POLYTECHNIQUE
MONTRÉAL

INF6804 Computer Vision

Project
Practical Assignment 2

Mohamed Aziz Younes - 2417117
Mathieu La Brie - 2403325

Polytechnique Montreal
2025-05-20

Contents

1 Presentation of the two compared approaches

1.1 Introduction

Segmenting regions of interest in videos is critical for numerous computer vision applications, such as scene understanding, action recognition, and video editing. Two advanced methods for video segmentation are YOLOv8 and CLIPSeg. YOLOv8 is an efficient object detection and segmentation model known for its speed and accuracy, integrating real-time instance segmentation. In contrast, CLIPSeg employs contrastive language-image pretraining, allowing zero-shot segmentation driven by textual prompts without the need for task-specific training.

1.2 YOLO (You Only Look Once)

The YOLO algorithm is a state-of-the-art object detection and segmentation framework widely used in computer vision tasks. Unlike traditional detection methods similar to those used in TP1 that apply classifiers or detectors at multiple locations within an image, YOLO processes the entire image at once, predicting the bounding boxes, class probabilities, and segmentation masks in one forward pass. This approach enhances both detection speed and accuracy.

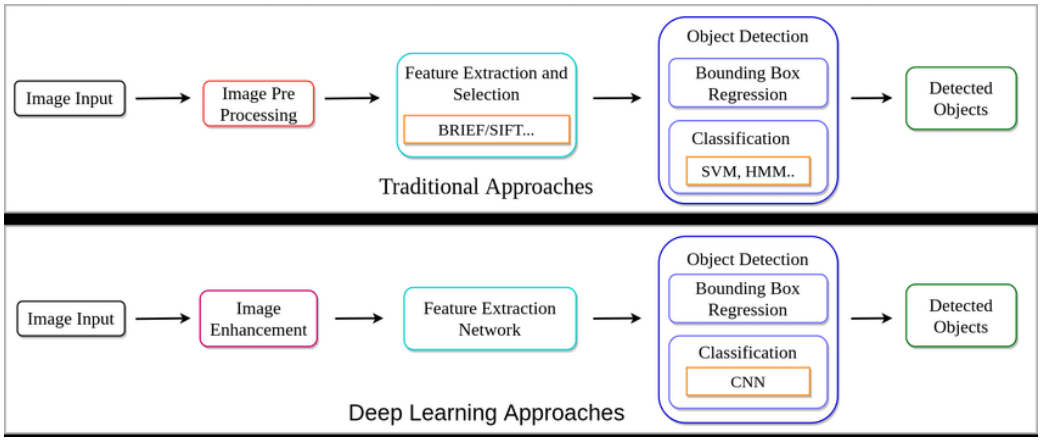


Figure 1: Traditional vs DeepLearning approaches [1]

1.2.1 Basic Principles of YOLO

To start, YOLO preprocesses input images by resizing them to a standard size (e.g., 640x640 pixels) to ensure standard input dimensions for the neural network. Unlike some traditional methods, YOLO divides the input image into a grid. Each grid cell predicts bounding boxes, class probabilities, and, in the case of segmentation tasks, pixel-level masks at the same time.

Each grid cell then predicts several bounding boxes with associated confidence scores, higher scores indicate how certain the model is of its prediction. The boxes also include coordinates and dimensions that are normalized relative to the grid. Finally, the model predicts the class of the object in the bounding box.

1.2.2 YOLOv8 Segmentation

Backbone: In YOLOv8, the backbone serves as a CNN-based feature extractor responsible for processing input images and extracting meaningful representations. A commonly used backbone in YOLOv8 is CSPDarknet, a variation of Darknet that incorporates Cross Stage Partial (CSP) connections to improve gradient flow and computational efficiency. The backbone is designed to capture hierarchical spatial and semantic features by progressively downsampling the input image through convolutional layers, batch normalization, and activation functions. This process enables the model to learn rich feature maps that effectively represent objects at different scales. The extracted features are then passed to the neck and head of the YOLOv8 architecture for further processing, including feature fusion, object detection, and classification. By leveraging CSPDarknet or other efficient CNN-based backbones, YOLOv8 achieves a balance between accuracy and inference speed, making it well-suited for real-time object detection applications.

