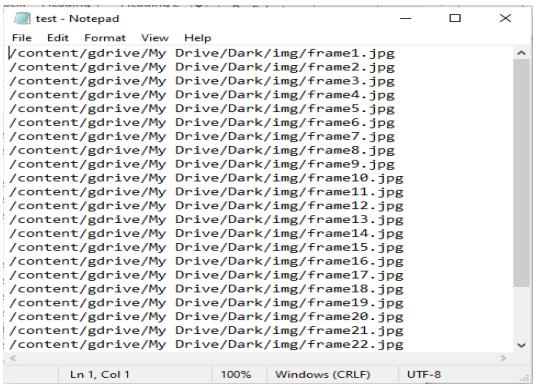


Figure 10: test.txt

- Step 3: Creating *.names file to list the class names for our custom data.

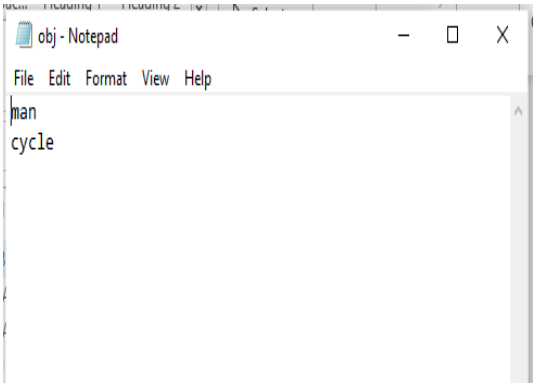


Figure 11: NAMES file

- Step 4: Creating *.data file with class count (2 corresponding to man and cycle),paths to train and test datasets (we use the same images twice here, but in practice you'll want to validate your results on a separate set of images), and the path to your *.names file.

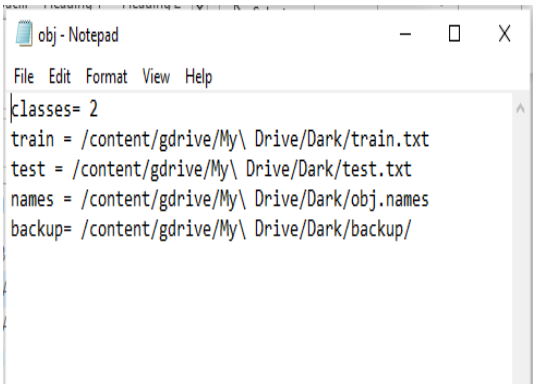


Figure 12: DATA file

- Step 5: Updating yolov3.cfg. By default each YOLO layer has 255 outputs: 85 values per anchor [4 box coordinates + 1 object confidence + 80 class confidences corresponding to the COCO dataset], times 3 anchors. Update the settings to filters= [5 + n] * 3 and classes=n, where n is your class count. That is, 21 for custom dataset with 2 classes and 18 for custom dataset with 1 class. Change this in all three YOLO layers of the .cfg file. Optional: Update hyperparameter values in the .cfg for better results.

5.4 Model Training and Inference

Create a folder "Dark" in your google drive.

1. Create the following folders:
   - backup (the weights generated during training will get updated here),
   - cuDNN and
   - bin
2. Download the following folders from my git repository `https://github.com/MahaSundaresan/CustomYolov3_MRI/tree/master` and place it in your local "Dark":
   - img
   - obj.data
   - obj.names
   - obj.txt
   - test.txt
   - train.txt
   - yolov3.cfg
3. Download `https://pjreddie.com/media/files/darknet53.conv.74` to start training the model initially. It is initial weights for training custom data used as pre-trained feature extractor (backbone) for YOLOv3. It excludes the weights of fully connected final layers which output the class probabilities for classification[18]. Place this file in "Dark".

