Computer Assisted Image Analysis II
Course 1TD398
Project Report for Breakfast Analysis on 2-D Image by Team 10

March 8, 2020

1. Subject Presentation

1.1. Project Task

Food Detection on 2-D plate images using the YOLO object detection algorithm.

1.2. Motivation

To estimate calories contents and nutrients after food detection of food and record the user's eating habits for a study of the metabolic process used by medical doctors.

1.3. Literature Survey

1. An Image Processing Approach for Calorie Intake Measurement - In this paper, color rasterization is performed with a 4^{th} level pyramid. This algorithm helped them with the illumination effect over the image and used the nearest neighbor evaluation with the 8X8 matrix. For the detection, they have used Support Vector Machines.[7]
2. Recognition and Volume Estimation of Food Intake using a Mobile Device - They have used a pairwise classifier, which is a combination of color and texture features at different scales called AdaBoost, to segment the food from the background. implemented Support Vector Machines for the detection of the food.[4]
3. Food Detection on Plate Based on the HSV Color Model - They have developed a model to identify the objects on the plate based on its hue value.[6]

The data set considered for the above projects are non-processed food items (Examples. Eggs, Cabbage, Vegetables) and not processed food items (Examples. Pizza, Burger, Rice). They use an Image processing approach to segment the food from the plate and use the Support Vector Machine algorithm to detect which are ideal for the non-processed food items.

1.4. Prerequisites

(a) Darknet - An open-source neural network framework written in C language by Joseph Redmon for testing and training computer vision models..[5]

(b) GPU (Graphics Processing Unit) - A graphics GPU is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.[9]

(c) CUDA (Compute Unified Device Architecture) - It is a parallel computing platform and application programming interface (API) model created by Nvidia. Darknet uses CUDA technology, which makes it fast and allows it to compute on a GPU.

(d) Google Colaboratory - Google Colaboratory enable us to write and execute Python in our browser with free access to GPU.

2. Method

YOLO(You Only Look Once) is a network that uses a Deep Learning algorithm to classify where the object is present and identity it within the image. This model is on the darknet framework, which is an open-source neural network framework for training and testing computer vision models.

The Architecture of YOLO is split into three main parts:

1. The Predictions Vector
   The input image given to the algorithm is divided into SXS grids that are used to predict the object in the network and use this information to classify the object. Each object will have one grid responsible for predicting, which is the center of the object. Each of these grids produces B bounding boxes and C confidence.The bounding boxes have five components:*x,y,w,h,confidence*.The *(x,y)* determines the center of the bounding box, which is relative to the grid.the *(w,h)* represents the width and height of the bounding box.all the values have been normalized to [0,1].the C confidence represents the probability of the existence of the object in the bounding box.1 indicates about how the bounding box is shown.[1]

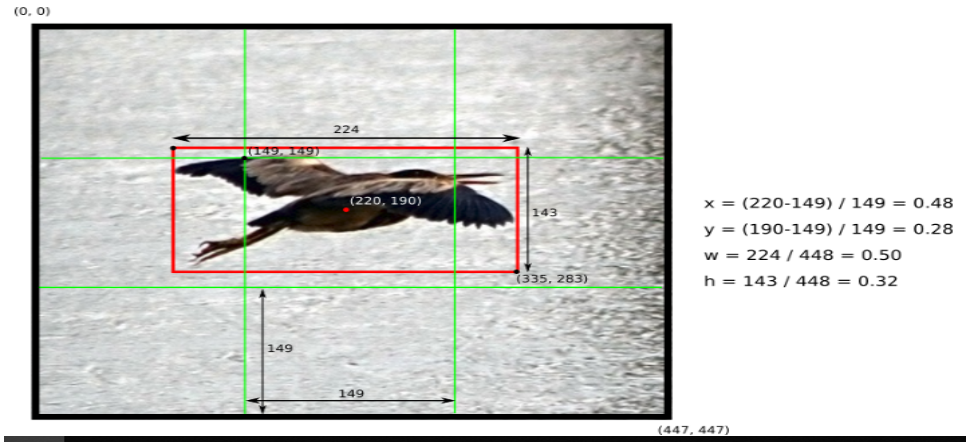


Figure 1: Bounding Box

2. The Network
   The YOLO network is structured like a regular Convolution Neural Network. It consists of 24 convolution layers and 4 Max-Pooling layers, followed by two fully connected layers.

3. The Loss Function
   For us to choose one bounding box to predict the object in the grids since the YOLO algorithm predicts multiple grids for each of the objects. We use loss function to compute the loss for each true positive. it selects the highest intersection over union(IOU) with the ground truth[3] the IOU is a metric used to measure the accuracy of the object detector in a particular data set.