1.3 CLIPSeg Zero-Shot

CLIPSeg is a zero-shot segmentation framework that combines the vision-language capabilities of the CLIP model (seen in the last TP) with pixel-level segmentation tasks. Unlike traditional segmentation models, CLIPSeg can perform segmentation without any training data specific to the segmentation task, which can make it adaptable to new scenarios but can sometimes hinder its performance.

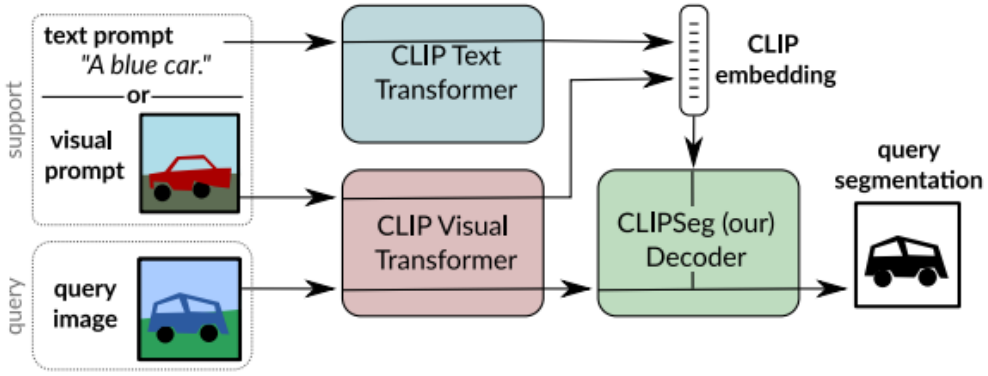


Figure 2: CLIPseg simplified. [2]

1.3.1 Basic Principles of CLIP

CLIP is pretrained on a massive dataset containing pairs of images and descriptive texts. This approach allows CLIP to associate images with corresponding textual descriptions.In order to train such a model, CLIP uses contrastive learning, where matching image-text pairs are encouraged to have similar embeddings while mismatched pairs have dissimilar embeddings.

1.3.2 CLIPSeg for Zero-Shot Segmentation

CLIPSeg uses textual prompts to guide segmentation, allowing users to specify objects or regions of interest through simple text queries. The model generates pixel-wise masks corresponding to these textual prompts without task-specific training. It also utilizes attention mechanisms within transformer architectures to identify relevant regions in the image that correspond to the input textual prompts. These attention maps directly translate into segmentation masks.

CLIPSeg has a transformer-based, encoder-decoder architecture. Its encoder is a pre-trained CLIP vision-language model based on the vision transformer (ViT-B/16) network. It generates an embedding and attention activations for a target image.

For the decoder, CLIPSeg stacks three standard transformer blocks that combine the target image embedding, its activations, and the conditioning prompt to output a binary segmentation mask.

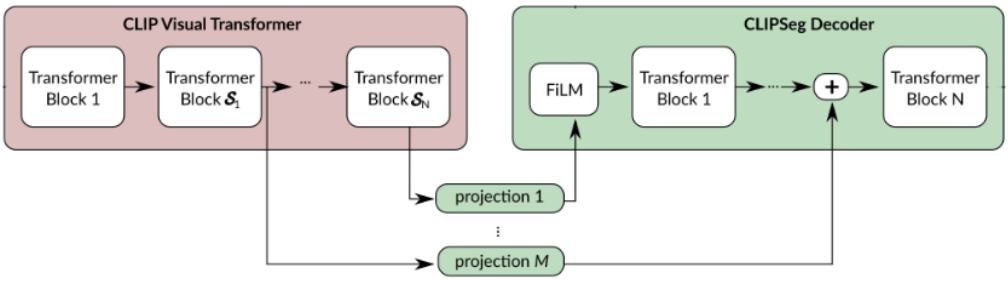


Figure 3: CLIPseg encoder decoder. [2]

Conditioning

Text or image prompts are additional multimodal inputs that the model must combine with a target image in some way. Influencing a model's results in this fashion, with additional input, is called "conditioning." For example, a popular conditioning technique for natural language processing (NLP) combines token position embeddings with input text embeddings.

