Tiny YOLO v3: Object Detection for Late Meal

Anabelle Chang
Princeton University
`anabelle@`

Christine Kwon
Princeton University
`cmkwon@`

Christy Lee
Princeton University
`christyl@`

Abstract

We intend to address the challenge of determining the total cost of all Late Meal food items present in a single frame by utilizing YOLO v3 (You Only Look Once). We train and evaluate on a data set of 15 classes and 2,178 images which were compiled through online searches and personal photography and then tagged manually. We find a generally high level of precision across the various object classes and analyze factors which affect our custom trained Tiny YOLO v3 model, including the number of object classes, the number of epochs used during training, and various inconsistencies in the physical appearances of products.

1 Introduction

At Princeton University, undergraduate class schedules often conflict with designated dining hall hours for lunch and dinner; this leaves students scrambling for meals at odd times or skipping them entirely. In order to combat this issue, the university implemented a policy called ”Late Meal” in which students enrolled in the meal plan are allowed items from the Frist Food Gallery under a certain monetary limit, free of charge, between the hours of 2:00 pm through 3:45 pm and 8:30 pm through 10:00 pm on weekdays. Beyond the practical meal aspect, Late Meal has evolved into an integral element of the social scene for freshmen and sophomores at Princeton University: there is a universal congregation of underclassmen for food and snacks at each of the aforementioned times. As such, Late Meal plays a large role in the daily lives of Princeton students, which makes this project of supporting the Late Meal experience a practical application of computer vision for a large target audience.

Every Princeton University student has experienced the challenge of keeping their Late Meal purchase below the designated price limit. The obscure price tags of many items make it very difficult to determine the price of each item, and thus difficult to calculate the total price of one's purchase. Late meal is limited to a total value of \$6.95 for lunch and \$7.95 for dinner. However, when picking out items, it is easy to go over these limits. This leads students to either unexpectedly pay extra money at the register or to even leave the checkout line to put items back on the shelves in order to avoid paying extra expenses out of pocket. The missing price tags are especially prevalent in the ”Grab-n-go” section of the Frist Gallery, where most items are not individually labeled with prices. Instead, a small price list sits on the side wall and displays item names and the corresponding prices. However, this price list often omits the price of many integral items and is not consistently updated. Moreover, when there is a rush of students in the area, it can be difficult to access and read this list.

Therefore, we are interested in implementing an object detector that utilizes YOLO v3, allows students to hold their phone camera towards their desired Late Meal items, and with live updates, outputs the cost of the detected item. At this point, they can choose to add the item to their cart, which keeps track of their running bill. We choose to work with object detection instead of object classification because the completed app should allow frames which contain multiple objects for student ease and efficiency. With this goal in mind, we utilize YOLO v3, or You Only Look Once, an object detection algorithm known for the speed and efficiency in its single convolutional neural network.

This exploration has a second purpose beyond its practical application: we aim to further understand the properties of YOLO v3 that make it useful in performing this task as well as the factors which influence its results in order to consider the advancement of new detection algorithms designed for this purpose. Specifically, we consider the potential difference in precision between models trained in different amounts of classes, the optimal number of epochs in training before precision begins to decrease, and the general robustness of the system to common inconsistencies such as color variation or shape distortion (i.e. when a chip bag is crumpled out of shape).

One of the challenges of this task lies in the fact that the data set we need for this task does not yet exist. Therefore, we are personally building and annotating the images which we selected. Of these images, the majority are taken from online searches, which means that most are clear photos of objects posed in a consistent way. With real student use, however, depending on where and from what angle the photo is taken, the object may appear quite distorted or unclear based on its positioning within the frame; thus, our training set may not be fully indicative of what the detection model will see in practical use. We hope to make our detector as robust to occlusion, noise, and distortion as possible.

2 Related Work

Initially, this project was inspired by currently existing applications of computer vision in similar food retail environments. This includes the novel Amazon Go[1] stores which allow shoppers to simply grab the items they need and then leave, automatically sending a receipt of their purchase to their Amazon account. The use of computer vision is a significant part of the A.I. architecture which allows Amazon Go shoppers to skip the checkout line. Similarly, we were also inspired by the 85° C Bakery which partnered with Viscovery, a Vision Artificial Intelligence lab, in order to create an automatic checkout process which scans a tray to calculate a customer's total bill, no employee necessary. Beyond the realm of food retail, applications which use computer vision for general retail are becoming popularized as well.

