Spencer Stafford
04/21/2025
CSE185 Spring 2025

Lab 4 Report

Data Collection and Labeling:

I started the lab assignment by taking around 62 photos of the Spot-It! Cards during lab session. I tried to make sure that each photo had a different match and a slightly different rotation to ensure that the model had a suitable library of “scenarios” to learn from during training. Although 62 seems to be a fairly low number for a dataset, I moved on to the next part of the process: labeling the data.

In order for the model to learn, it needs to be told what an object is and where it appears in an image. This entails labeling all of the images in your dataset and then separating them into training and validation sets. To label my data, I used an open-source tool called Label Studio. I inputted the class list given to us by Ross, and drew bounding boxes (with appropriate class names) around every object in every image in my dataset.

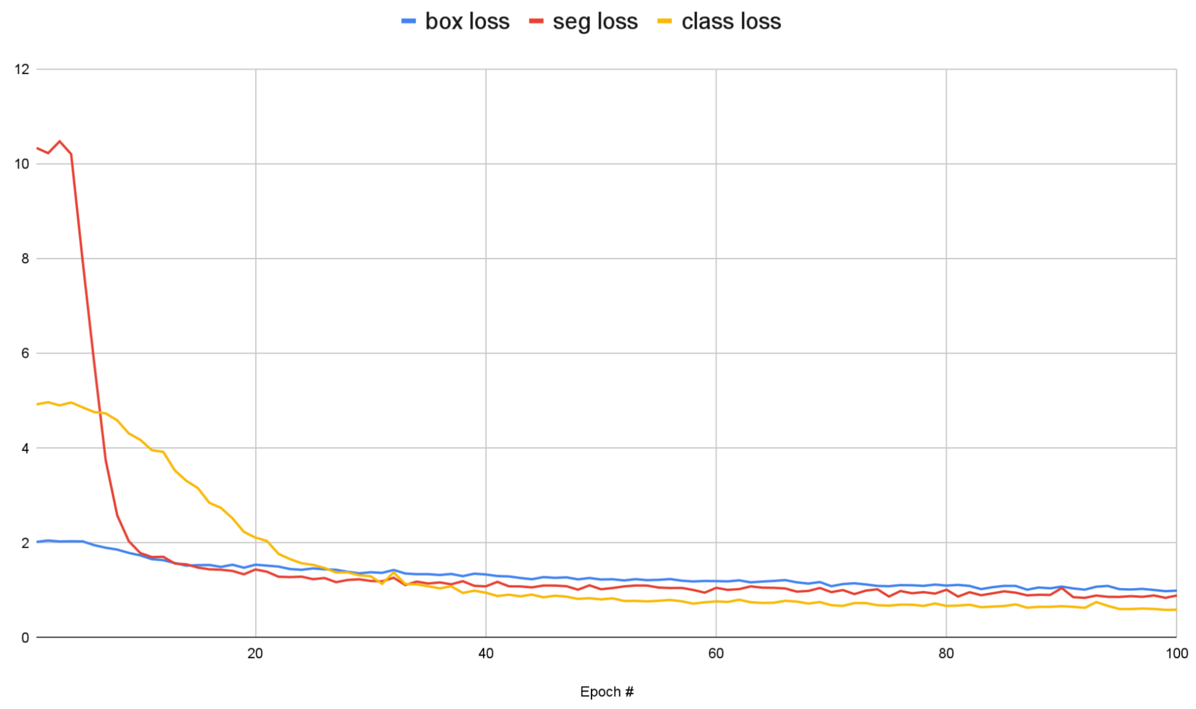
When I finished labelling each image, I exported the now-labelled dataset. To train a model, you need both a training set and a validation set. To make these sets, I split my annotated images in half: 31 images and their associated labels would go to the training set, and 31 would go to the validation set. I then made a .YAML file, which included the paths to the training and validation sets, as well as the amount of classes and all of the class names. The YAML file essentially tells the model where the data is, how much there is to classify, etc (at least for a YOLO model).

