Matthew Backlund

CS 5600 Project 1

I started this project by getting an environment set up for the project to run in. I set up a python venv called *beeenv* and imported the necessary packages. However, I quickly ran into issues with the dependencies being incompatible with each other. After about 2 hours looking at forum posts and asking ChatGPT (which was absolutely no help) I realized that the version of CUDA I had installed on my computer was incompatible with pytorch. After this realization I uninstalled CUDA and removed all packages from *beeenv*, then redownloaded CUDA 12.4 and installed versions of pytorch, torchvision, and torchaudio that were compatible with this CUDA version.

After getting CUDA set up correctly I tried to install the remaining packages my project needed, specifically onnx, but couldn’t download anything. Somehow my venv had become corrupted and had lost the ability to download pip packages. After retracing my steps I realized that I had changed the name of one of the parent folders which caused *beeenv* to try to download to a path that didn’t exist anymore. After deleting and recreating *beeenv* I was finally able to get all the python packages I needed installed.

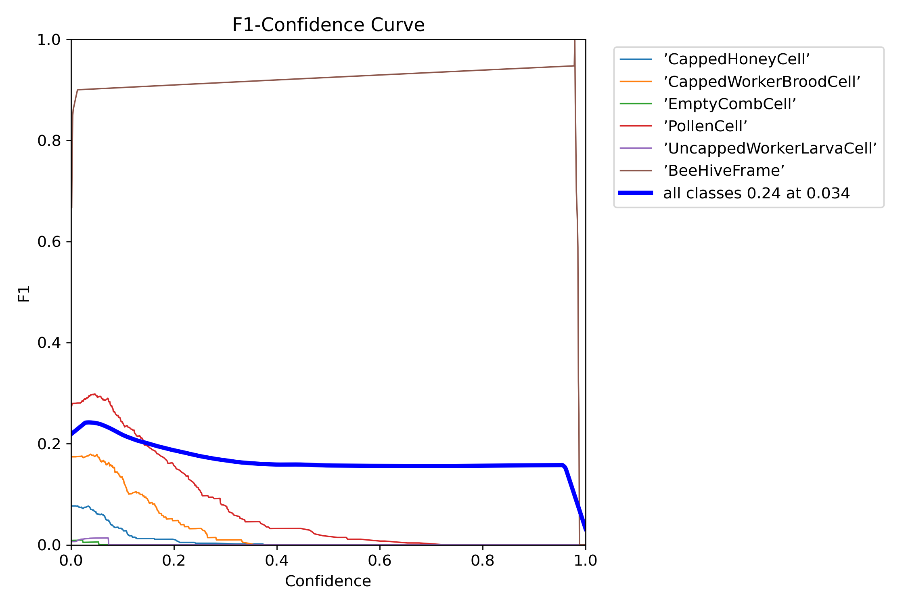
My next challenge arose when the YOLO would still crash almost immediately when it started running with the following error:

RuntimeError:

An attempt has been made to start a new process before the current process has finished its bootstrapping phase.

After a bit of research I realized that the starter code we’d been given needed to be inside of a function and that the program should have an `if \_\_name\_\_ == “\_\_main\_\_”’ block. My understanding of this is that multiprocessing is implemented differently on windows and uses ‘spawn’ instead of ‘fork’ (The Linux Foundation, n.d.). Because of this our code needs to be wrapped in an if-clause to protect it from executing multiple times.

Once the code was wrapped in an if-clause I was finally able to get the code to run, only to be sorely disappointed. The model was able identify Beehive frames with about 95% accuracy, but that was it. All the cells were labeled as background, and all the background was labeled as cells. I bumped the number of epochs up to 300 and got nearly identical results, with only the average certainty for beehive frames increasing slightly.

After running a few more test runs my model wasn’t getting any better. I was still using the default YOLO model (yolov8n.pt), so I decided to use the model I did 300 epochs on as the new baseline model. I trained about 300 more epochs on this model. This gave me immediate positive results. Now the model was able to correctly identify about 6% of pollen cells and about 2% of capped worker brood cells! This does not sound like much, but it was a huge improvement from where I had been previously. Results from this model can be found in the following figures:A group of graphs showing different sizes of data

Description automatically generated with medium confidence

I moved this new model to be the new starting model and did the same thing again. However, this time the model stopped training after 101 runs with the following output:

EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 1, best model saved as best.pt.

The model had stopped training because it was not getting any better results after 100 epochs. My model had gotten as good as it could get with only increasing the number of epochs it was running.