Despite the rising popularity of computer vision applications in every day life, as well as the known efficacy of YOLO v3 in the field of real time object detection, there is not too much research which currently exists on training YOLO v3 on custom data sets in an optimal manner. Some related research which has been done includes Yi, et al.'s paper "An improved tiny-yolov3 pedestrian detection algorithm"[8], which adapts Tiny YOLO v3 with k-means clustering in their training set in order to train a more optimal pedestrian detector. Similarly, we aim to optimize the training of YOLO v3 in order to most effectively classify our classes of objects.

[1]https://www.amazon.com/b?ie=UTF8&node=16008589011

3 Methods

3.1 Data Accumulation

As mentioned previously, one of the largest hurdles we faced when it came to implementation was that there is no currently existing image data set which contains the specific objects we needed. The object selection comes directly from the group of packaged food and drink products offered in the ”Grab-n-go” section of Princeton University’s Late Meal, and can be seen listed in Figure 1. Therefore, we needed to assemble our own data set.

In order to compile our data set for training and testing, we utilized python scripts written to scrape images from the web, a Google Chrome extension called Fatkun Batch Download in order to mass download and sort through images which result from Google Image searches, as well as manual photography in the Late Meal area. This careful curation of images allowed us to compile a unique data set that was specific to this project. Training the algorithm on a unique data set allowed us to create an object detector tailored for Princeton University students. Most of the images found on Google contained objects which appeared posed against solid color backgrounds, which we realized could be a problem because in real life applications of the detector, students would take photos in which objects can look imperfect. Thus, we attempted to combat this problem by mixing in photos we took in person from various angles, lighting, and orientations in order to train on a more realistic data set. Our data set consists 2,178 sample images total, distributed so that there were roughly 100 to 200 of each menu item. Though there was some discrepancy in number of training examples when images of certain products were more or less readily available, the images across each class were generally balanced.

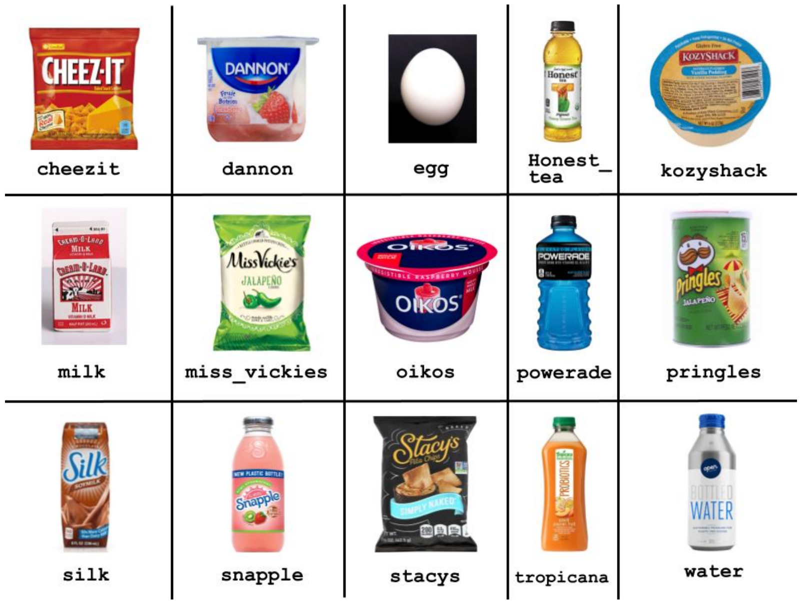


Figure 1: The fifteen object classes included in our training set.