Conditioning techniques that influence not just the first but every layer of the model give better results. CLIPSeg uses one such technique called feature-wise linear modulation (FiLM), proposed by Perez et al.[3]

2 Performance hypotheses in specific use cases

2.1 Use case 1: Segmenting small or distant objects

When focusing on segmenting small or distant objects, our hypothesis is that **YOLOv8 will outperform CLIPseg**. YOLOv8 is designed primarily for high-accuracy object detection. It uses advanced feature extraction and multiscale representations that make it very good at localizing and segmenting objects even under difficult conditions.

In contrast, CLIPseg incorporates CLIP's semantic understanding to guide segmentation tasks. Though this text-to-image alignment is quite good for ensuring semantic consistency, the method should be less specialized at capturing tiny or distant objects in cluttered scenes. Consequently, we anticipate that YOLOv8 will yield higher segmentation quality in scenarios requiring accurate handling of small or distant objects.

2.2 Use case 2: Detecting moving objects

When it comes to detecting moving objects, our hypothesis is that **YOLOv8 will outperform CLIPseg**. YOLOv8 excels at robust object detection, its architecture is optimized for real-time tasks, which makes it well-suited to track and segment objects that shift position and/or exhibit motion blur.

In contrast, similary to the first use case, CLIPseg relies on semantic alignment between image and text features. While this semantic consistency can mean it will outperform YOLOv8 for static segmentation task, it may struggle when objects rapidly change appearance or position. Consequently, we anticipate that YOLOv8's detection-focused framework will yield higher segmentation quality when capturing moving objects in dynamic scenes.

2.3 Use case 3: Comparing computational speed

When evaluating computational efficiency, we hypothesise that **YOLOv8 outpaces CLIPseg in speed**. YOLOv8's architecture is fine-tuned for low-latency detection and maintains high throughput, making it ideal for real-time applications.

In contrast, CLIPseg relies on a transformer-based pipeline that, while powerful for semantic alignment, can be more resource-intensive. This increased complexity tends to slow inference, particularly for high-resolution inputs or large batch sizes. Hence, in scenarios where rapid processing is vital, YOLOv8 is likely to deliver faster performance.

3 Description of experiments, datasets and evaluation criteria

3.1 Use case 1: Segmenting small or distant objects

For Segmenting small or distant objects, we used a CDNET 2012 video where objects appear small (e.g., vehicles far from the camera). This video also a challenge due to the number of objects needed to segment. This would make it especially hard for the CLIPseg method.

Here we use for qualitative and quantitative methods to compare the results. To begin, we run both models and compare the resulting masks visually to the ground truth and the original image. We then compare results using the average IoU score, precision, recall and f1-score.

3.2 Use case 2: Detecting moving objects

In this use case, we used the train station video where there are two humans. One is still and one is moving which makes it perfect for comparing the two methods.

Our evaluation criteria involves comparing the results for both the moving and still segmentation task simultaneously. Here we once again visually compare each of the 4 results to the ground truth and original video and then compare the average IoU score, precision, recall and f1-scores.

3.3 Use case 3: Comparing computational speed

For this use case, pretty much any video would do, so we chose the office video. Since there is no use in comparing the qualitative results in this scenario, we simply used the average inference times as well as their distribution in order to get a good understanding of the results.

4 Description of the implementations used

To compare fixed-class segmentation and zero-shot segmentation, we implemented both YOLOv8 and CLIPSeg for video object segmentation. This section details the implementation process, including model selection, data preprocessing, and parameter settings.

4.1 Implementation of YOLOv8 for Fixed-Class Segmentation

We utilized the `YOLOv8n-seg.pt` model, a lightweight version that balances performance and computational cost. The implementation was done using the Ultralytics YOLOv8 library in PyTorch. Each video frame was loaded using OpenCV and resized to 512x512 pixels to ensure compatibility with the model. The YOLOv8 model was then applied to detect and segment objects, extracting object masks and overlaying them on the original frame. Post-processing involved blending the segmentation results with the input frames for visualization and saving the segmented video for evaluation.

