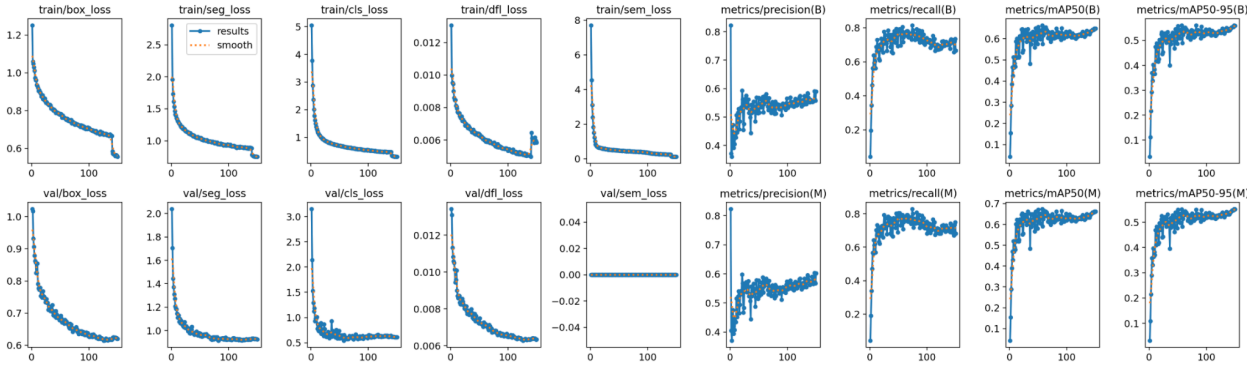
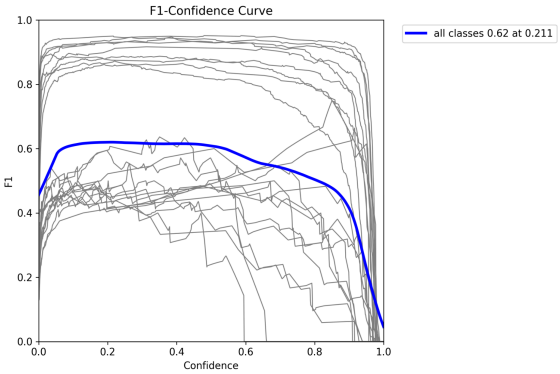
Car Review Segmentation Report

Detector Choice and Configuration

For the detector, I chose to go with the YOLO26 segmentation model and finetune the model on the provided car parts dataset from ultralytics. The fine tuning was necessary as the original segmentation did not have many car parts in its classes for detection so by training it I was able to add the 23 specific car parts in the dataset and get better results specific to this use case. It was finetuned for 150 epochs. The training curves are shown below.





For the confidence threshold of the model, I tried to stick to the value that maximized the F1 score on the right. This in theory would have best balanced the accuracy of the predictions as well as ensuring that all the available parts do get detected. However in reality I had a lot of missing detections or inaccurate detections that added on a lot of noise down the pipeline. This was a sever limitation in generating accurate intervals that fully encompassed the possible timestamps, and could perhaps be better tuned with more epochs (as the validation and training have not yet begun to diverge). Another problem was this model had significant trouble distinguishing between the left side and the right side of the car. This would result in an incorrect classification, which further harmed the matching accuracy between the query and retrieval.

Video Sampling

In class we were told that we only needed to analyze the exterior review section of the video which lasted from 18:39 to 24:45. I used ffmpeg to download this youtube video and then extracted the frames from the clip at 1 fps. I judged this to be a good balance between "resolution" and compute time since the trade back of a higher fps is faster inference for less accurate intervals.

Image to Video Matching Logic

First I ran the finetuned yolo models to classify labels of each frame that I had extracted. As per the requirements I created a parquet file with video_id, timestamp, class_label, bounding_box, and confidence score for each frame. Creating this file saves a lot of time on compute since I only have to calculate these labels once and I can run multiple experiments to evaluate the best parameters. Next, I loaded this file with Pandas and performed a bunch of filtering on it to try to correct the model's errors.

Confidence averaging:
This was the first filtering I performed. Essentially, I converted the data set into a table with the rows being the timestamp in order, and the columns being classes. The values were the confidence score of each class. Then I applied a rolling window across the time access in an attempt to smooth out the detections over time. I tried two windows, an averaging window and a max pool window. The averaging window would try to solve both false positives and negatives by looking at surrounding timestamps. The max pool would try to fix false negatives by putting more trust in the strong confidence levels. I ended up going with max pool because my process was unable to find many matches so I deemed lowering confidence levels was the problem.

Binary Mapping.
I then took this table and converted it to a binary table with 1 representing detection and 0 representing no detection. THe criteria was simply whether it was below or above a threshold.

Interpolation
Next I tried to salve the problem of the sparse false negatives I was detecting everywhere. Essentially if there was an n number of 0s in between 1s, I would change those 0s to 1s.

For matching I simply ran YOLO on the query image and got a list of classes. Then I compared each timestamp in the interpolated binary table to see whether they had a certain number of shared classes. Then I grouped together these "matches" to create a list of possible clips. I performed the same interpolation to bridge any short gaps in between clips to create a large clip

chunk. I also performed some filtering to get rid of short clips. Ultimately I returned the clip with the largest length, which somewhat made the filtering useless.

Limitations/Failure Cases

By far the thing that contributed most to the failures was the impreciseness of the model. This could have been due to not tuned hyperparameters, but I also suspect that the video was not completely in distribution of our finetune dataset. This is because I would often see it start to label the person presenting with car parts (with high confidence too). THis along with it's inability to distinguish left and right and often missing parts led to an incredibly difficult matching process.

There are a couple images in the query set that I was unable to find a match of significant length for as well. This could maybe be improved with more specific tuning on my hyperparameters as well.