3.2 Data Processing

We used a 90/10 ratio to split the data set into a training and a testing set, roughly of sizes 2,000 and 200 respectively. In order to prepare our images for training after downloading, we utilized Microsoft's VoTT, or Visual Object Tagging Tool, software to tag our objects. The bounding box was determined manually based on what we believed would be recognizable as a full item to the human eye. Items that were positioned incorrectly or largely cut off were not included. Each image was exported as a .jpeg format, and has an accompanying .txt file that indicates the bounding box of the object within the photo. Interestingly, the most recently released versions of VoTT did not support exporting the annotations in the format necessary for training in YOLO v3. As such, we used an earlier version of VoTT, an example frame of which can be seen in Figure 2. The final steps of data processing included removing any images which could not be used or any empty annotation text files which had been created in the process, as well as compiling all of the classes' images into two folders for the training and testing sets (in the aforementioned 90/10 split). To use our dataset with the YOLO v3 algorithm to build a custom detector, we then updated the configuration file to update the anchors based on the training images and number of filters based on the number of classes.

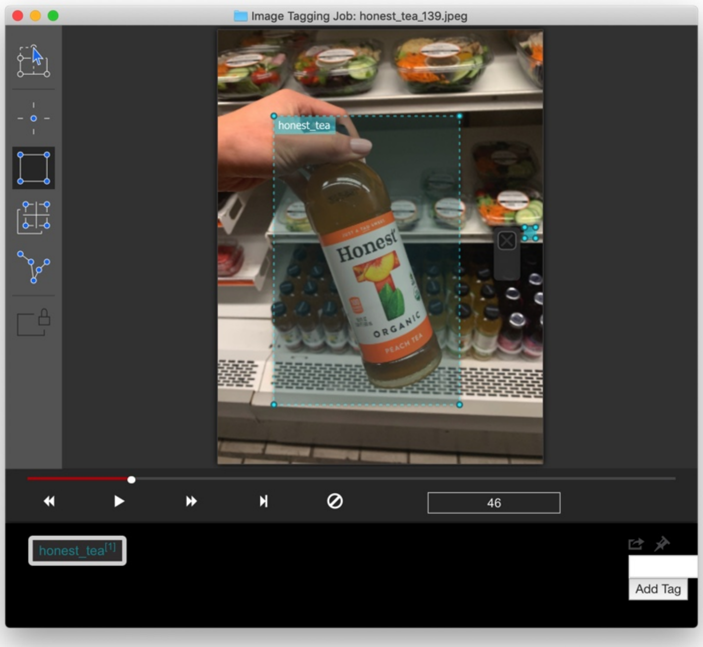


Figure 2: An example annotation in Microsoft's VoTT (Visual Object Tagging Tool) 1.7.2 software

3.3 YOLO v3

YOLO v3 is a real time object detection system with a Darknet-53 architecture, which is a 53-layer convolutional neural network.

YOLO v3 is able to perform more efficiently than many other classification and detection algorithms because it is a one stage target detection algorithm, meaning it only has to run once over an entire image. Multi-stage target detection algorithms iterate through multiple neural networks, thus increasing accuracy and precision but sacrificing speed and efficiency. Thus, the single stage allows the YOLO v3 detection algorithm to perform much faster and more efficiently than others used for similar purpose, although at the cost of slight loss of accuracy.

The algorithm initially resizes the image to 416 by 416 pixels and then divides the image into a 13x13 grid, in which each cell predicts 5 bounding boxes, or boxes that serve as candidate boxes that may enclose an object we want to detect, accompanied by a confidence score that indicates the likelihood of the bounding box containing a possible object. The bounding box is also represented by a probability distribution over all the possible classes specified during training. This probability distribution, in conjunction with the confidence score, determines the final probability, or score, that the bounding box contains a specific object class. The boxes with scores below the user-specified threshold are discarded.

We decided to utilize Tiny YOLO v3 over other target detection algorithms such as RetinaNet because despite a slight loss in accuracy, the one-stage target detection algorithm allows for much quicker detection. If this detector were to be utilized in an actual Late Meal setting, efficiency in detection is crucial, and Tiny YOLO v3 is ideal for real-time detection.

3.4 Evaluation Metrics

We use evaluation statistics that utilize the numbers of true/false negatives and positives. Specifically, we define:

$$\text{precision} = \frac{TP}{TP+FN}$$

$$\text{recall} = \frac{TP}{TP+FN}$$

$$\text{F-1 Score} = 2 * \frac{precision * recall}{precision + recall}$$

We also consider the intersection over union (IoU) value, a measure used in YOLO v3 defined as the area of intersection between the predicted bounding box and the actual bounding box divided by the area of union of the two boxes. When the prediction is nearly accurate, IoU will have a value close to 1. As the prediction gets less accurate, the IoU value gets smaller.

