CAP 6419: SAM Segmentation with Bounding Box Input Automation

Devansh Sharma

September 11, 2025

1 Introduction

Image segmentation is a fundamental task in computer vision. Traditional models require extensive, pixel-perfect training data. The Segment Anything Model 2 (SAM2) paradigm changes this by allowing users to generate masks by providing prompts, such as points, boxes, or text. This project focuses on fully automated methods for generating these prompts, eliminating the need for manual interaction.

The primary goal of this assignment is to design and evaluate two distinct pipelines for prompting SAM2 to segment "people" and "vehicles" from the CamVid dashcam dataset. We establish a baseline using a text-prompted Vision-Language Model (VLM) and propose a novel pipeline using a specialized, pre-trained object detector to test the hypothesis that specialist models provide superior spatial prompts.

2 Methodology

2.1 System Architecture and Pipeline

The project is built on a modular, two-stage pipeline. The first stage generates bounding box proposals, and the second stage uses these proposals to generate segmentation masks. This design allows for different detector components to be easily swapped and compared.

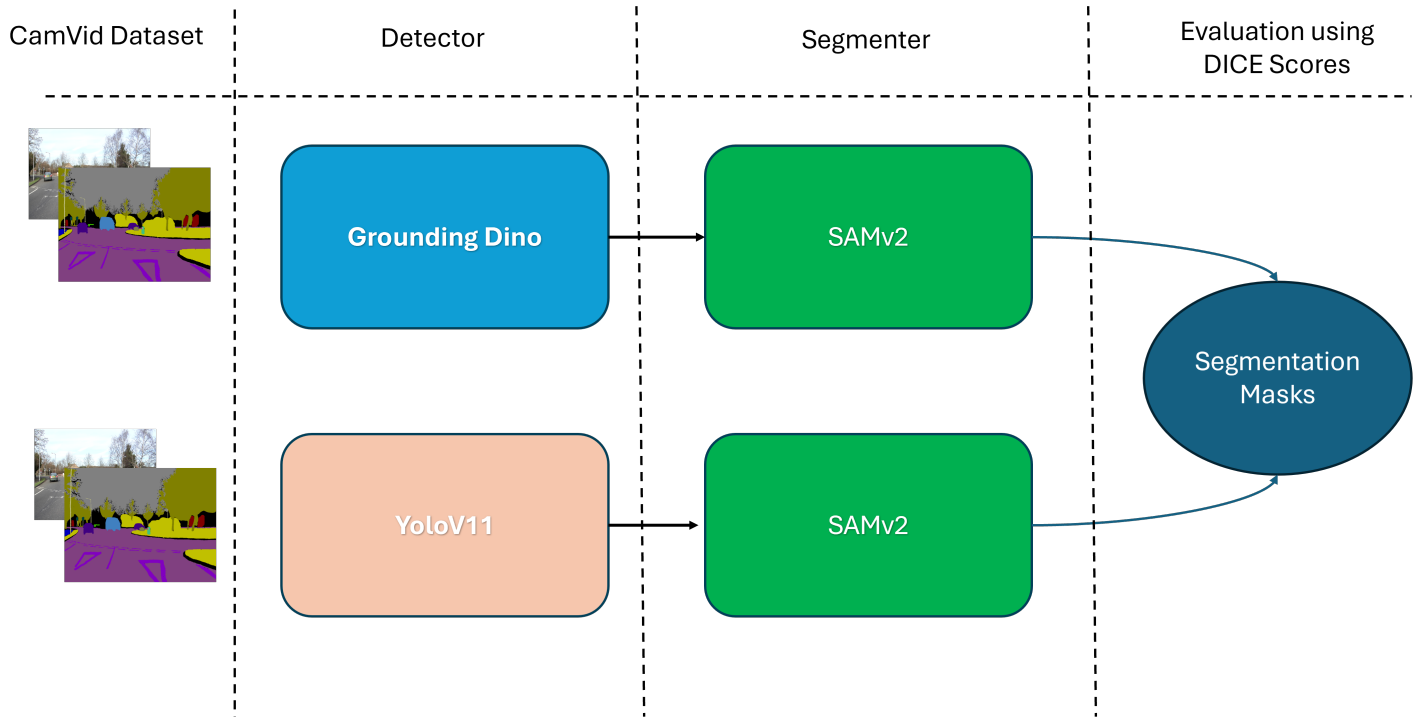

Figure 1: The two-stage pipeline. A detector (DINO or YOLOv11) generates bounding boxes, which are then passed to the SAM2 Segmenter to produce the final masks.

The entire system was refactored into a class-based structure, including a ‘Detector‘ class for each detection model and a ‘Segmenter‘ class for SAM2, to ensure code reusability and a clean workflow managed by an ‘evaluate.py‘ script.

2.2 Component Models

2.2.1 Grounding DINO

Grounding DINO is a zero-shot object detector that unifies object detection with language. It takes an image and a text prompt (e.g., "a person on a bicycle") as input and outputs bounding boxes corresponding to the objects described in the text. This served as our flexible, text-based baseline prompter.