4.2 Implementation of CLIPSeg for Zero-Shot Segmentation

CLIPSeg was chosen as the zero-shot segmentation model due to its ability to generalize to unseen objects using textual prompts. The implementation used the Hugging Face Transformers library with the `"CIDAS/clipseg-rd64-refined"` model. Each video frame was first converted to RGB format and resized to 512x512 pixels for compatibility with the model. A textual prompt describing the target object was provided to guide the segmentation process. The CLIPSeg model then generated a pixel-wise segmentation mask based on the text input, which was subsequently binarized and overlaid onto the original frame. The final segmented video was saved for further evaluation.

4.3 Implementation Challenges and Modifications

Several challenges were encountered during implementation. For YOLOv8, tuning non-maximum suppression (NMS) was necessary to minimize redundant detections and improve segmentation accuracy. CLIPSeg's segmentation masks tended to be noisier than YOLOv8's, necessitating post-processing to achieve smoother contours. Despite these challenges, both implementations successfully demonstrated their respective strengths in video object segmentation.

5 Experimentation results

5.1 Use case 1: Segmenting small or distant objects

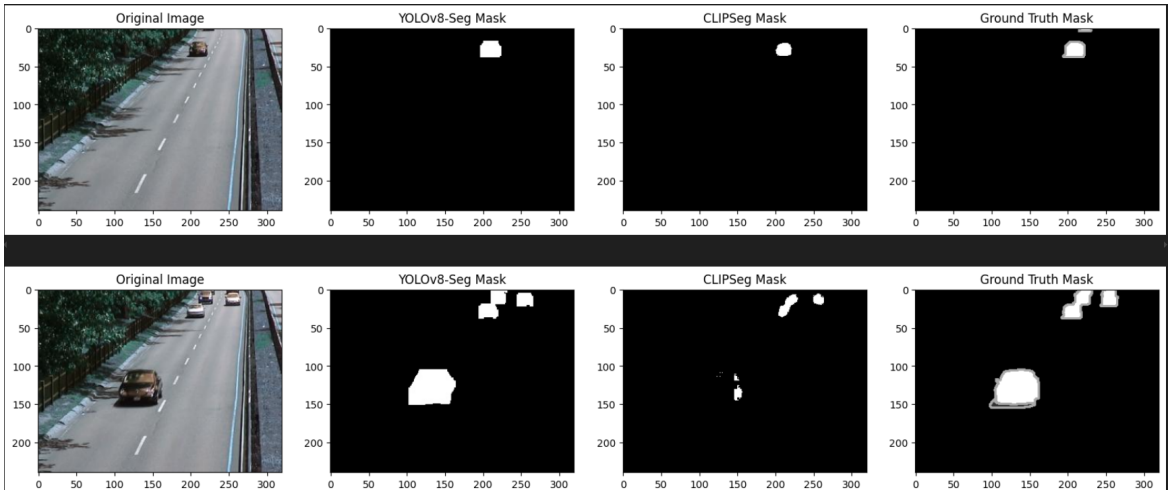
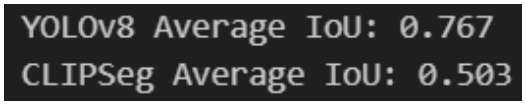


Figure 4: YOLOv8 vs CLIP-seg for small objects

Results above show the qualitative results comparing the YOLOv8 architecture to CLIP-seg. On the right is the ground truth mask. These results are reflected in the distribution of the IoU scores seen in the graph below.