The primary evaluation metric we focus on is mean average precision (mAP), which is the standard for evaluating the effectiveness of detection systems. mAP in this case is the mean of all object classes' average precision (AP) values, which is the area under the precision-recall curve.

4 Results and Analysis

4.1 Single-Class vs. Multi-Class Detection

We wanted to examine how different number of classes affected detector performance. More specifically, we analyzed the difference in performance between a Tiny YOLO v3 model trained on a single class, for which we used the ”Snapple” class, and a separate model trained on all 15 classes present in our data set.

We expected that the single-class model would tend to over-predict and produce more false positive results. We reached this hypothesis because if there is only one class it knows how to predict, the detector would tend to try to detect as many objects of that class as possible. Moreover, we believe that a single-class model will predict its positive detections with higher levels of confidence (objectness scores). As can be seen in Figure 3 and Figure 4, the single-class detector falsely predicted a liquor bottle as a Snapple object with high levels of confidence whereas the multi-class detector did not make the false positive prediction. However, it is of note that this latter detector predicted the Snapple with significantly less confidence than did the single-class detector.

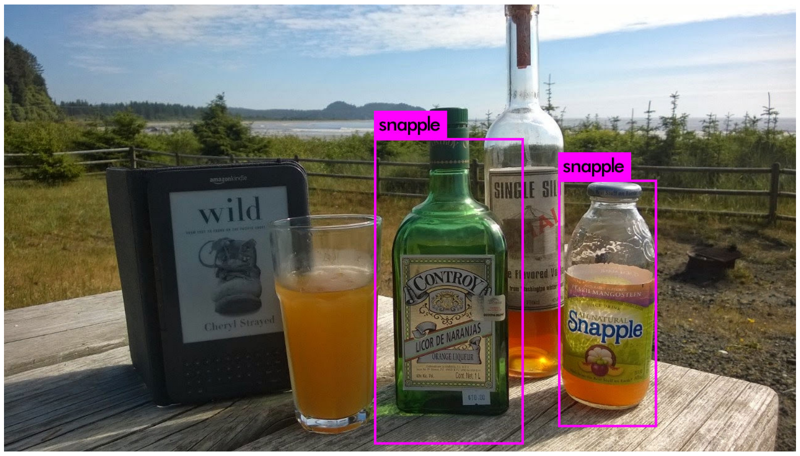


Figure 3: The items predicted by the single-class detector with objectness scores of 84%, and 70% under the label ”snapple”.

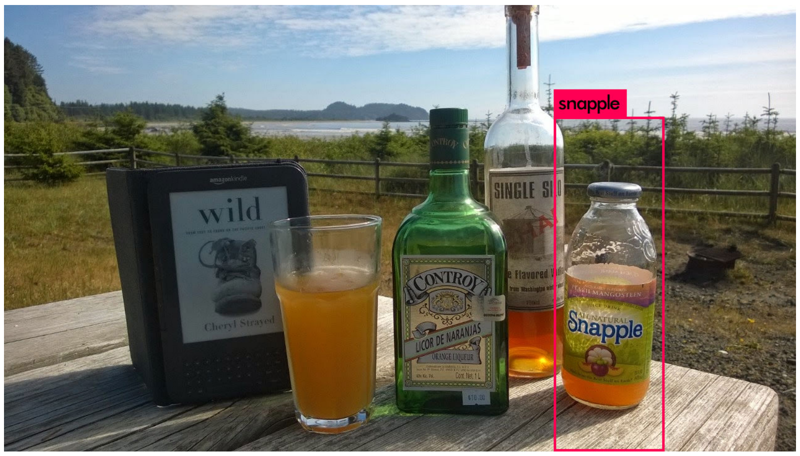


Figure 4: The items predicted by the multi-class detector with objectness scores of 15% under the label ”snapple”.

