Final Project Report

Lex Kim, Hylene Wu, Neel Jay, Jordan Maggin, Zhuo Cheng Xie

CMSC421

LITERATURE REVIEW

YOLO (You Only Look Once) object detection is an end-to-end method for real-time object detection that differs from previous methods that rely on using classifiers to identify and localize objects. YOLO was first introduced at the 2016 CVPR conference by Redmon et al [1] with the key innovation of simultaneously predicting bounding boxes and class probabilities. This differs from previous methods that performed these tasks in series, first identifying bounding boxes and then classifying them into objects. The parallel approach of YOLO makes it ideal for our real-world application which requires speed and accuracy to detect aircraft.

YOLO has been iterated on and improved several times since it was first introduced, with YOLO9000 (Redmon et al [2]) coming out the following year, and followed again by YOLOv3 (Redmon et al [3]).

A notable issue with YOLO is difficulty detecting overlapping objects as well as small objects. Additionally, the early versions of YOLO relied on a deep-learning framework called Darknet which was written by Redmon and is far more complicated to use than frameworks like Pytorch and Tensorflow.

However, YOLOv8, the model we used for our project, is built on top of Pytorch and is far simpler to use than the previous versions, while also being more accurate and faster when compared to other object detection models [4].

Another prominent model for object detection is Faster R-CNN by Ren et al [5], which uses a region proposal network to identify potential bounding boxes, and then classifies those bounding boxes. However, Faster R-CNN is slow as it takes time to generate potential bounding boxes, and then classify them. SSD (Single Shot Multibox Detector) by Liu et al [6] addresses this problem by eliminating the region proposal network and instead using localization and classification in one shot like YOLO. It does this by generating feature maps, applying default bounding boxes, and then using a classifier to determine the object all in one go.

INTRODUCTION

The Federal Aviation Administration (FAA) oversees tens of millions of flights in the United States, spanning nearly 20,000 airports, with over 500 air traffic control towers, to orchestrate the safe travel of passengers. To keep track of the large number of aircraft passing through an airport every single day, the FAA relies on cutting-edge technology to keep track of the planes on the ground. The goal of this project is to produce an end-to-end solution for use in airport control towers to track aircraft via a live camera feed. This problem is known as object detection and consists of putting a bounding box around an object in an image or video and classifying that object. For the scope of this project, this meant fine-tuning a preexisting object detection model to locate and identify aircraft in an image and video.

To produce a valuable product, the object detection model needs to be accurate, able to pinpoint an aircraft in an image, and do it fast, making real-time use feasible. This is a difficult task as there is generally a tradeoff between accuracy and speed for computer vision models. However, a model that is performant and accurate has implications beyond our use case in airports, for example, autonomous driving, manufacturing quality insurance, robotics, etc.

INITIAL GOALS

The challenging nature of our project led us to incrementally adapt our model to achieve our final goal of detecting aircraft types in images and videos. We first began by training an object detection model using the YOLOv8 architecture with one class (airplane). This model performed very well, with 93% accuracy and real-time inference. While the model performed exceptionally well, it did not cover the scope of our problem. Our goal was to classify different vehicle types commonly found in airports, so we trained an additional model with more classes (various vehicle types) in order to achieve our goal. Unlike the previous model, this model performed far worse. This is likely due to the training dataset having too little data as well as having multiple under-represented classes (truck, aerostat, firetruck…) which would have skewed the model’s accuracy results simply because the model wasn’t fed enough data to predict those classes accurately.

SOLUTION

We selected the YOLOv8 architecture for this project because of its simplicity (easy to train via the Ultraltyics notebooks), its accuracy compared to other models, and its speed. For previous YOLO models, one had to manually freeze layers in the model to retrain the model using transfer learning. However, a benefit of the YOLOV8 architecture is that this process is abstracted from the user, making training simpler. We trained our model on the Roboflow platform, which provides industry-grade pipelines for data creation, model finetuning, and deployment.

