CSCI-B657: Final Project Interim Report

Ethan Eldridge – etmeldr
Gavin Hemmerlein – ghemmer

May 4, 2021

1 Introduction

With the advent of rapid developments in the field of computer vision, deep neural networks have become useful tools to vastly improve the response time of an object identification model. The object detection task is the method of defining the object (or objects) to be detected, drawing a bounding box around the object in an image, and using accuracy calculations to identify the recognized object according to its predicted labels. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them. An example of this can be either following a particular vehicle on a road path or tracking a ball in any sports game like golf, cricket, baseball, etc. Object tracking is slightly different in comparison to object detection, as the algorithm not only needs to detect the particular object but also follow the object across time and space with the bounding box around it. Due to the myriad of challenges that applying this technique across time and space proposes, this project will be focusing primarily on object detection in images. The various algorithms to perform these tasks are Region-based Convolutional Neural Networks (R-CNN), Single Shot Detector (SSD) algorithms, and You Only Look Once (YOLO) algorithms, among many others.

2 Project Background

2.1 Detection, Recognition, and Identification

Detection, Recognition, and Identification are all important aspects of an imaging task. These components are used to describe the fidelity of the image within a scene. With the completion of each component, a certain level of detail is revealed to the viewer.

Detection is relatively straightforward in images. Detection is the understanding that something of desire is in the field of view. Once a desired object has

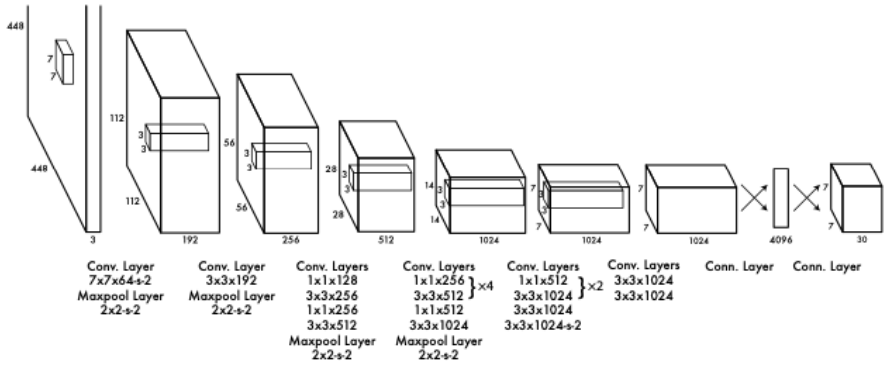
been defined, there are a wide variety of tools that can be implemented to detect similar-looking objects in the image. Detection is the area with the least amount of fidelity.

Recognition is the next step in narrowing down a model's prediction. Though the Detection step is able to identify objects that are similar to the desired object, there are often confounding variables that require some additional detection. The Recognition component of the object detection task takes the objects from the Detection component and attempts to filter out categorical outliers that may have made it past initial detection. An example of this is being able to tell the difference between a human and a human-shaped object, such as a shadow or amorphous figure. This is also used frequently within computer vision fields.

The final and most helpful component of object detection is Identification. This step teaches the model to distinguish between class-specific instances. For example, an object identified as a human can be further identified as a postal worker or a police officer. Identification is where the model begins to pull out semantic information from the Detected and Recognized object.

2.2 The YOLO Algorithm

YOLO is a relatively recent algorithm that is intended to quickly analyze pictures and find a categorization at the loss of some accuracy. This approach allows for near real-time analysis of a scene. This is achieved by a series of gridded segmentation, detection, and non-maximal suppression stages that allow the algorithm to detect objects at a much faster rate than other convolutional neural networks, which may require multiple passes of a single image to detect several classes in that image. Below is an image of the typical network architecture for a YOLO model:



