Shreyon Roy
sr5655
Artificial Intelligence CS-GY 6613
Section INT2
Spring 2026
Assignment 2

Assignment 2 Report

This assignment builds a visual retrieval system where given a query image of a car exterior component, the system retrieves relevant video clips where that component appears. So the first task is to use an object detector that can extract semantic structure from various frames of a video. I used YOLO as my object detector and specifically YOLOv26 fine tuned on a car parts segmentation dataset. I used YOLOv26 because it is simple and efficient for small object detection. It has structured outputs that are great for indexing. YOLO comes from ultralytics so I installed that. Then I initialized a model from yolo26n-seg.pt. After that, I fine tuned the model for 10 epochs on car parts using carparts-seg.yaml. This allowed the model to identify car parts given images of a car's exterior. I also used an image size of 640 while fine tuning my model. In order to make sure the fine tuning process ran in a quicker amount of time I used A100 GPU which is very efficient for these purposes. After fine tuning, the model was used to make predictions on each frame image of the video. I used a confidence threshold of 0.25 because it helps the model detect as many potential objects as possible. Since there are many small objects in the exterior of the car, it is crucial that those parts are not missed. It also reduces false negatives. While running predictions on each of the frame images, I set streaming to True which helped to process the data incrementally because there were a large number of files. I also chose not to save any of the predicted annotated images by setting save to False because it would take

up a lot of time and resources. As a result, the object detections from the images were semantically meaningful because they labeled car parts like wheel, back glass, left mirror, right mirror, front door, etc.

For video sampling, I followed the instructions outlined in the assignment description. I installed ffmpeg yt-dlp. I then used it to download the source video given the URL. I did not specify any clips in particular but I just sampled one frame per second from the video. I decided to use 1 FPS because I wanted a good amount of coverage of exterior components and help with retrieving continuous video segments. Since the Toyota review video did not move at a very rapid speed, I did not feel the need to have it cover more than 1 frame per second. Otherwise, there would be many redundant frames and impact the computation efficiency. I gathered 2794 frames. Additionally, because I sampled with 1 frame per second, I knew this would help to determine timestamps for future retrieval because each frame could signify 1 second that elapsed which could simplify the calculations.

The way that the image to video matching worked is I first performed object detection on the frames from the given YouTube video. This gave information about which frames contained which exterior car parts based on boundary boxes and class labels. This same object detection process was conducted on query images from the hugging face parquet file. Boundary boxes and class labels were produced. Then the query images which contained a specific part were matched with any frames that contained the same part. The class label was identified in the query image and searched for throughout the dataframe of object detection results from the video. The object detection results from the video act as a database and the class labels from the query images act as READ operations from that database. The frame indexes where there are matching car parts are used to generate a subset of matched frames. Based on which frames contained a specific

part, the timestamps were recorded, and then the timestamps were combined to indicate a continuous time segment. Within my logic, I had a gap duration boundary which prevented timestamps containing the same car part from being combined if their timestamp difference exceeded 3 seconds. This would help distinguish between two separate time segments. In this way, this gap duration helps time segments achieve temporal coherence.

One failure of my retrieval system was that certain class labels were not specific enough. For example, when I fine tuned my model, it was able to identify specific parts of the exterior of a car but there were some parts like “wheel” which are not specific. As a result, if any wheel is detected in the query image, it will retrieve all frames where a wheel appears. They may not be in the same position of the wheel. Additionally, I noticed that in a large portion of the video, there is a lot of focus on the interior of the car, so there are large sections of the video that are skipped because the query dataset only features exterior components. Additionally, one limitation I noticed is that if there are any changes in lighting and camera angle, then this might cause mismatches amongst detected components. Furthermore, sampling at 1 frame per second was more feasible because the video did not move at a very fast pace. However, there could be instances where there might be fast camera motion to show something which might get skipped because it did not appear in the frame.

Overall, this strategy of using semantic object part detection is effective in creating an image to video retrieval system. Because the video is converted to frames, it is simpler to predict which car components appear in each frame. These predicted components act like a database where images can act like a READ feature to extract which portion of the video a specific part appeared. This is a simple but effective strategy for matching and searching.