In contrast, we predicted that the multi-class model would produce more false negative results since there are more classes that the detector can be confused with. Furthermore, the multi-class model will generally produce lower confidence scores when predicting its positive detections. As can be observed in Figure 5, the single-class detector correctly detected 5 Snapple items in the image with mostly high objectness scores. In Figure 6, we can see how the multi-detector class faults by failing to recognize the leftmost Snapple as an item, and by misidentifying the rightmost Snapple as a Tropicana juice. Morever, the objectness scores of the multi-class detector are consistently lower than those of the single-class detector.

Though there is still additional testing to be done, all of our explorations in this realm were consistent with the aforementioned hypotheses.

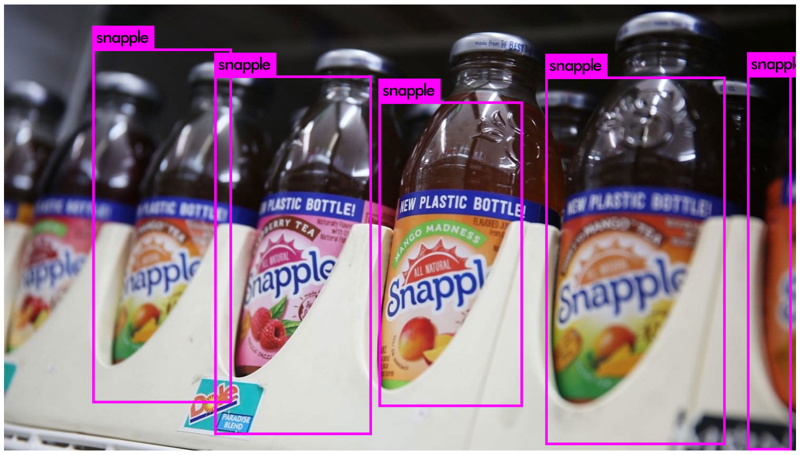


Figure 5: The items predicted by the single-class detector with objectness scores of 16%, 44%, 84%, 87%, and 26% under the label ”snapple”.

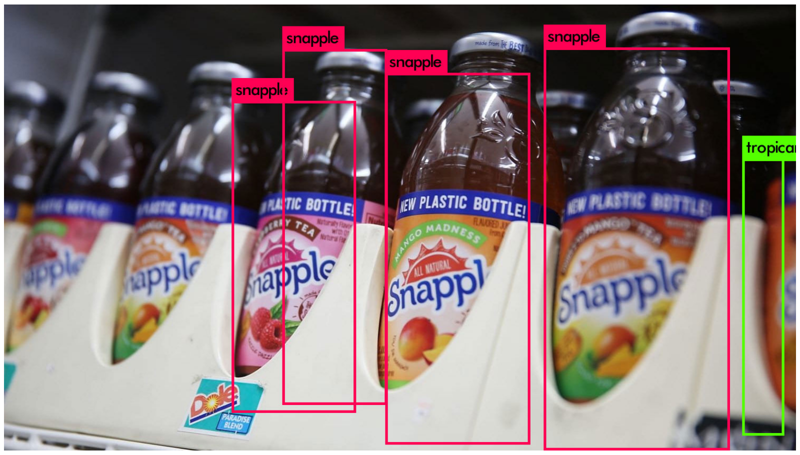


Figure 6: The items predicted by the multi-class detector with objectness scores of 32%, 62%, 25%, 28% under the label ”snapple” and 12% under the label ”tropicana_probiotics”.

4.2 Changes in Precision with Different Weights

Another aspect of the Tiny YOLO v3 model we wished to explore was the relationship between precision and the number of epochs used during training. In order to analyze this, we considered the weights calculated after 1000, 2000, and 3000 epochs for both the single class model and the original 15 class model. Results for the original model's average precision values can be seen at Figures 7, 8, and 9, while the three levels of precision for the single-object model can be seen at Figure 10.

In terms of our actual findings, we saw the expected increase in precision between the weights generated after 1000 epochs and after 2000 epochs for single-class model. This improvement results from continued training and thus a better model. However, this trend does not continue when the model uses the weights generated after 3000 epochs. Rather than benefiting from continued training, this model begins to over fit specifically to the training data, and precision begins to decrease, seen in Figure 11.