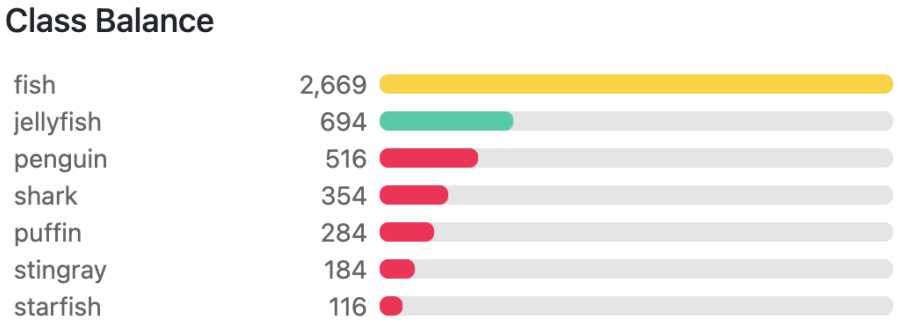
YOLO Algorithm [J. Redmon, et al.]

The goal of this project is to take a pre-trained YOLO model and apply it to a new dataset. This process will include training the model, testing the model,

and tuning its hyperparameters to acheive optimal performance. The YOLO model implemented in this project is the YOLOv5 model from Ultralytics. This iteration of the YOLO algorithm was chosen because it is written in the PyTorch deep learning framework, allowing for ease of use and implementation on custom datasets. We hypothesize that, given its extensive training on other images, the model will be able to generalize to the new dataset and improve its detection accuracy over several training iterations.

2.3 Dataset

Though the YOLOv5 model had been initially trained on the COCO image detection dataset, this project applied the model's existing weights to training and testing on the Roboflow Aquarium dataset. This dataset was chosen as it was easily accessible and had already been formatted for use by YOLO algorithms. This means that in addition to images and class labels, the dataset also included approrpriate bounding boxes for the model to train on. Additionally, given that the classes featured in this dataset (fish and other marine animals) are not featured in the COCO dataset that the YOLOv5 model was trained on, we thought that this dataset would offer a reasonable challenge to the pre-trained model. The dataset itself consisted of 638 images collected by Roboflow from two aquariums in the United States: The Henry Doorly Zoo in Omaha (October 16, 2020) and the National Aquarium in Baltimore (November 14, 2020). The number of instances of each class across all images of the dataset can be found below, taken from the Robowflow website:



Class Balance of Aquarium Dataset

3 Methods

To start the construction of the data pipeline, the dependencies and other required utilities for the model needed to be set up. This process involved locating the required files in the Ultralytics repository for YOLOv5 and downloading them to the disk. Once the necessary data and utilities had been set up, the next step was to download the model itself. There are several different iterations

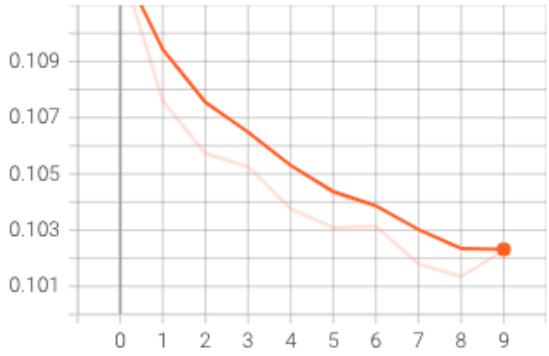
of the YOLOv5 algorithm avaliable from the Ultralytics repository, all of which feature a different number of parameters, different accuracy rates, and different runtimes. Despite its relatively average accuracy rate among the other versions of the algorithm, "YOLOv5s" iteration of the algorithm was implemented for this project as it featured the fastest runtime of all others. This was particularly crucial as the group had limited resources for hardware acceleration.

Once the data had been pre-processed and the model had been initialized, the next step was to train the model and examine its inferences. A series of .yaml files and training scripts provided by Ultralytics were used to train the model. Changes were made to these files in order to accommodate the number of classes in the dataset (7) and the number of epochs for each training session. After each training session was completed, the results were plotted and several inference images were given to the model for examination.