2.2.2 YOLOv11

You Only Look Once (YOLO) is a family of state-of-the-art, single-stage object detectors known for their exceptional speed and accuracy. We used the YOLOv11x model, which is pre-trained on a large dataset of common objects. Unlike Grounding DINO, it does not take a text prompt but instead detects objects from a fixed set of classes. We filter its output to keep only detections relevant to our task.

2.2.3 SAM2

The Segment Anything Model 2 (SAM2) is a foundation model for segmentation. We used the 'sam2_l.pt' weights integrated via the 'ultralytics' library. It takes an image and a prompt (in our case, bounding boxes) and produces high-quality, pixel-perfect segmentation masks for the prompted objects.

2.3 Implementation Details

The 'supervision' library was used extensively for data handling. Bounding boxes from both detectors were standardized into 'supervision.Detections' objects. These objects were then enhanced by attaching the output masks from SAM2. For visualization, 'supervision''s 'BoxAnnotator' and 'MaskAnnotator' were used to draw the final outputs.

2.4 Evaluation Protocol

The performance of each pipeline was measured on the 100-image validation set of the CamVid dataset. The evaluation metric used was the **Dice Score**, which measures the overlap between the predicted mask and the ground truth mask. The formula is:

$$\text{Dice Score} = \frac{2 \times |A \cap B|}{|A| + |B|}$$

where A is the set of pixels in the predicted mask and B is the set of pixels in the ground truth mask. A score of 1 indicates a perfect match.

3 Results

Both pipelines were run on the full validation set. The YOLOv11x pipeline demonstrated significantly better performance than the Grounding DINO baseline.

Pipeline	**Average Dice Score**
Pipeline A (Baseline: **DINO** + SAM2)	0.4253
Pipeline B (New: **YOLOv11x** + SAM2)	**0.6321**

Table 1: Final performance comparison on the CamVid validation set.

4 Discussion

The results clearly indicate that the YOLOv11x-prompted pipeline is the superior method for this task. The primary reason is likely the **domain specialization**. YOLOv11x is a highly optimized specialist for common object classes, and although it was not fine-tuned on CamVid, its pre-training allows it to produce consistently accurate bounding boxes for "people" and "vehicles".

Grounding DINO, while incredibly flexible, is a generalist. Its performance is sensitive to the text prompt and the specific visual characteristics of the dataset, which may differ from its broad training data. On the challenging, lower-resolution images of the CamVid dataset, it produced less reliable bounding boxes, which in turn gave SAM2 poorer prompts to work with. This illustrates the classic trade-off: the specialist outperforms the generalist on its specific task.

Figure 2: Example output from the DINO pipeline.

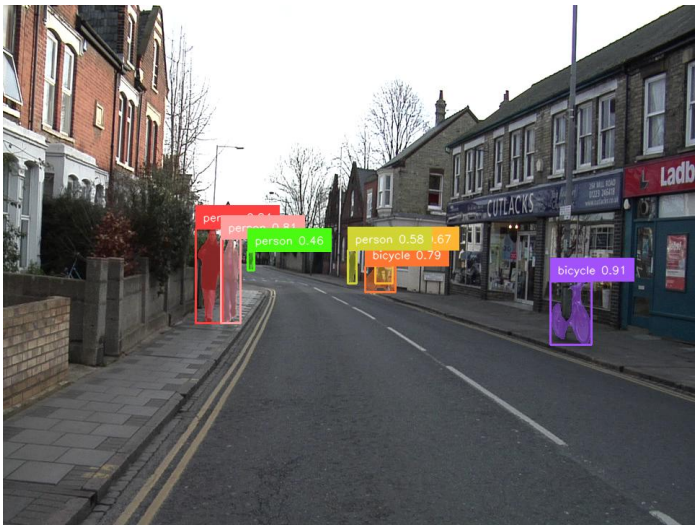

Figure 3: Example output from the YOLOv11 pipeline.

5 Conclusion

This project successfully designed and evaluated two automated pipelines for prompting the SAM2 segmentation model. We established a baseline using the text-based Grounding DINO and proposed a method using the class-based YOLOv11x detector. The evaluation on the CamVid dataset showed that the YOLOv11x pipeline achieved a 48.6% higher average Dice Score than the baseline. This confirms our hypothesis that for tasks with well-defined, common object classes, a specialized object detector is a more effective prompter for foundation segmentation models than a zero-shot text-based model.

A Code Snippets

A.1 YOLOv11 Detector Class

```python
# In yolo_detector.py
def predict(self, image_path: str, conf_threshold: float = 0.4) -> sv.Detections:
    # Perform detection with the specified confidence threshold
    results = self.model(image_path, conf=conf_threshold, verbose=False)[0]

    # Convert the results to a supervision.Detections object
    detections = sv.Detections.from_ultralytics(results)

    # Create a boolean mask for our target classes
    mask = np.isin(detections.class_id, self.target_class_ids)

    # Apply the mask to filter the detections
    filtered_detections = detections[mask]

    return filtered_detections
```

Listing 1: The predict method of the YOLOv11Detector class, showing filtering logic.