The architecture above is the general architecture on the working of YOLO algorithm.there are different versions of YOLO that have minor changes in the neural network.YOLOv1 didn't do well when there two objects are close, which means when one grid contains the center points of two different objects, they introduced anchor boxes in the second version of YOLO, which allows one grid to detect multiple objects. I have used tiny-YOLOv3 for my project that is currently the quickest to detect. Still, accuracy is very low compared to other algorithms.this is a light version of YOLO having the anchor boxes and requires less time to train compared to the original YOLOv3. Due to the unavailability of resources and time, I have used tiny-YOLOv3 instead of YOLOv3 for my project.

3. IMPLEMENTATION

3.1. Image Dataset

The model is trained on five food items collected from OIDv4 Toolkit that collects images from Open Images Dataset V4.[8] We get various kinds of image dataset which includes segmentation type and detection type. But, for our model, we need images that contain detection type for the bounding boxes(BB). All the images contain a bounding box label text file downloaded along with the images. the following are the dataset food items considered for this model along with the number of images for each class:

1. Pancake - 362 images
2. Milk - 156 images
3. Apple - 400 images
4. Bread - 400 images
5. Cheese - 398 images

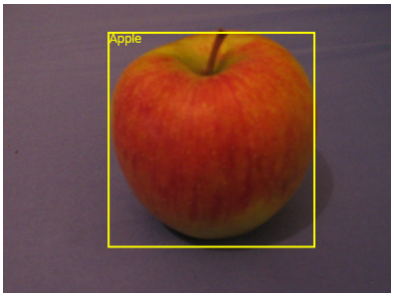


Figure 2: Dataset example

3.2. Produce the Training and Text files

The YOLO algorithm takes the image input in a YOLO forma. Each of the dataset images will have one text file each which has the following composition:
LABELID XCENTER YCENTER WIDTH HEIGHT shown in 2.

90% of the data is used for training and 10% is used for testing the model.this is implemented using process.py that divides the dataset equally based on the class.

File Edit Format View Help
0 0.40734265734265734 0.08166666666666667 0.22727272727272727 0.11666666666666667

Figure 3: YOLO Data BB text file

3.3. Produce the Anchor file

The significance of anchor is been mentioned in the architecture of YOLO.the problem is shown in figure 4 where the person and car are overlapping in the image. To overcome this, we use the anchor box technique where we can create a longer grid cell vector to associate multiple classes with each cell.

Figure 4: Anchor Boxes

3.4. Modifying the configuration file

The configuration file of the tiny-YOLOv3 is based on the COCO dataset, which contains 80 number of classes.we need to change the configuration file to train for our custom data.the changes are as follow:

1. Change the number of classes
2. Update the filter in each of the layers based on the formula.

$$filters = (classes + 5) * 3$$

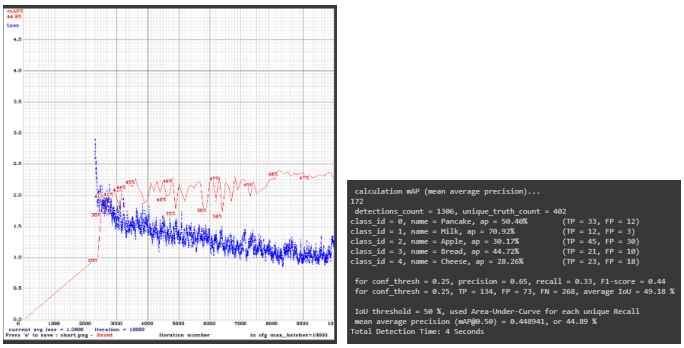
3. Update the Max_batches to number of classes * 1000 and steps to be 80% and 90% of Max_batches.

There are two more files which needs to be created called "object.names" that contains the number of classes, and the "object.data" that contains the location of configuration file,training and testing files for the model to run.Now we are ready to train our tiny-YOLOv3 model on our custom dataset.

4. RESULT

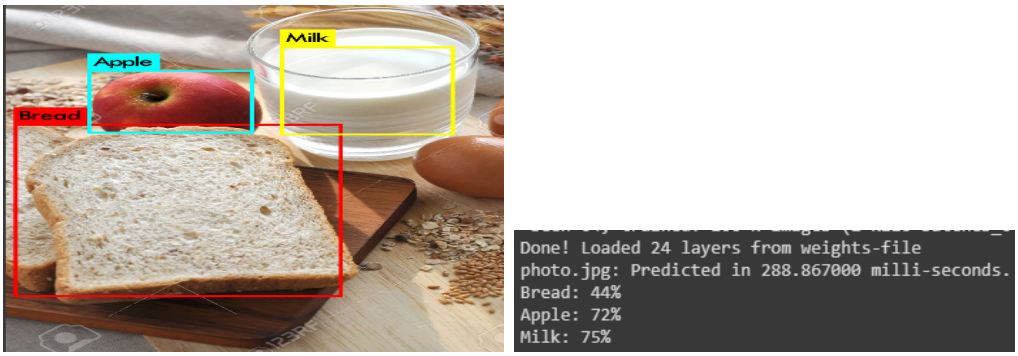
The training of the model takes a very long time. The 5a shows the whole process where the X-axis shows the number of epochs, and Y-Axis shows the loss rate. It is measured using a metric called mAP(Mean Average Precision). the mAP for object detection is the average of the average precision calculated for all the classes.[2] Average precision is a measure that combines recall and precision for the results.After training, the model saves the weights after every 1000 iterations to

backup folder. we get the best result when the weight was in 9000 weights where the mAP value is 44.89% for all the classes in the dataset.5b