I did some research on the earlier method I had been using, where I used a pre-trained DNN as a starting point for another. From my reading, the strategy is called transfer learning. It is often used to take knowledge a model has learned from one dataset and leverage that knowledge on another dataset. Since we are using the same dataset each time we train, there is not a huge advantage to using this strategy unless I begin doing hyperparameter tuning to change the way the model trains each run.

At this point my only thought was to start hyperparameter tuning. I did some research on hyperparameters for YOLO and settled on epochs, image size, learning rate, momentum, and weight decay as the starting hyperparameters to tune. Descriptions for these hyperparameters can be found below (Ultralytics, 2024):

* **Epochs:** Total number of epochs, where each epoch represents a full pass over the entire dataset.
* **Image Size:** Target image size which all models will be resized to prior to training. Affects model accuracy but can also increase computation complexity.
* **Learning Rate 0:** Initial learning rate. Influences how rapidly model weights are updated.
* **Momentum:** Influences how quickly past gradients are incorporated into the current update.
* **Weight Decay:** Regularization technique that prevents overfitting.

Unfortunately, I was unable to do proper hyperparameter tuning by doing a grid search as I had originally intended. My grid search code worked as far as I could tell, but the code kept crashing when I tried to run it overnight because of various settings on my system. Because of this I elected to focus on one hyperparameter at a time to see what worked best on its own. I then combined the observations from these tests to make one model. This approach didn’t give me the optimal model that I might have otherwise achieved by using a grid search, but it did give me the opportunity to learn more about some of the hyperparameters for YOLO and see how changing these parameters changed the overall accuracy of the predictions

I started by experimenting with image size because I thought that it would be one of the main things that changed accuracy. I was correct that the accuracy would change by increasing the image size, but it changed in the wrong direction. Increasing image size from 640 to 960 increased the training time by about 30 times. However, this did not yield any favorable results. The overall f1 score was lower, mainly because the model started predicting background areas to be bee cells. This model was actually worse than the baseline model that I had trained earlier, so I discarded it and moved on.

A graph of a graph

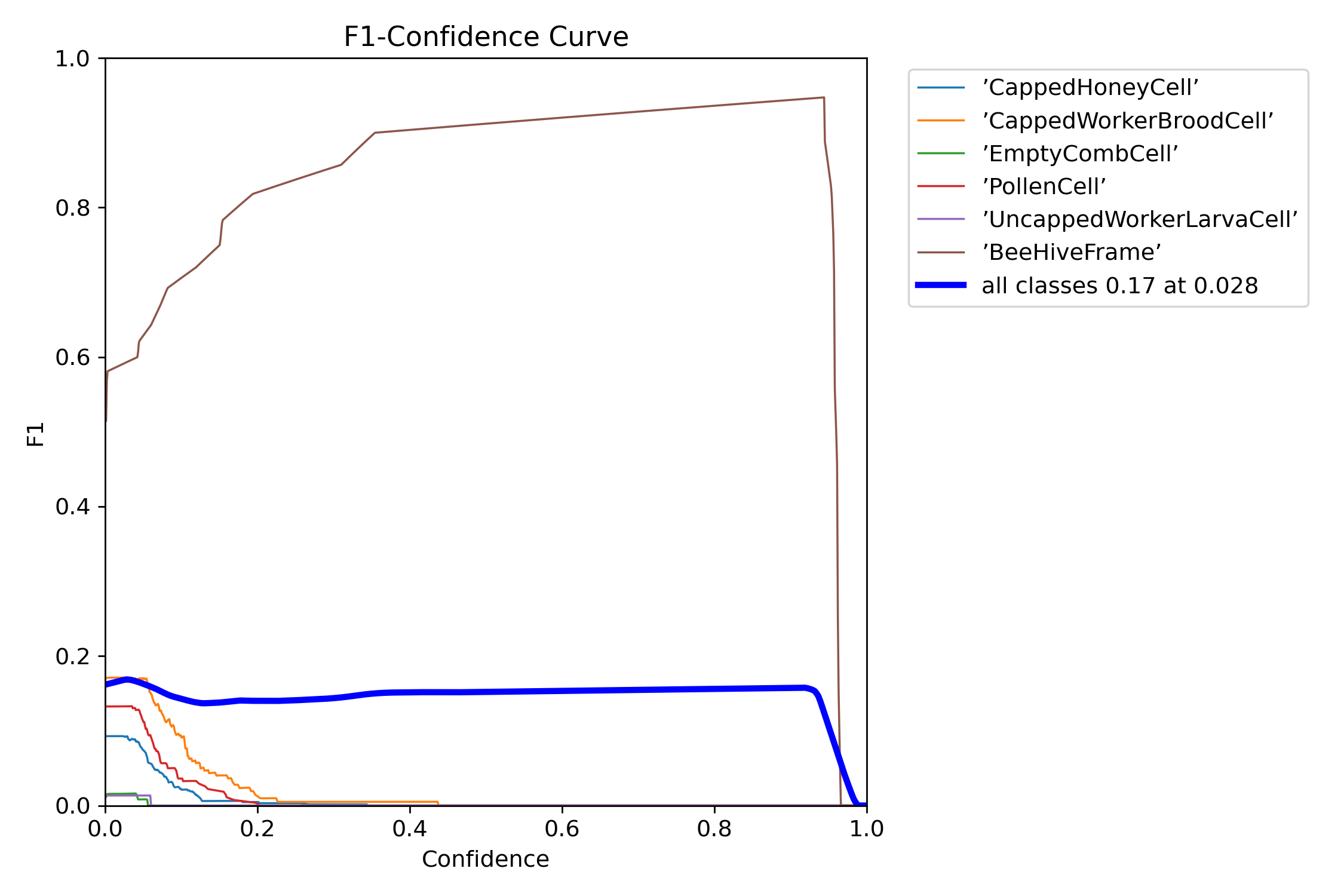
Description automatically generated with medium confidence