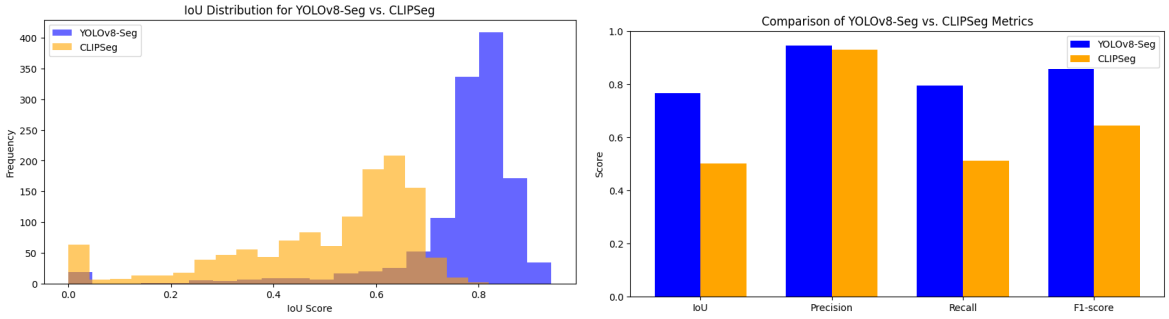


Figure 5: YOLOv8 vs CLIP-seg: Distant object segmentation scores

On the left is the distribution for the IoU scores for both the methods. The high value in for the CLIP-seg method (in orange) is due to the high number of missed objects in the segmentation task. These results are represented more clearly on the right, where a clear difference in scores can be observed no matter the metric used.

5.2 Use case 2: Detecting moving objects

The figure below compares the two different queries for the two methods with the ground truth. One thing to notice is that the YOLO model seems to detect another object in the background and that both the moving and still results are the same. In contrast, CLIP-seg only detects the moving objects and produces no masks for the "still" objects.

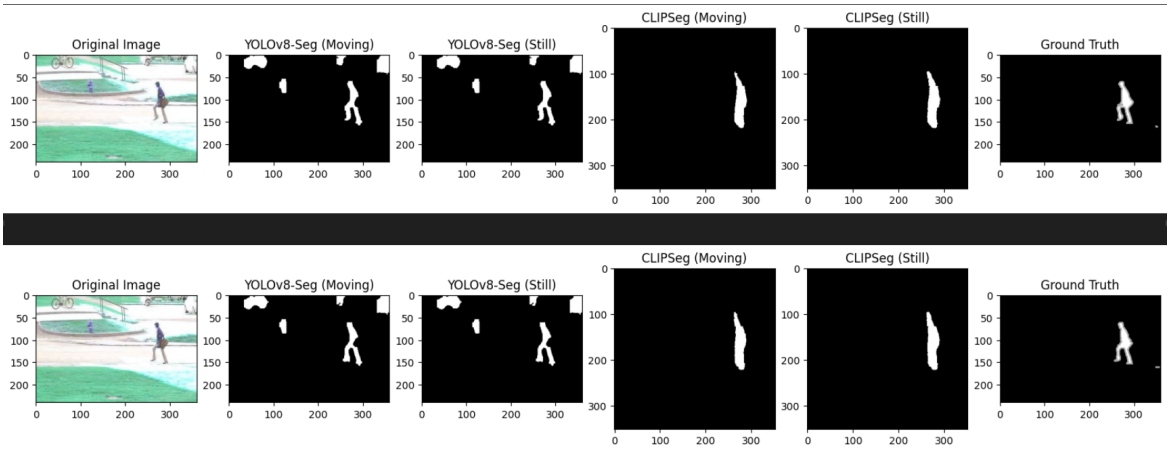


Figure 6: YOLOv8 vs CLIP-seg for moving and still objects

Using these results more quantitatively, below is a graph of the IoU score, Precision, Recall and F1-score for this segmentation task. Here, except for the still CLIP-seg results, we obtain much closer performance in comparison with the last use case.