However, this trend was not evident in the multi-class model, whose precision increased between 2000 epochs and 3000 epochs, seen in Figure 11. This is most likely due to the larger amount of data and classes the model was trained on; it would require many more epochs to over fit for each class.

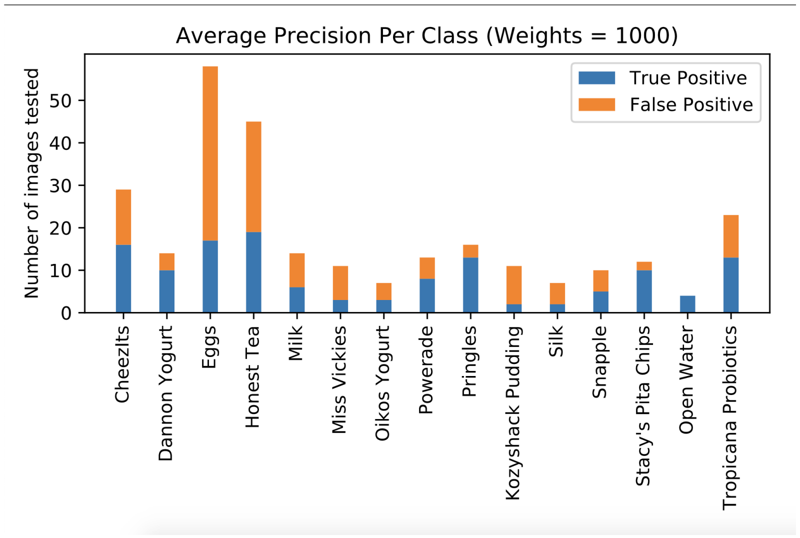


Figure 7: Average precision (AP) for a 15-class detector with weights calculated after 1000 epochs.

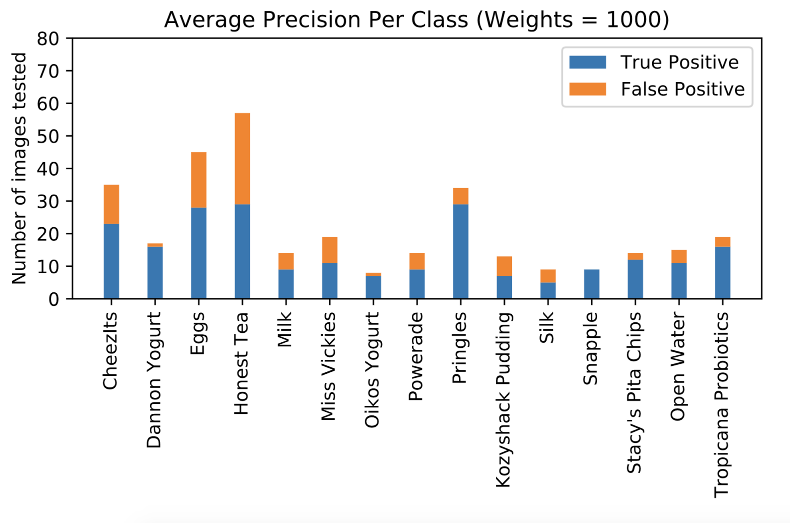


Figure 8: Average precision (AP) for a 15-class detector with weights calculated after 2000 epochs.

4.3 Robustness to Visual Inconsistencies

Many of the items in our data set had visual inconsistencies, such as the variation in color across different flavors of Powerade drinks, or shape distortion often observed in Miss Vickies and Stacy's chip bags. Looking at the average precision scores for items in these classes, Stacy's was evaluated to have a score of 99% compared to Miss Vickies, which scored a 57%. This disparity is most likely due to the consistent colors, logos, and shapes used on the Stacy's chip bags, while Miss Vickies underwent a recent logo redesign that was reflected in the inconsistency throughout its images in