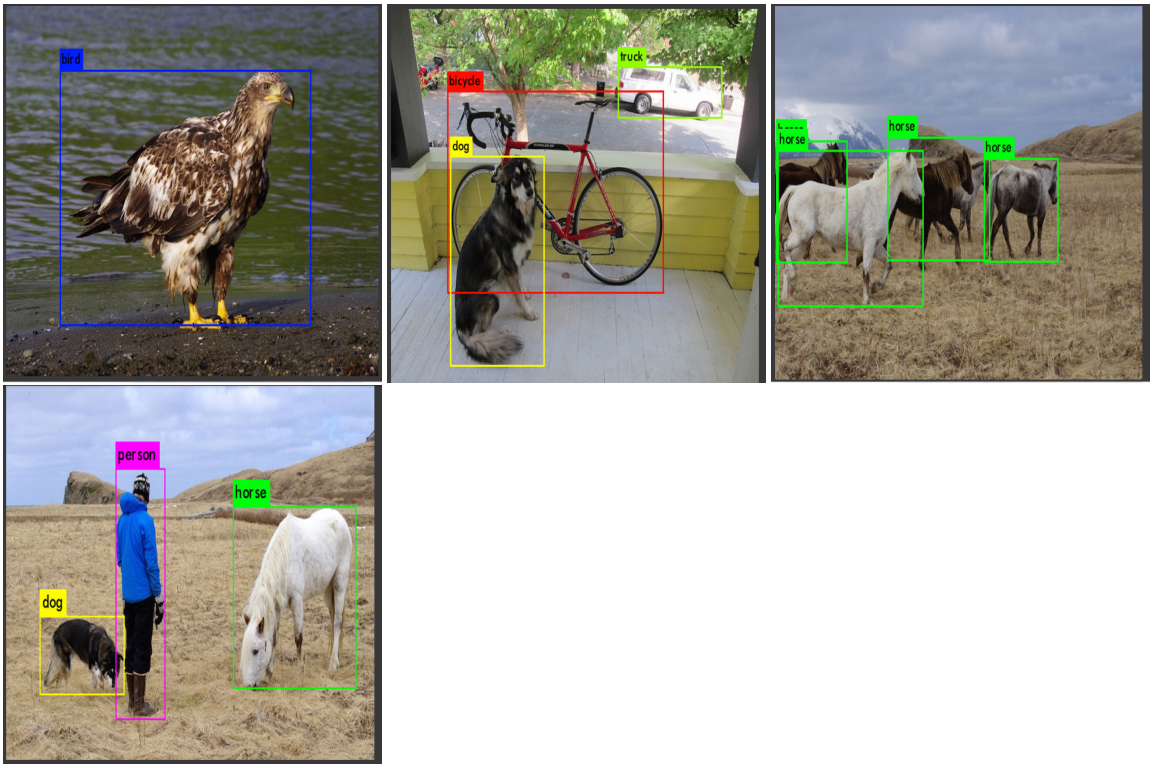
4. Now open the notebook from my git repository `https://github.com/MahaSundaresan/CustomYolov3_MRI/tree/master` and follow the instructions to execute the code and start training.

5.5 Result

We see the predicted image, mAP calculation after every 1000 iterations and a chart indicating average loss, mAP and iterations.

6 Results

1. COCO2014 dataset:



2. Custom dataset with man and cycle:
    - Precision: 59%
    - Recall: 83%
    - Average precision for man: 93.44%
    - Average precision for cycle: 69.65%
    - mAP: 81.54%

