Grocery Product Detection

Srinithish Kandagadla, Amit Makashir, Gurjaspal Singh Bedi
School of Informatics, Computing and Engineering

Abstract

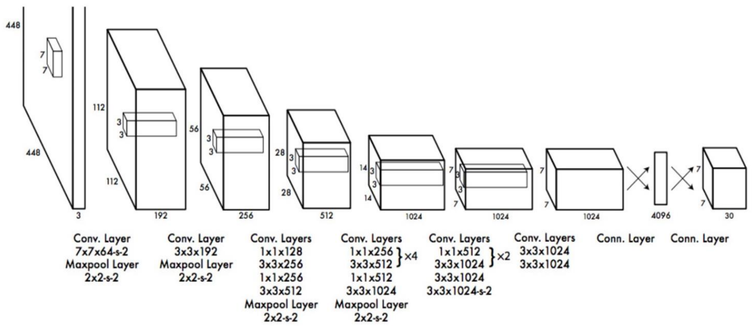
The billing process at the big stores like Kroger and Walmart is time consuming and cumbersome. We want to make that process easy and frictionless. Taking just a picture of objects would identify the products in the image and their corresponding quantities which would help in faster and more convenient checkout.

Our work and Future scope

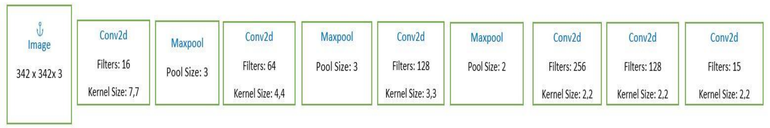
We tried to build an end-to-end system based on YOLO but on a smaller scale without using any APIs. Our intention was to tweak the system specifically for our use case.

We intend to scale this project to include large scale of products with better accuracy. To be useful in a real world setting, this network must be deployed to a smaller device (eg: mobile phone). So our future scope would also include, compressing the network.

Original YOLO Network



Network Architecture



Methods and Data

DATA: We clicked 180 images of the grocery products and then augmented them by rotating and resizing (342 x 342). The final count of the images after the data augmentation is 720. The images contain 10 different products in different possible combinations. We created the bounding boxes for all the products in the each image using Labelimg.

METHOD: We are using "You Only Look Once" (YOLO) for object detection but with smaller network. Since the scope of this project is only to detect grocery products, hence we are using a smaller network. However, this can be scaled to detect larger classes of items by modifying the network and training with the relevant data.

YOLO METHOD:
We divide each image into 19 x 19 grid and each cell in the grid is responsible for detecting exactly one product and its bounding box. To make this possible, each grid has the following parameters.

a. **Probability of Objectness:** Probability of object being present in that cell.
b. **Bounding box dimensions:** The center (x, y) and width (w) and height (h) of the bounding box, normalized with image size.
c. **Probability of different product classes:** Probability of the product.

Since we had 10 different classes of products in total and we had created a 19 x 19 grid, our target prediction is 19 x 19 x 15 (1 + 4+ 10)

Non Max Suppression: As each cell will certainly predict a bounding box and the class we need to eliminate less probable and duplicate predictions. First, we remove bounding boxes which have probabilities below certain threshold. Second, we suppress the bounding boxes based on Intersection over Union (IOU) threshold and class label (different than the standard YOLO)

Confidence Loss:

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

Localization Loss:

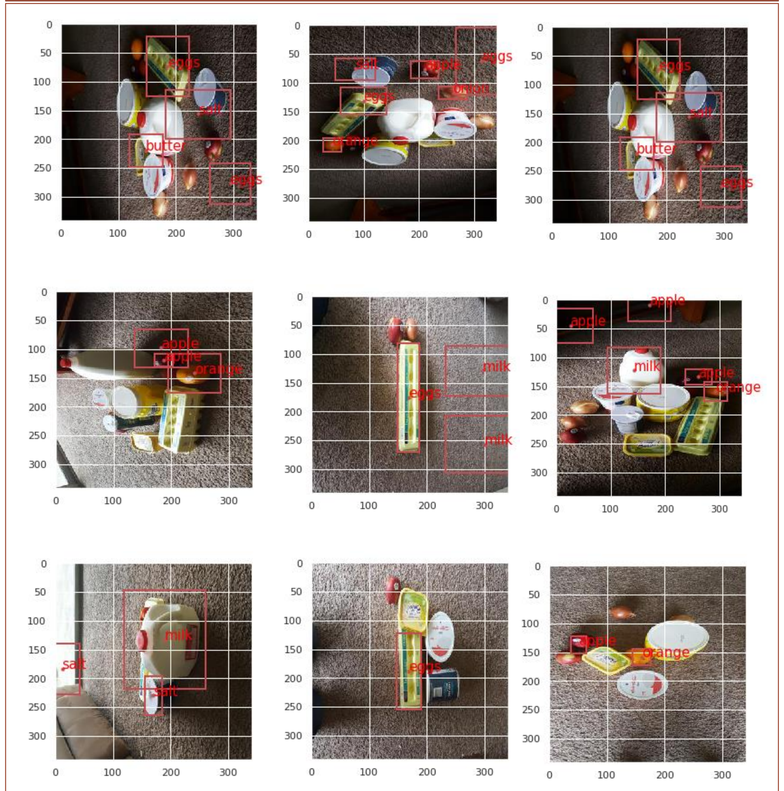
$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2$$

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

Classification Loss:

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

Results



Conclusion

1. When there was single object, model was able to predict the bounding boxes perfectly.
2. One of our objective was to detect obscured or overlapping products, in few examples model was able to do that.
3. Model was able to predict multiple products in single image.
4. Model was able to predict products in different orientation.
5. We used less number of convolution layers than the original YOLO.

Contact

Srinithish, Amit, Gurjaspal
Indiana University
Email: gbedi@iu.edu

References

1. You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
2. YOLO9000: Better, Faster, Stronger - Joseph Redmon
3. Data Augmentation for Object Detection- Ayoosh Kathuria
4. A Github repository by Experiencor (https://github.com/experiencor/keras-yolo2)