You Only Look Once (YOLO)

- Faster but less accurate compared to a Convolutional Neural Network (CNN)
- Struggles with smaller objects
    - Might have problems detecting the robots depending on the camera view
- 45 frames per second
    - process streaming video in real time with less than 25 milliseconds of latency.
- Regression instead of classification
- How it works
    - Splits image into NxN grid
        - Each cell predicts the class of object it contains
        - Along with confidence value
    - Cuts out cells that do not contain objects
    - Disregard empty sections of each cell (Intersection Over Union)
    - Non-Max Suppression
        - Keep only the boxes with high-confidence values
- Model trained with CVAT.ai
    - Easy to use
    - Put a bounding box around each object in an image manually
    - Took around 12 hours for 300 photos

YOLO will be used for this project. It is simple to use and faster than most other CNN options. OpenCV will be used along with YOLO to modify camera footage. CVAT.ai also makes the model fairly simple to train.

Our Model uses around 500 photos and was trained over 300 epochs
The epoch numbers can be adjusted in the trainModel.py

Original Model Training took 30 hours to complete
Secondary Model Training took 24 hours to complete

Train/Val/Box Loss
- Difference between human set bounding boxes and the predicted bounding boxes from the model
- Lower value = more accurate model

Train/Val/CLS Loss
- Measures the correctness of classification (AMR robot is detected as a robot)
- Lower value = more accurate model

Train/Val/ DFL Loss
- Problem with class imbalances
- Adjust the weights of each object
    - Harder to classify objects=higher weights

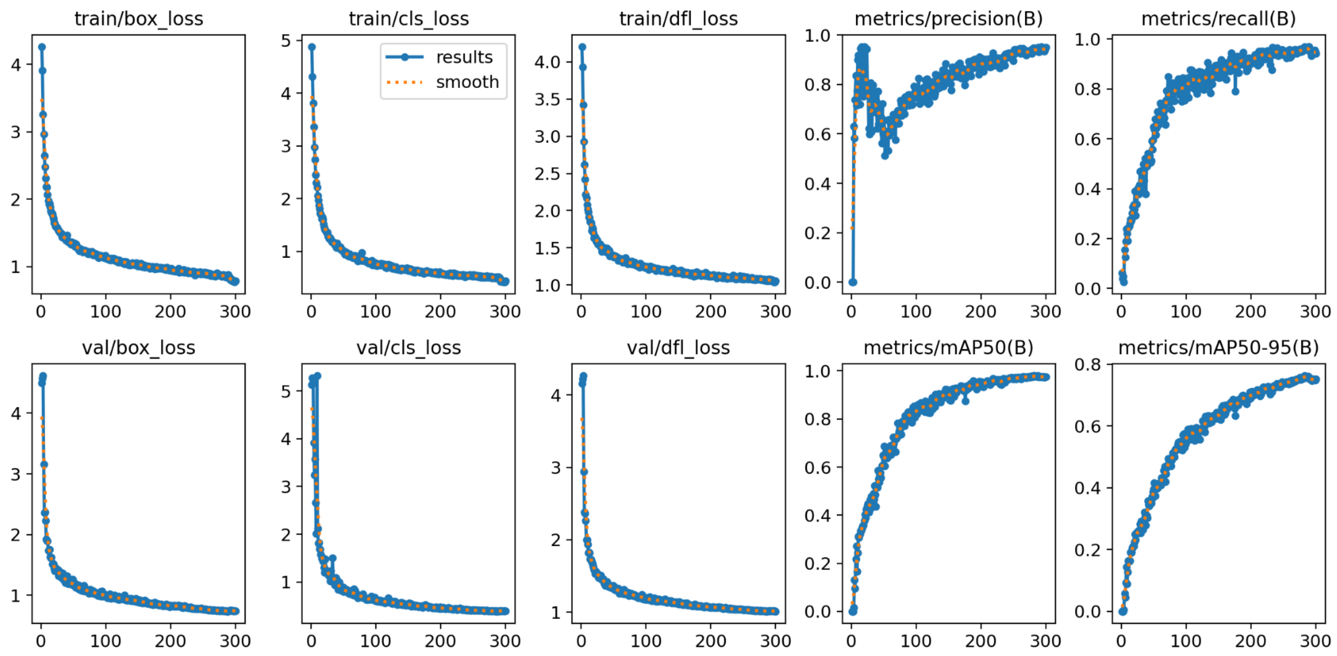
Metrics/Precision B
- How confident the model is across training cycles

Metrics/Recall

- Objects that were detected correctly

MAP

- Mean Average Precision
- 50 = detection must be above 50% confident
- 50-95=Detection along a range of 50-95



YOLO Latency

The latency of YOLO object detection drastically differs based on the hardware running the Python script. While running on my laptop, I was averaging a 300ms delay. However, if I run the script on my desktop computer, the delay drops to 52ms.

We ran object detection on 30 fps video at a resolution of 480p.

OpenCV (Object Detection and Tracking)

- Free
- Open Source
- Support for C++, Python, and Java
- 1.5 times faster than Keras
- Supports Linux, Mac, Windows, iOS, and Android
- Extremely Popular = vast amount of tutorials online for object detection
    - However, official documentation is messy
- Focused on computer vision
- licensed under the Apache 2 License
    - Free to use for anyone/anything
    - No Liability
    - Can not use company brand name or logo in trademarks
    - “Use at your own risk”

OpenCV is a good library. However, when paired with another library like YOLO is where it shines. Its immense popularity leads to a vast array of documentation and tutorials on every corner of the library. OpenCV was originally coded in C++. However, the open-source software has added interfaces to support other big programming languages, including Java, and Python, giving it versatility in how to write code using the library. Not only does it support some of the most popular programming languages, but it also supports common operating systems as well. This includes Linux, Mac, Windows, and even phone operating systems such as iOS and Android. OpenCV possesses the ability to do both object detection and object recognition within one library. However, for our use case, the pre-trained models will not be suitable. We will need to use a separate model to recognize the AMR robots as well as the shelves and any other objects we may need to track.

KERAS and TensorFlow

Originally, I attempted to use the Python libraries Keras and TensorFlow. However, I found that the training for these libraries was more difficult compared to YOLO. Also, YOLO training provides a large amount of data and graphs with the model which makes it extremely easy to analyze the accuracy and diagnose any problems with the model. I did find a site to train a Keras and Tensorflow model made by Google (Teachable Machine). However, this training did not allow the user to put bounding boxes around each object, which led to the YOLO model being more accurate.