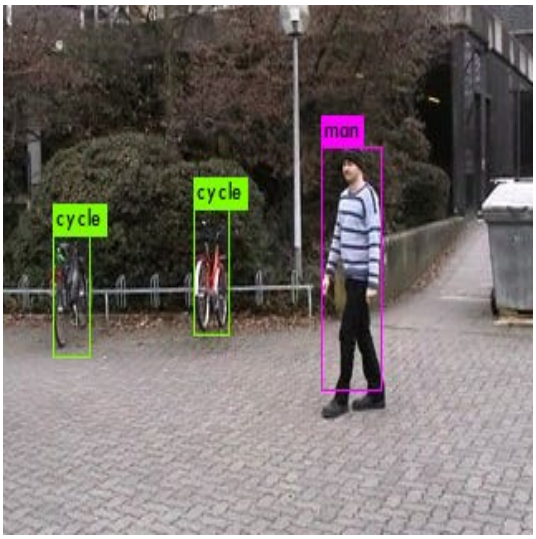


Figure 13: Predicted image with bounding box

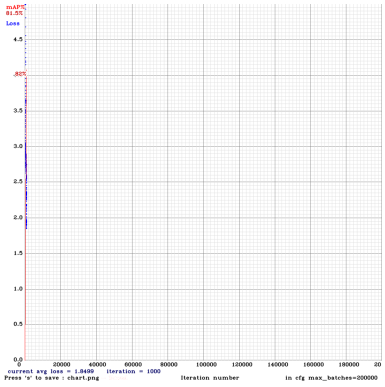


Figure 14: Chart

```
Cannot load image
28Label file name is too short:
Can't open label file. (This can be normal only if you use MSCOCO):
Label file name is too short:
Can't open label file. (This can be normal only if you use MSCOCO):

 detections_count = 467, unique_truth_count = 75
class_id = 0, name = man, ap = 93.44%           (TP = 25, FP = 22)
class_id = 1, name = cycle, ap = 69.65%         (TP = 37, FP = 21)

 for thresh = 0.25, precision = 0.59, recall = 0.83, F1-score = 0.69
 for thresh = 0.25, TP = 62, FP = 43, FN = 13, average IoU = 36.41 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.815427, or 81.54 %
Total Detection Time: 1.000000 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

 mean_average_precision (mAP@0.5) = 0.815427
Saving weights to /content/gdrive/My Drive/Dark/backup//yolov3_1000.weights
Saving weights to /content/gdrive/My Drive/Dark/backup//yolov3_last.weights
Resizing
512 x 512
```