To train the model, we simply had to import the dataset from our Roboflow workspace, which already had the dataset split into training, validation, and testing, and then run a single YOLO command. For our models, we trained with an epoch size of 25. This training session took about 35 minutes on the airplane set (~4800 images) and almost 1 hour on the multi-vehicle set (~6700 images).

Visualization of model results was a key part of serving our initial use case. Our UI allows bounding box annotation of both images and videos. In addition, the BYTETrack algorithm is applied to video inputs, which allows for multi-object tracking with good results for high and low-confidence detections [7]. This allows for accurate tracing and heat-mapping annotations for various airplanes on video. In an air traffic control setting, these annotations can be used for a variety of functions, including evaluating aircraft landings/takeoffs or identifying danger areas in airport airspace.

RESULTS / EVALUATION

Stats:

Precision: 93.6%

mAP: 92.7%

Recall: 85.5%

Qualitative Analysis:

The first model (aircraft only) performs exceptionally well on images, youtube videos, and live cameras. However, as is common with the YOLO models, it struggles with object occlusion, low image quality, and small objects. For example, it works poorly on certain footage of aircraft in the air, taken from the ground. This is likely due to the lack of representation of this type of data in the dataset that was used to train the model.

The second model (vehicle and aircraft) performed far worse than model one. This is likely due to the smaller data set size, making it difficult for the model to learn the different features of aircraft and other vehicles. However, considering the small dataset size, and relatively lower training time, the model performs better than one would expect. Additionally, Figure 2 illustrates that the model's precision did not improve after training, furthermore indicating that a larger dataset was needed, as well as more training.

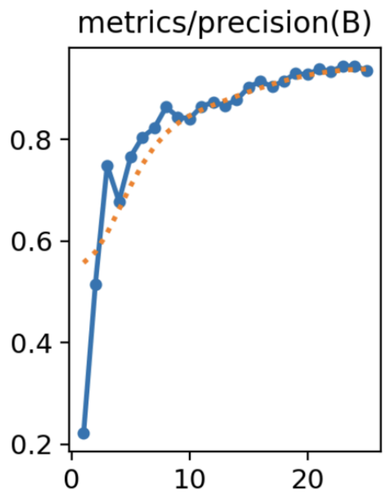

Figure 1
(Model 1)

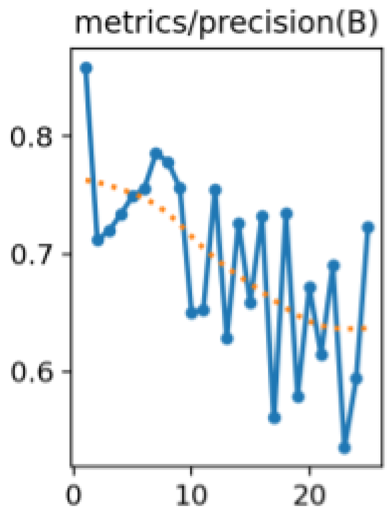

Figure 2
(Model 2)

INDIVIDUAL CONTRIBUTIONS

Hylene: Trained models

Lex: Built GUI

Neel: Built Backend

Zhuo: Worked on slides and report

Jordan: Wrote Report

REFERENCES

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv preprint arXiv:1506.02640.
2. Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv:1612.08242.
3. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.
4. Glue, Rustem. “Yolov8, EfficientDet, Faster R-CNN or Yolov5 for Remote Sensing.” *Medium*, Medium, 2 May 2023, medium.com/@rustemgal/yolov8-efficientdet-faster-r-cnn-or-yolov5-for-remote-sensing-12487c40ef68.
5. Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv preprint arXiv:1506.01497.
6. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Lecture Notes in Computer Science (pp. 21–37). Springer International Publishing. doi:10.1007/978-3-319-46448-0_2
7. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box. ArXiv:2110.06864 [Cs]. https://arxiv.org/abs/2110.06864