Training the YOLOv8 Model:

Now that I had the annotated, formatted data, it was time to start training the model. I chose to use YOLOv8 as my model because everything I read about it told me that it has good training performance and speed, which is something that would be valuable for this assignment.

After loading in the model, I wrote some code to train it on my dataset. When training a YOLO model, you must include the dataset, epoch amount, batch size, and image size as parameters. The dataset is, of course, the dataset that I annotated. The epoch amount tells the model how many cycles it will be training for. I started with 20 epochs, but after getting subpar performance from that, I increased it to 100. The batch size is the amount of images that the model will be able to learn from at once. The batch size represents the amount of images the model is allowed to look at before adjusting its weights. I chose a standard 16 for that. Image size is the size that the image will be resized to before entering the model. I also chose a standard 640 for that.

After configuring all of the training settings, I ran the model. The results of the training on box loss (associated with bounding box accuracy / accuracy in knowing where an object is), seg loss (measuring how well the model predicts segmentation masks), and class loss (evaluates how well the model classifies objects in a bounding box or seg mask) are below.



As seen in the graph (which I made by taking all of the epoch-to-epoch info and putting it on an excel sheet), the model sees significant improvement to the general loss in the first 20 epochs. From there on, the improvement is a lot less drastic, but still gradual, and that was reflected in the resulting test set performance. After initially training my program on only 20 epochs and seeing how badly it performed on the evaluation set (only getting about 18% detection rate), I was surprised at how much could be learned by doubling or tripling that number. I tried training it on 40, 60, 80 epochs, even 120 epochs, but I found that the best performance came from using 100.

Testing with the Evaluation Set:

After successfully training my model to detect the Spot-It! Objects, it was time to run it against the evaluation set given to us by Ross. As each image is passed through, the model tries its best to detect and classify all of the objects appearing. In order to make a prediction about which object is the match on the Spot-It! Card, I made a counter that counted the frequencies of each class per image. If an image saw 1 pair of matching detections, then that would be made the prediction. If there were no duplicate classes detected, then no prediction is made. If there are multiple pairs of duplicates, then the first detected one would be made the prediction. After receiving results from the evaluation set, I decided to change the multiple-pairs-found case so that, instead of making the first detected pair the prediction, it wrote “Too many choices”. This was so that I could more accurately gauge my models accuracy.

Successes and Shortcomings:

My model was successful in correctly detecting about 58% of the matches in the evaluation set. The training process saw significant learning and loss reduction up to a certain amount of epochs. Although the model was successful in its learning, I think that the small size of the dataset really set back the performance. A smaller dataset means less images for the model to learn from, which means that it will never perform as well as a model trained on a larger one for the same (or even less) epochs. Furthermore, the smaller set stops working at a certain point. For me, that point was at anything after 100 epochs. The model's performance would degrade from that point onward, and I think that is due to the low number of images in the set, perhaps even a sign of overfitting.

Improvements Made (as of 04/24/2025):

In order to address the small-dataset issue that I outlined above, I took around 52 more photos and annotated them accordingly. I evenly split and added them to my training and validation sets, making those sets have about 57 images each. Using this new dataset, I followed the exact method outlined above, this time running for 135 epochs. The output of the training process saw an accuracy increase of ~13%, jumping from the 58% I got with the smaller dataset to 71% accuracy with the expanded dataset. Anything above 135 epochs was prematurely stopped by YOLOv8's "patience" feature, meaning that the model stopped learning meaningful information after that point.

Box Link to my Dataset, Saved Weights, and results CSV file:

https://ucmerced.box.com/s/nimafarn8v5m1df5jsyp38twyjf6dkqs

Citations:

Training YOLOv5: https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/
YOLOv8 training fundamentals: https://docs.ultralytics.com/modes/train/#multi-gpu-training
YOLOv8 roboflow overview: https://roboflow.com/model/yolov8
Label Studio Documentation: https://labelstud.io/