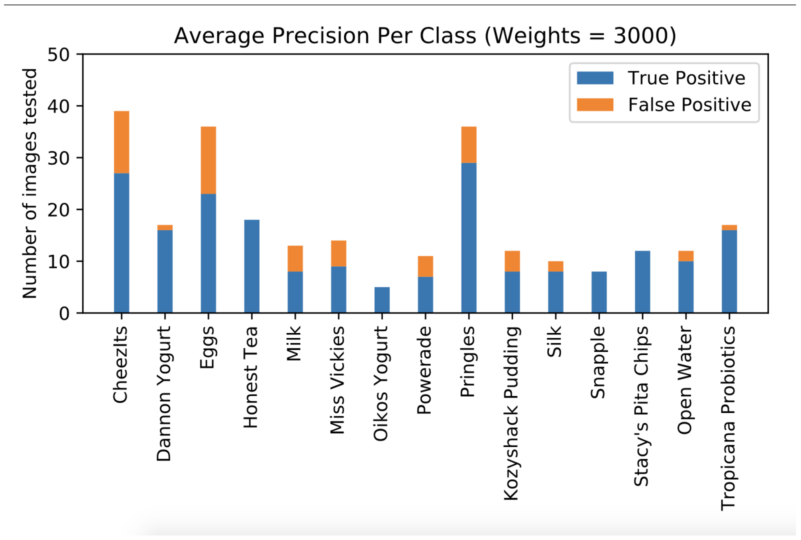


Figure 9: Average precision (AP) for a 15-class detector with weights calculated after 3000 epochs.

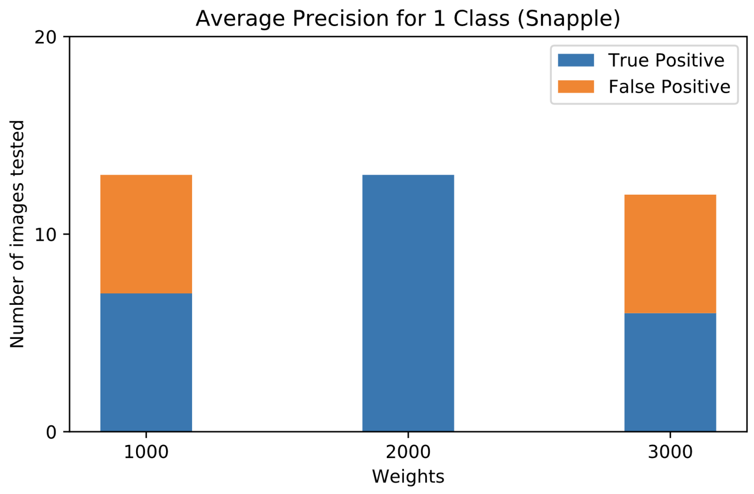
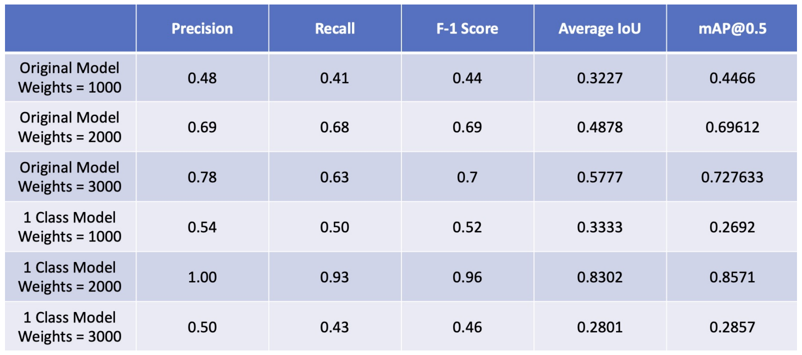


Figure 10: Average precision (AP) for a single class detector with weights calculated after 1000, 2000, and 3000 epochs respectively.

the data set. This analysis is generally consistent with other products that have consistent logo designs, such as Snapple bottles, Dannon yogurt, Tropicana probiotics, Open Water, and Oikos Greek Yogurt, which all have high precision values of over 80%-90%. Meanwhile, products other than Miss Vickies that also had inconsistent logo design (such as Silk) also had lower levels of precision.

There were also quite a few color variations across different flavors of products. This observation is also reflected in the Powerade drink, which had consistent logos but drastic color changes of red, blue and yellow throughout the data set, contributing to a lower average precision score of 68%.