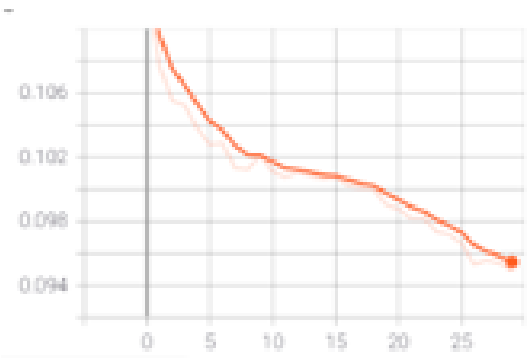
4 Results

4.1 Quantitative

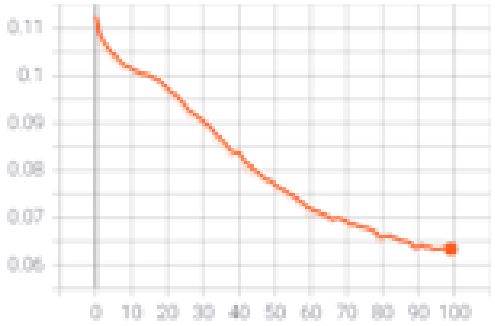
The following are the results of training the YOLOv5s model on the Roboflow Aquarium dataset. First, we have the box loss curves following training sessions of 10, 30, 100, and 500 epochs:



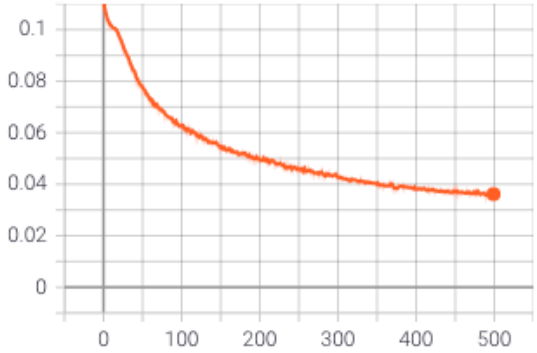
Training Epochs (x-axis) vs Training Loss (y-axis) for 10 Epochs



Training Epochs (x-axis) vs Training Loss (y-axis) for 30 Epochs



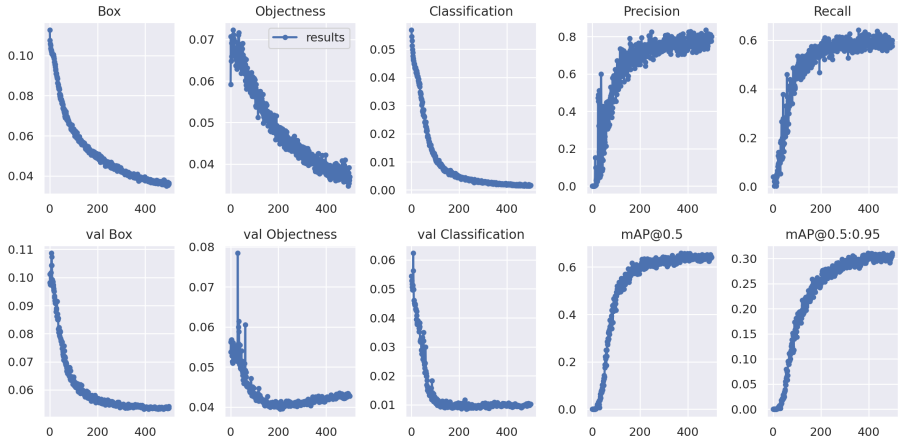
Training Epochs (x-axis) vs Training Loss (y-axis) for 100 Epochs



Training Epochs (x-axis) vs Training Loss (y-axis) for 500 Epochs

From these figures, one can observe notable differences between the minimum training losses achieved during each training session. Though all training sessions showed a decrease in loss, the 10-epoch session achieved a minimum of just under 0.103, while the 500-epoch session was able to drive the loss below 0.04.