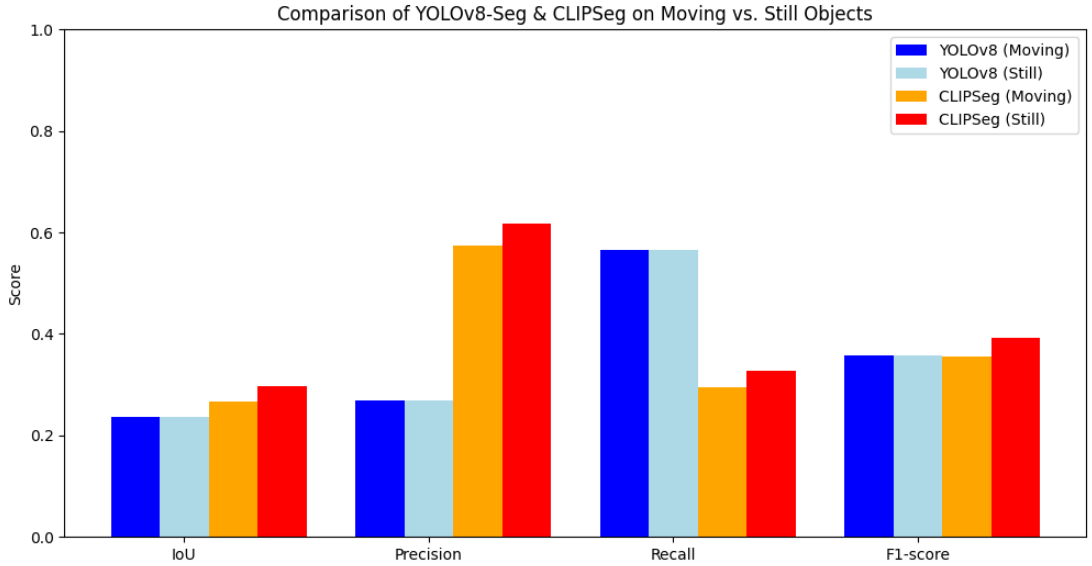


Figure 7: YOLOv8 vs CLIP-seg: Moving vs Still Scores

Finally, a graph of the distribution for the frequency of the IoU scores helps to put the obtained results in perspective.

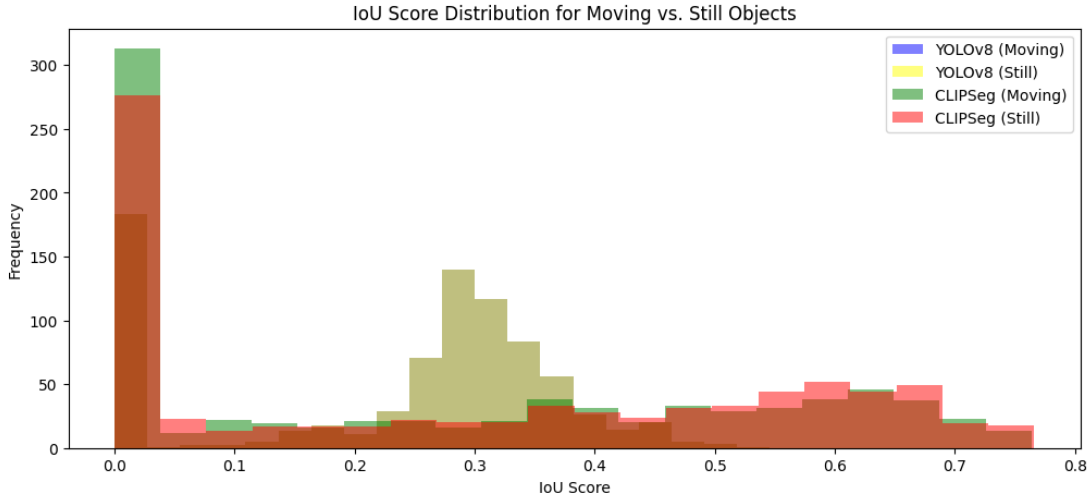


Figure 8: YOLOv8 vs CLIP-seg: Moving vs Still IoU Score Distribution

5.3 Use case 3: Comparing computational speed

For this criteria, we compare using both the average processing time (left) and its distribution (right). Here, there is no mistaking which model outperforms the other since the YOLOv8 model in blue is clearly better for this use case