Figure 15: mAP

3. Liver dataset

- Precision: 91%
- Recall: 80%
- mAP: 86.87%

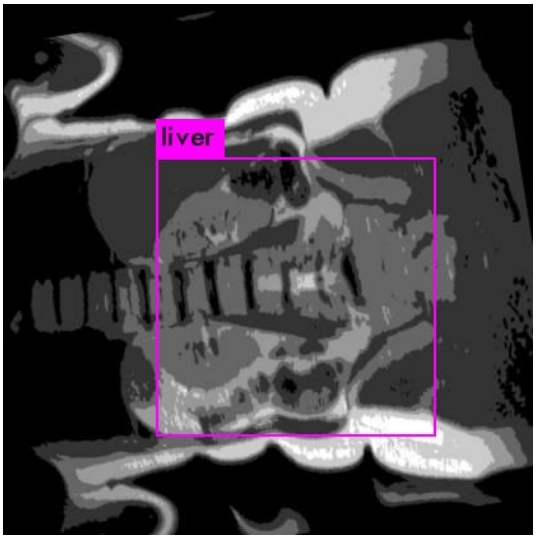


Figure 16: Predicted liver in Coronal view

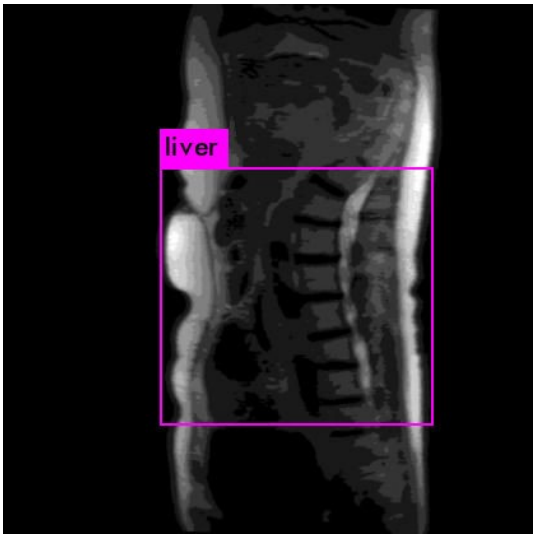


Figure 17: Predicted liver in Sagittal view

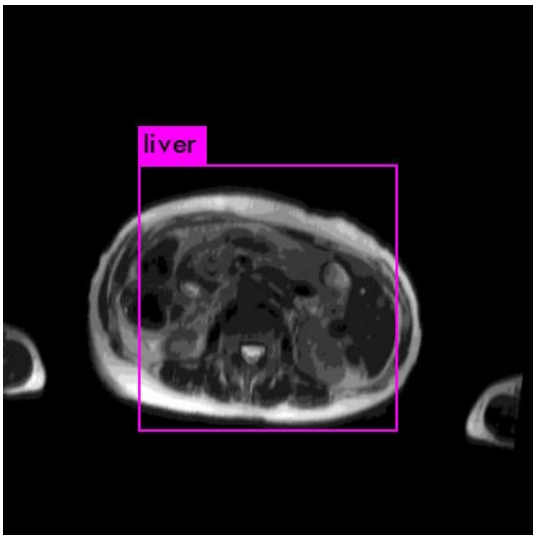


Figure 18: Predicted liver in Axial view

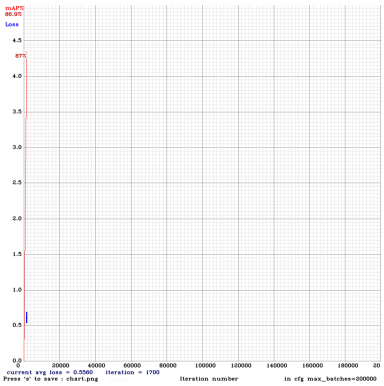


Figure 19: Chart

```
 1700: 0.557239, 0.556000 avg loss, 0.001000 rate, 1.437954 seconds, 108800 images
Resizing to initial size: 608 x 608
 try to allocate additional workspace_size = 120.51 MB
 CUDA allocate done!
valid: Using default 'data/train.txt'
valid: Using default '/content/gdrive/My Drive/drknet/train.txt'

 calculation mAP (mean average precision)...
76
 detections_count = 166, unique_truth_count = 75
class_id = 0, name = liver, ap = 86.87%          (TP = 60, FP = 6)

 for thresh = 0.25, precision = 0.91, recall = 0.80, F1-score = 0.85
 for thresh = 0.25, TP = 60, FP = 6, FN = 15, average IoU = 61.32 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.868711, or 86.87 %
Total Detection Time: 1.000000 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Figure 20: mAP

7 Further Steps

1. Train on more data without having to perform augmentation. Since in the real-world scenario, there cannot be an MRI image with rotations by varying degrees.

2. Further extensions of the problem could be in terms of identifying and quantifying the fat or lesions present in the liver.

3. Learning about MRI physics and clinical imaging protocol could be helpful in understanding the problem faced by radiologists better and in turn produce more desirable results.

8 Conclusion