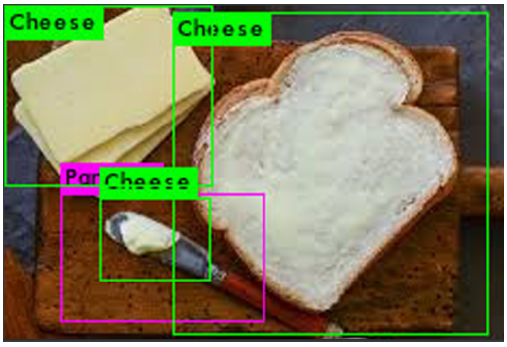
(a) Result of training data (b) Result on test data

The results of the trained model stored in tiny_YOLOv3_best. Weights which has an mAP value of 44.89%.Used on the test images and results are shown in figures 8.the time taken for the detection is very less, which is one of the key features of the tiny-YOLOv3 model. The model can find the food items precisely when kept apart and have a proper view of the picture. 6a shows the result of food item detection of bread, apple, and milk.the confidence level of each of the detection seen in 6b. Our model did not do well for the test cases in 7a and 8a when the food items have overlapped. Cheese is overlapping on the bread, and it is not able to detect bread because of this. Since our model accuracy is not great, it also misclassifies the food item. The detection of pancakes seen in the 7b because of the color of the background, which is similar to the pancake. our model detects only two pancakes out of five pancakes, seen from the side view, and the confidence level illustrated in 8a.



(a) Result 1 (b) Prediction of elements in Result 1

Figure 6: Results of Test Image-1 consisting of Bread,Milk and Apple

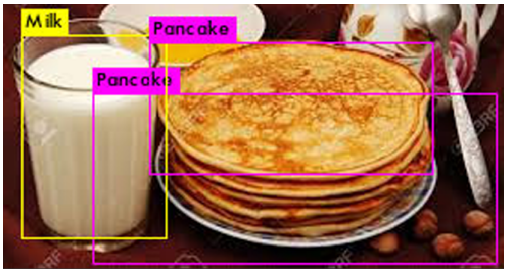


(a) Result 2

```
photo1.jpg: Predicted in 296.900000 milli-seconds.
Cheese: 71%
Pancake: 12%
Cheese: 38%
Cheese: 39%
```

(b) Prediction of elements in Result 2

Figure 7: Results of Test Image-2 consisting of Bread and Cheese



(a) Result 3

```
photo2.jpg: Predicted in 294.661000 milli-seconds.
Milk: 52%
Pancake: 32%
Pancake: 80%
```

(b) Prediction of elements in Result 2

Figure 8: Results of Test Image-3 consisting of Pancakes and Milk

5. DISCUSSION

The primary focus of this study is to detect the food item using a deep learning technique, accomplished using the YOLO object detection algorithm. The accuracy of the Tiny-YOLOv3 is not good but is significantly fast in detecting. Few factors for low accuracy are that the number of data samples of each class is not sufficient and the resolution of the pictures. The implementation of other versions of the YOLO object detection algorithm would have increased the accuracy. Still, there is a constant trade-off for speed in lightweight models and accuracy in larger models.I learned about how the YOLO object detection works and the architecture behind it. The implementation of it is easy, but the challenging part is prepossessing to convert the data into YOLO format.

References

[1] Hackernoon. *Understanding YOLO*. URL: https://hackernoon.com/understanding-yolo-f5a74bbc7967.

[2] Medium. *Breaking Down Mean Average Precision (mAP)*. URL: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.

[3] MissingLink.ai. *YOLO Deep Learning: Donť Think Twice*. URL: https://missinglink.ai/guides/computer-vision/yolo-deep-learning-dont-think-twice/.

[4] Manika Puri. "(Recognition and Volume Estimation of Food Intake using a Mobile Device)". In: *IEEExplore* (2009).

[5] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[6] Smit Trambadia. "Food Detection on Plate Based on the HSV Color Model)". In: *IEEExplore* (2016).

[7] Gregorio Villalobos. "An Image Procesing Approach for Calorie Intake Measurement)". In: *ResearchGate* (2012). DOI: DOI:10.1109/MeMeA.2012.62266364.

[8] Angelo Vittorio. *Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset*. https://github.com/EscVM/OIDv4_ToolKit. 2018.

[9] Wikipedia contributors. *LaTeX — Wikipedia, The Free Encyclopedia*. 2011. URL: https://en.wikipedia.org/wiki/Graphics_processing_unit.