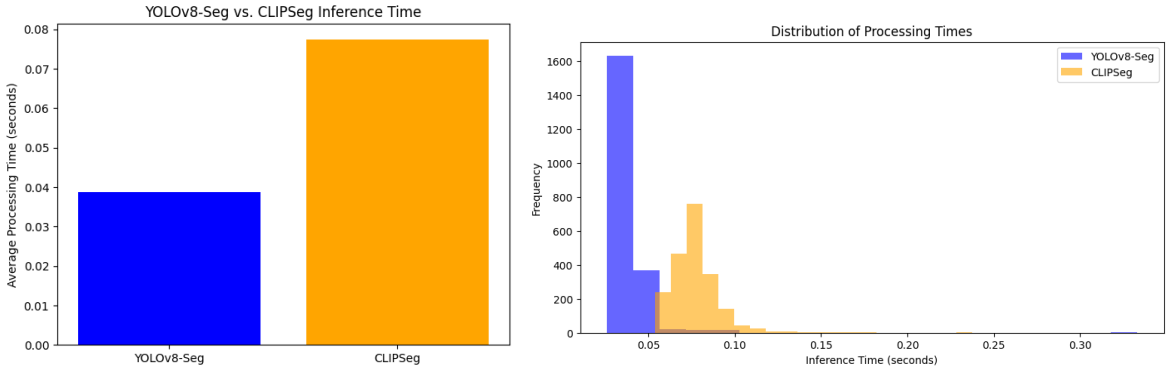


Figure 9: YOLOv8 vs CLIP-seg: Computational speed Scores

6 Discussion on results and prior hypotheses

6.1 Use case 1: Segmenting small or distant objects

For this hypothesis, we correctly predicted that YOLOv8 would outperform CLIP-seg when segmenting small and/or distant objects. From the qualitative results, one can already see how CLIP-seg struggles to detect segment multiple objects. This leads to an extremely poor average performance overall. For YOLOv8, this task was pretty easy and the model manages to get very accurate results.

These results may be due to choosing a video that over complicates things for the CLIP model. Indeed, though the results would still have been worse for the CLIP model, a video containing...

6.2 Use case 2: Detecting moving objects

When detecting moving objects, the CLIP model surprisingly performed as good if not better than the YOLOv8 model. Its strong semantic segmentation provided more accurate results than the YOLOv8 model, who wrongly classified the background for both moving and still results. The fact that both results are the same for the YOLO model may show that it cannot make the distinction between moving and still objects, further cementing its performance for this use case.

It seems that our original hypothesis may have overestimated YOLOv8's capacity to see if an object is in motion when the image quality isn't very good. The fact that the objects in the scene aren't moving fast enough to cause blur may also contribute to this assertion.

6.3 Use case 3: Comparing computational speed

In comparing the computational speed, we correctly hypothesised that YOLOv8 would be the better model for this metric. This reasoning is not only backed by the obtained results, but was also be expected by the different uses that each different model has. Since YOLOv8 was trained for real-time segmentation, its no surprise that it performed the tasks twice as fast on average compared to CLIP-seg.

7 Conclusion

Overall, these experiments highlight differing strengths between YOLOv8 and CLIP-seg. While YOLOv8 excelled at segmenting small or distant objects and offered faster inference, CLIP-seg demonstrated impressive semantic fidelity, performing notably well in detecting moving objects. These findings suggest that choosing between the two models should be guided by specific task requirements: YOLOv8 is more suitable for scenarios demanding real-time performance and accurate detection of tiny or faraway targets, whereas CLIP-seg may be advantageous in situations where semantic consistency and nuanced object interpretation are paramount. Future work could explore hybrid approaches that combine the speed and localization prowess of YOLOv8 with the rich semantic understanding of CLIP-based methods.

List of Figures

References

[1] "Pipeline comparison of traditional and deep learning approaches for object detection." https://www.researchgate.net/figure/Pipeline-comparison-of-traditional-and-deep-learning-approaches-for-object-detection-fig3_353596790, 2021. Accessed: 2025-03-11.

[2] "CLIPseg simplified." ://arxiv.org/pdf/2112.10003.

[3] "FiLM: Visual Reasoning with a General Conditioning Layer." https://arxiv.org/abs/1709.07871. Accessed: 2025-03-03.