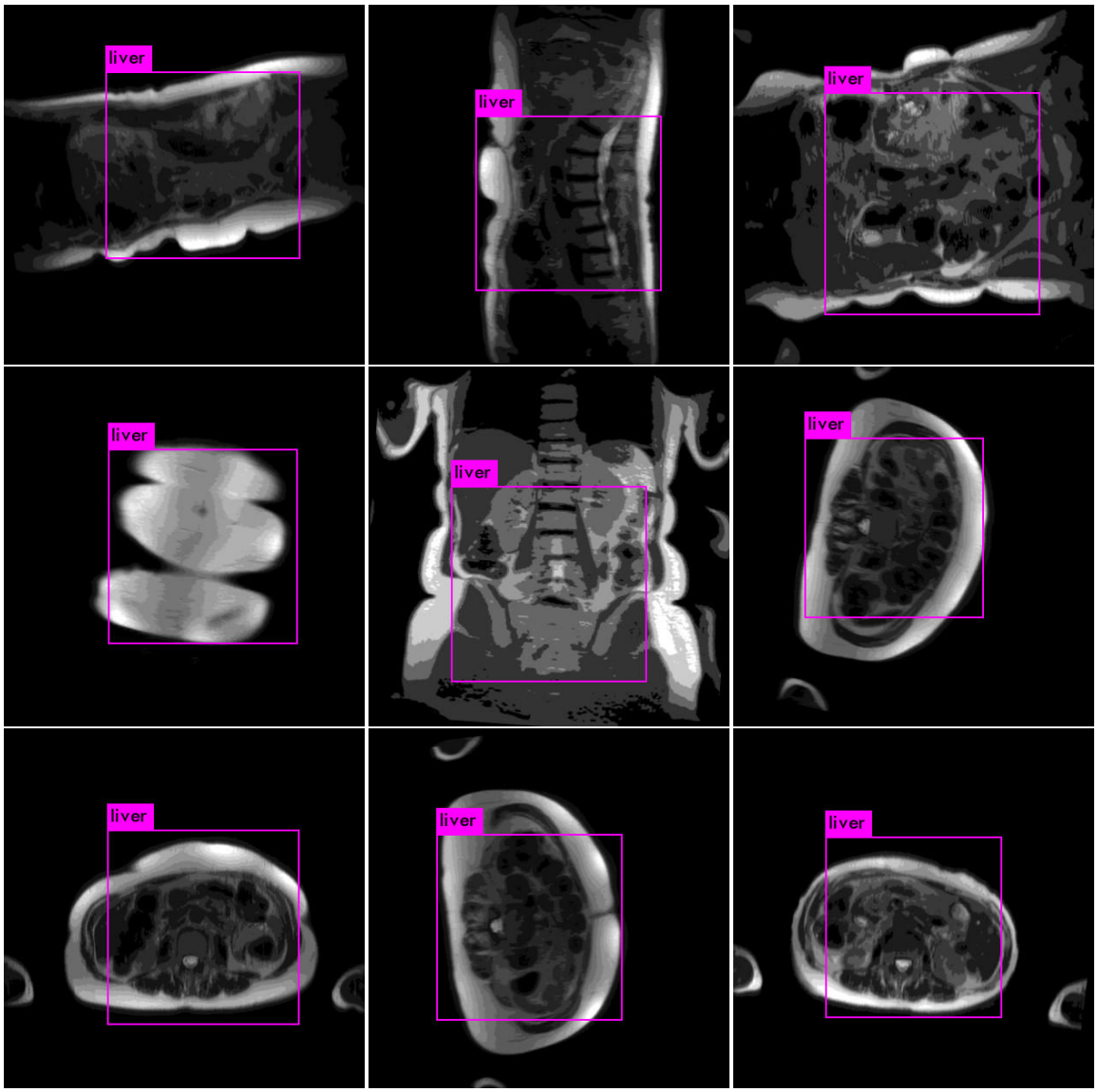
To summarize the project, identifying the location of the (liver) and drawing a bounding box around it can help in significantly reducing the time consumed during MRI examinations. This can also help the emerging rapid imaging protocols such as the rapid fat and iron quantification protocol developed here at the University of Wisconsin [19].

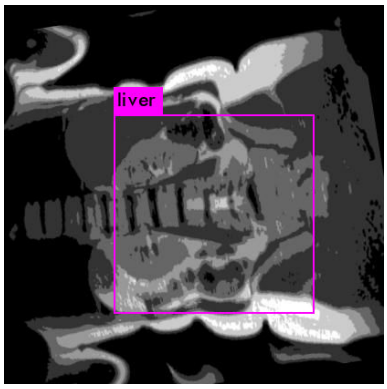
9 Acknowledgements

I would like to thank Dr. Diego Hernando from the Department of Radiology at UW Madison for sharing this problem statement and including me in his team. It was my pleasure to interact and learn from Dr. Scott Reeder, Dr. Jitka Starekova, Ruiqi Geng, and David T Harris. Ruiqi Geng helped in sharing the DICOM MRI images by de-identifying them. I also thank Dr. Matthew Malloy for offering a lot of support and guidance throughout the implementation of this project.

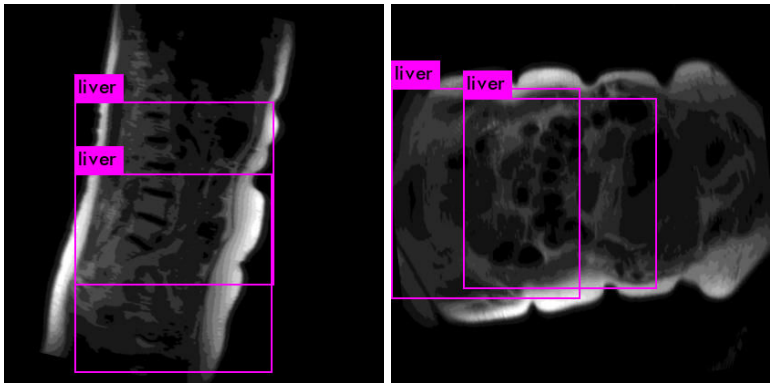
10 Additional Results

10.1 Correct predictions





10.2 Incorrect predictions



References

[1] Kiran Gangahdar, Deepa Santhosh, and Kedar N Chintapalli. Mri evaluation of masses in the noncirrhotic liver.

[2] James P Earls. Comparison studies of ct and mri in patients with hepatic metastases. *Magn Reson Imaging*, 7:1040–1047, 1997.

[3] American College of Radiology (ACR) Radiological Society of North America (RSNA). Magnetic resonance imaging (mri) safety. `https://www.radiologyinfo.org/en/info.cfm?pg=safety-mr`, February 2019.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[5] Jason Brownlee. A gentle introduction to object recognition with deep learning. `https://machinelearningmastery.com/object-recognition-with-deep-learning/`, 2019 (accessed August 14, 2020).