The next hyperparameter I tested was learning rate (lr0), which changes the rate at which the model learns. I tested the values 0.001, 0.01, and 0.02. When I used a learning rate of 0.001 my model stopped early as there was no improvement. I tried training it with the default YOLO model and it converged after about 280 epochs, but with slightly worse results. The same things happened with the other values I tried: no improvement when training on top of the previous best model and worse results when training on the default model. I elected to move on from this hyperparameter to see if other parameters yielded more promising results.

I decided to test out weight decay next. I chose weight decay because it is supposed to help with overfitting. I thought that maybe the model was overfitting to the train data because there was so little of it, causing poor performance on the validation set. I started with a value of 0.0005 and immediately began running into the same problems as before. At this point I didn’t know what to do. It seemed that I had hit the limit of what YOLO was capable of in terms of this dataset.

After taking a break from model training and hyperparameter testing to do some research, I realized that I had only used the YOLO nano model up to this point in the assignment. YOLO also has a small, medium, and large model available. Once I realized this, I switched over to the large model to get a benchmark to see if this was something worth pursuing. Because of the incredibly long computation time on the large model, I was only able to do 100 epochs. The results from this model were even more disappointing than the ones from the nano model. Even though it had only been 100 epochs, I had expected this larger model to have better performance. However, the performance on this one was even worse than on the nano model with 100 epochs. As I pondered on this problem, I realized that 100 epochs was probably enough time to get okay backpropagation on a smaller, simpler model like the nano model, but was not enough for a large model. The large model likely has a much higher ceiling, but its floor is also low because of the complexity of the model. I didn’t have the time or compute resources to run the large model for more epochs because just 100 epochs took about 2 hours on my pc, so I elected to run the small model for 500 epochs to compare to the nano model at 500 epochs.

The F-1 score for the large model at 100 epochs can be found below.



I again had little luck with the small model. It only trained for 242 epochs before stopping because of no improvement. It was only able to predict about 2% of pollen cells correctly. I am not sure why these models struggled so heavily on this dataset.

Overall, I’d call this research project a bit of a failure. I am not sure what else could have been done to improve the quality of the predictions made by these models. I would have loved to have access to a better computer so that I could run the larger models for more epochs, but with my situation that was impossible. In doing research at the end of this project, I stumbled upon a website that explained some of the challenges that I faced because of the choice to use YOLO v8 instead of YOLO v7 (Dupont, 2024). This new iteration uses an anchor-free design. In past YOLO iterations anchors were keys for locating objects. This new design is more versatile for detecting objects with varying shapes but also requires more training images, especially diverse ones, to predict accurately. If I were to come back to this project I would use YOLOv7 to make up for the lack of data.

# References

Dupont, M. (2024, May 3). *Boosting Accuracy in Object Detection: Tips for YOLOv8*. Retrieved from Labelvisor: https://www.labelvisor.com/boosting-accuracy-in-object-detection-tips-for-yolov8/

The Linux Foundation. (n.d.). *PyTorch*. Retrieved from https://pytorch.org/docs/stable/notes/windows.html#multiprocessing-error-without-if-clause-protection

Ultralytics. (2024). *Configuration*. Retrieved from Ultralytics YOLO Docs: https://docs.ultralytics.com/usage/cfg/#train-settings