Ultimately, we noticed that shape distortion made a negligible difference to the precision of the detection, with other factors, such as color, having a much larger influence during training. Interestingly, we noticed that eggs had a consistently low precision score. Though continued investigation



	Precision	Recall	F-1 Score	Average IoU	mAP@0.5
Original Model Weights = 1000	0.48	0.41	0.44	0.3227	0.4466
Original Model Weights = 2000	0.69	0.68	0.69	0.4878	0.69612
Original Model Weights = 3000	0.78	0.63	0.7	0.5777	0.727633
1 Class Model Weights = 1000	0.54	0.50	0.52	0.3333	0.2692
1 Class Model Weights = 2000	1.00	0.93	0.96	0.8302	0.8571
1 Class Model Weights = 3000	0.50	0.43	0.46	0.2801	0.2857

Figure 11: Evaluation metric values across the two different models, each at three different sets of weights.

into egg confusion matrices has yet to be done, we hypothesize that this phenomenon can be attributed to the lack of logo or distinct color which separates it from other objects. The eggs offered at Late Meal differ in color from white to light-brown colors, and though they are extremely consistent in shape, it is possible that its features are not distinctive enough for the model to confidently detect it.

5 Discussion and Conclusion

Ultimately, we believe we have developed a strong basis for a Late Meal detection application which can be expanded for full use with additional objects and training. We discovered that Tiny YOLO v3 is an effective algorithm for the purpose of our project due to its high efficiency, since it simply needs to run through a convolutional neural network of fewer than 20 layers once. A trade off of this efficiency, however, is that in comparison to some other object detection algorithms, Tiny YOLO v3 has slightly lower scores of precision. However, this loss is justifiable for the purpose of solving our problem, and our mean average precision value is still quite high.

For the purposes of our exploration into the different factors which influence the precision of a Tiny YOLO v3 model trained from scratch, our testing and analyses are still very preliminary. However, we make important steps into beginning to consider what affects accuracy, as well as the factors of training that lead to the best optimizations for this app.

5.1 Strengths and Weaknesses

A strength of our project lies in our curation of a customized data set of around 2,200 images that were manually annotated. The inclusion of manually taken photos in our training data set allowed for a very realistic data set that is representative of images that would actually be taken by students at Late Meal. Furthermore, due to the single-scan nature of the YOLO v3 algorithm, our detector performs extremely efficiently. This particular strength, however, is also accompanied by a downside because in the fact that a bit of precision is sacrificed in turn for speed. However, we conclude that the mean average precision values we obtained using YOLO v3 are high enough for the purpose of our study at 73%. A second weakness of our project lies in the small size of our data set. We only utilized about 100 to 200 images to train each class, and the size of the test set was even smaller. However, we view this as a trade off of accumulating a customized data set by manual annotating the images, which is a very time costly process.

5.2 Future Extensions

Given further time and resources, we would like to create a mobile application which utilizes this object detector so that students can simply hover their phone camera over Late Meal items to determine the applicable price on the spot. This will allow for a more practical and useful application of our research. We would also like to further develop the detector to include all of the items in the Frist Gallery beyond the prepackaged items in the Grab-n-Go section. In addition to training over more classes, we would like to obtain more images for each class of varying angles, lighting, and orientations. With this increased level of variation in training, we would hope to see greater accuracy.

Along similar lines of increasing detection precision, we hope to test for detecting logos rather than whole objects for the food products with various colors (i.e. Powerade bottles which come in blue, red, and yellow flavors at Late Meal), which would perhaps enhance accuracy for detection of such products.

References

[1] "Precision-recall¶." [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

[2] AlexeyAB, "Alexeyab/darknet," Jan 2020. [Online]. Available: https://github.com/AlexeyAB/darknet/.

[3] A. Kamal, "Yolo, yolov2 and yolov3: All you want to know," Oct 2019. [Online]. Available: https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899

[4] Rafi, "Train your own tiny yolo v3 on google colaboratory with the custom dataset," Jul 2019. [Online]. Available: https://medium.com/@today.rafi/train-your-own-tiny-yolo-v3-on-google-colaboratory-with-the-custom-dataset-2e35db02bf8f

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[7] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[8] Z. Yi, S. Yongliang, and Z. Jun, "An improved tiny-yolov3 pedestrian detection algorithm," Feb 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003040261930155X