[6] Ross Girshick. Fast r-cnn, 2015.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[10] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[12] Rahmad Sadli. The beginner's guide to implementing yolov3 in tensorflow 2.0 (part-1). `https://mc.ai/the-beginners-guide-to-implementing-yolo-v3-in-tensorflow-2-0-part-1/`, 2020 (accessed August 14, 2020).

[13] Shanchen Pang, Tong Ding, Sibo Qiao, Fan Meng, Shuo Wang, Pibao Li, and Xun Wang. A novel yolov3-arch model for identifying cholelithiasis and classifying gallstones on ct images. *PloS one*, 14(6):e0217647, 2019.

[14] Ayoosh Kathuria. What's new in yolo v3? `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`, 2018 (accessed August 14, 2020).

[15] Unknown. What does the notation map@[.5:.95] mean? `https://datascience.stackexchange.com/questions/16797/what-does-the-notation-map-5-95-mean`, 2017 (accessed August 14, 2020).

[16] Quang Nguyen. How to train yolov3 on google colab to detect custom objects (e.g: Gun detection). `https://medium.com/@quangnhatnguyenle/how-to-train-yolov3-on-google-colab-to-detect-custom-objects-e-g-gun-detection-d3a1ee` 2019 (accessed August 14, 2020).

[17] Glenn Jocher. Train custom data. `https://github.com/ultralytics/yolov3/wiki/Train-Custom-Data`, 2020 (accessed August 14, 2020).

[18] David MacLeod. yolov3.weight and darknet53.conv.74. `https://www.gitmemory.com/issue/eriklindernoren/PyTorch-YOLOv3/201/503310806`, 2020 (accessed August 14, 2020).

[19] B Dustin Pooler, Diego Hernando, and Scott B Reeder. Clinical implementation of a focused mri protocol for hepatic fat and iron quantification. *American Journal of Roentgenology*, 213(1):90–95, 2019.