A variety of other metrics were used to evaluate the model. These graphics were created from the Ultralytics training script and feature various types of loss calculations, precision, recall, and mAP scores:



Evaluation Metrics for YOLOv5s at 500 Epochs on Aquarium Dataset

4.2 Qualitative

After training had been completed, the model was given inference images to produce qualitative results from the experiment. The following two images are particularly interesting examples from this section, as they are concrete examples of the improvements the model was able to make given more training epochs:

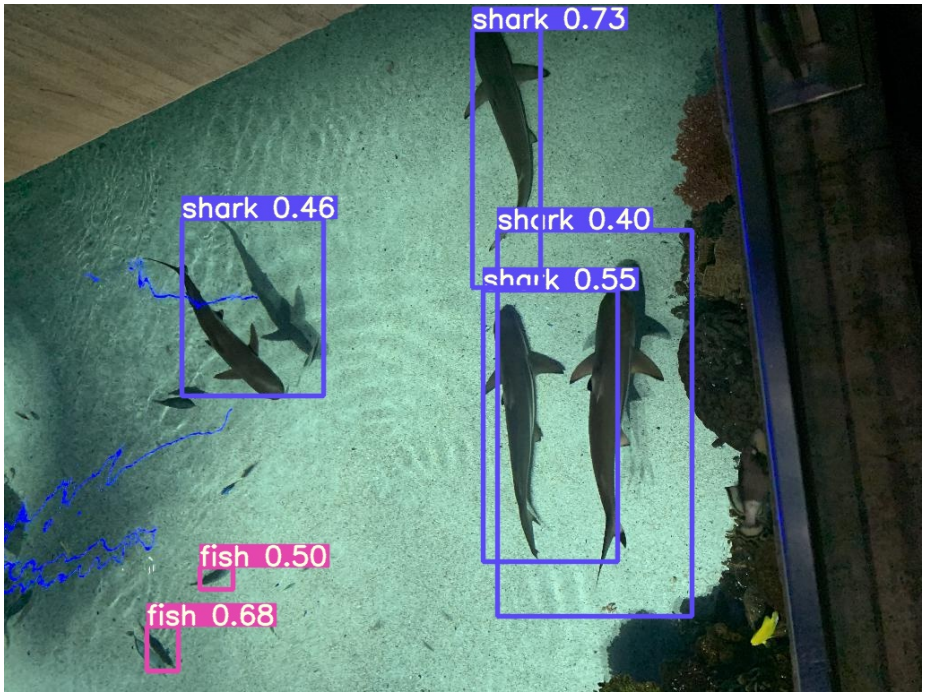


Image of Shark and Fish at 100 Epochs (some fish remain unclassified).

Image of Shark and Fish at 500 Epochs (higher confidence, more fish identified).

In comparing these two identical images, we can see qualitative evidence that the model was able to learn more about the dataset given more training time. For example, virtually all of the objects that were identified in the 100-epoch image have higher confidence values in the 500-epoch image. Additionally, there are two identified fish in the left and bottom-right sides of the 500-epoch image that were previously unidentified in the 100-epoch image. Finally, one can observe improvements in the size and shape of each bounding box in the 500-epoch image. The bounding box for the shark at the top of the image fits the shape of the shark more completely in the 500-epoch inference. The far-right shark is also identified more cleanly in the 500-epoch inference.

5 Discussion

Ultimately the model performed as expected on the new dataset. Despite having been trained on a different dataset (COCO) that shared none of the same object classes as the aquarium dataset, the model was nevertheless able to generalize to the aquarium dataset and identify its objects with reasonable accuracy. This result could still be improved by using a much larger aquarium dataset. According to the practices recommended by Ultralytics, the developers of YOLOv5s,

this model begins to achieve ideal performance rates when its dataset consists of more than 1,500 images per class. Within those images, more than 10,000 total instances per class must also be detectable. Given that the aquarium dataset featured only 638 images, and given that the most frequently occurring class (fish) had less than 3,000 instances across the entire dataset, it is clear that the model could have improved its results given more examples to train on.

6 Conclusion

The YOLO object detection algorithm is well-known for its ability to quickly and accurately detect images in a variety of different contexts. This experiment aimed to put the algorithm to the test by applying a pre-trained version of YOLOv5 to objects that it had never seen before. As hypothesized at the beginning of the experiement, the model was indeed able to generalize to the new dataset, making both qualitative and quantitative improvements in detection with additional training.