Identifying Cereal Boxes with Computer Vision

Mark Bosmediano

December 19, 2025

1 Introduction

Identifying objects is one of the key problems in Computer Vision. Training a model that can process pixels and identify their traits is the problem we have been studying to solve all semester, with each homework and lecture expanding our knowledge of computer vision techniques and strategies. As my final project, I decided to apply what I learned for myself and train a machine learning model to identify cereal boxes in many contexts.

2 The Model

The model I used is the Ultralytics YOLOv8. This is a one-stage, anchor-free object detector. As a one-stage detector, it predicts object locations and class labels in a single forward pass. Being anchor-free, it does not rely on predefined bounding box shapes, instead directly figuring out and establishing object boundaries from image features. This makes the model more flexible when detecting objects of varying sizes shapes.

This was useful in my project as it enabled my model to adapt its bounding boxes to cereal boxes of differing sizes, making it so I did not have to carefully define the bounding boxes ahead of time to account for differing box sizes.

To get more into the specifics, I used YOLOv8-nano, a lighter variation of YOLOv8. I decided to use this model due to its size to accuracy tradeoff being the most favorable among the wider options. When compared to other models like Faster R-CNN or the other YOLO versions, YOLOv8 gave the best performance for the limited compute on my laptop. Since this was also a relatively simple assignment with a limited dataset compared to more expansive projects like identifying faces, the gains from using larger models was not worth the extra time and risk of overfitting it would have brought.

3 The Dataset

Most of my dataset was sourced from a dataset found on Roboflow, an online platform for computer vision projects. I was able to find a dataset (link: https://universe.roboflow.com/johndow/cereal-0d7rp/dataset/1) that provided a set of over 500 labeled images pre-distributed into training,validation,and testing sets. I saved these into the /test, /train, and /valid folders in the Cereal_YOLO/dataset folder of my project. This dataset also included a number of randomized transforms applied to the images, such as random rotation and blur in order to make training more robust. This gave me a strong starting point to build my model from, with many brands and sizes already aquired and labeled for me. One issue I ran into with this dataset was that it was pre-labeled for polygon annotations, meaning its labels had extra data that recognized background elements such as the shelf the box was on among other things. This presented issues I will go into more detail with in a later section.

With this dataset acquired and the model trained and tested on the dataset's training data, I attempted to test my dataset on a few images I took of cereal boxes in my kitchen. However, the model performed very poorly on these. These images can be found in Cereal_YOLO/dataset/testingImages. This was due to the vastly different lighting and orientation compared to the initial dataset I trained on. In order to remedy this and make my model more robust to novel situations, I did two things. First, I took a series of photos from around the kitchen that did not include cereal boxes to provide the model with hard negative examples. With these negative examples, the model could learn more about what the expected background was like sans cereal boxes, which would help improve its accuracy when a cereal box did appear. These were labeled Hard_Neg1-20 and placed into the training images under Cereal_YOLO/dataset/train/images. I then took a series of examples with multiple cereal boxes under different lighting to help train the model for specifically identifying multiple cereal boxes in one image, providing separate bounding boxes for each. These are labeled Tig_Pack1-14 in the same training folder. These images are paired with label text documents in the Cereal_YOLO/dataset/train/labels.

4 The Process

As mentioned before, I began by collecting my dataset. Once these were established, I installed all the required packages into a python virtual environment and began to explore models. As mentioned before, I found YOLOv8-nano to be the best model for me. When I went to train the model, YOLO returned an error message about my dataset. This is when I realized that many of the entries in my dataset had extra information in its labels relating to polygonal annotations and had many other classes beyond the binary identification I was aiming for. In order to resolve this, I modified the given labels to be within the scope of my project by removing the excess information and setting the labels properly. Once I had the model trained, I began testing on the downloaded dataset's test set. The results can be found in Cereal_YOLO/runs/detect/predict. These results were strong, so I began testing on the new pohotos as stated. This is when I realized that my model's training was too narrow and did not properly generalize to newer situations.

The results of the initial tests can be found in Cereal_YOLO/runs/detect/predict2 and .../predict3. As you can see, the model struggled to find the boxes, often only boxing one of them or boxing empty space thinking it was a cereal box. In order to remedy this, I did research into fine-tuning my model, and learned about the technique of creating hard negatives and providing more complete examples of multiple cereal boxes for the model to train on. The results of this can be found in Cereal_YOLO/runs/detect/predict4. As you can see, the model was so much better at identifying the boxes, but it still seemed to be having trouble hallucinating extra boxes. In order to address this, I raised the confidence threshold of the model to exclude anything under .5 confidence, which as seen in Cereal_YOLO/runs/detect/predict5_FINAL. With this, I considered my work finished for the purposes of this project.

5 The Results

After all the training, the final model achieved a precision of 0.969 and recall of 1.000, indicating that the detector successfully identified all cereal boxes while maintaining a low false-positive rate. The model reached an mAP@0.5 of 0.995, demonstrating reliable object detection, and an mAP@0.5:0.95 of 0.900, reflecting strong but not perfect bounding box localization. As mentioned before, a qualitative analysis showed the bounding boxes to be accurate to the cereal boxes. This indicates strong performance on any cereal in multiple settings and under multiple light situations.

Despite the optimistic data above, the model is likely limited in its ability to identify cereal boxes in a wider array of contexts. Due to the fact many of the hard negatives were taken of the kitchen where the test images were taken, as were many of the fine tuning images involving multiple boxes, it is possible that the model might be limited to identifying boxes within that one room. However, the tests do indicate a strong resistance against changes in lighting, as the difference in lighting did not cause a reduction in accuracy.

6 Conclusion

This project was a very interesting exploration of binary object identification. It allowed me to do a lot of research into new technologies, and I hope to continue to learn more in the future as I continue to experiment with computer vision, both in my personal life and in my career as a software developer.