UNIVERSITÀ DI PISA

Msc in Artificial Intelligence and Data Engineering

A Hybrid Approach for Semantic Segmentation: Integrating SAM Model (2023_C4)

Symbolic and Evolutionary Artificial Intelligence project

Giuseppe Martino[1] and Salvatore Arancio Febbo[2]

[1]g.martino10@studenti.unipi.it
[2]s.aranciofebbo@studenti.unipi.it

Contents

1 Introduction

Semantic segmentation is one of the key challenges in the field of computer vision and plays a fundamental role in image analysis and understanding.
It consists in dividing an image into homogeneous regions and assigning a specific label to each region representing the object category. This technique provides detailed information about the content of the image, enabling discrimination between different object classes such as people, cars, buildings, roads, and background.
Despite significant advancements, semantic segmentation remains a challenging task due to the variety of shapes, sizes, backgrounds, and lighting conditions present in images.
Several techniques and approaches have been developed to address this problem, most of which include the use of Convolutional Neural Networks (CNN). In recent years, however, approaches based on Vision Transformers are increasingly obtaining excellent results

2 DPT

DPT (DensePredictionTransformers)[1] is a segmentation model released by Intel in March 2021 which uses vision transformers to perform the semantic segmentation task.

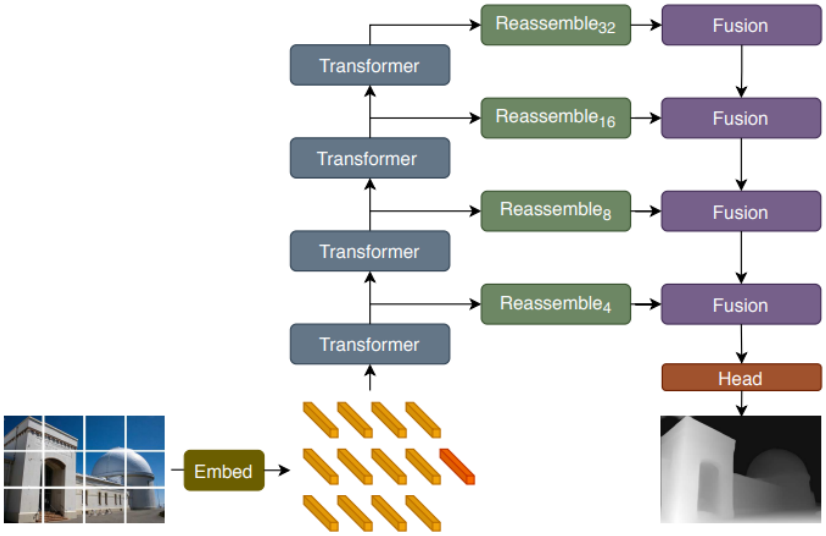


Figure 1: DPT architecture overview. Image taken from [1]

The images are divided into patches, which are then tokenized in this way: the patches are flattened into vectors and the embedding patches are found using either a linear projection, or taking as embeddings the features obtained by applying ResNet50 to the images and using it as feature extractor. A learnable position embedding is then concatenated to each of these embeddings to add spatial information. An additional token is then added, i.e. the redout token, which represents the global image and can be used for classification. Embeddings then go through several transformer layers and then are reassembled from different stages into image-like representations at different resolutions. Finally, fusion blocks use residual convolutional units to merge and combine feature maps. The size of the final representation output from this convolutional network is half that of the original image and an output head is added which provides the final prediction.

Figure 2: Original and segmented image obtained from DPT

3 SAM

The Segment Anything Model (SAM)[2] is a segmentation model developed by Meta Research and released in April, 2023. It is designed to segment any object. This includes *stuff and things*.

The SA-1B dataset which was used to train the model was specifically constructed, using **11 million** licensed high-resolution images (with an average size of 3300 x 4950 pixels). To obtain the masks for these images, they employed their own **data engine** capable of generating a total of **1.1 billion masks**.

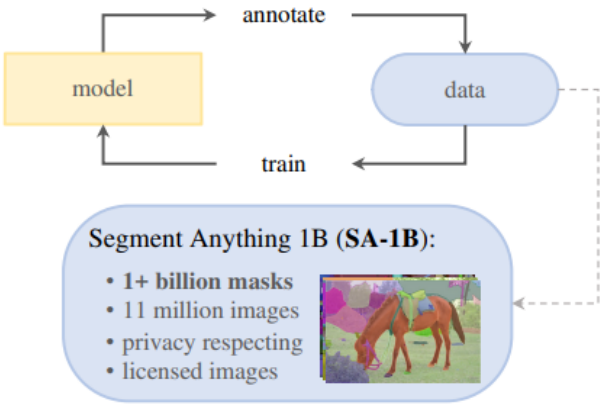


Figure 3: Data Engine. Image taken from [2]

This data engine consists of three stages: *assisted-manual*, *semi-automatic*, and *fully automatic*. It uses the same SAM (Segmentation Annotation Model) to create annotations, and the annotation process is referred to as "model-in-the-loop."
In the assisted-manual stage, a team of annotators manually annotates the images using a tool powered by SAM. At the beginning of this stage, SAM was trained using publicly available segmentation datasets.
Then, after sufficient annotation, SAM was retrained using only the newly annotated masks. In the second stage (semi-automatic), the most confident masks were generated with SAM, and annotators focused on annotating the remaining unannotated objects.
In the final stage, the annotation was fully automatic: SAM was provided with a grid of

32x32 points for each image, and for each point, a set of masks was predicted.

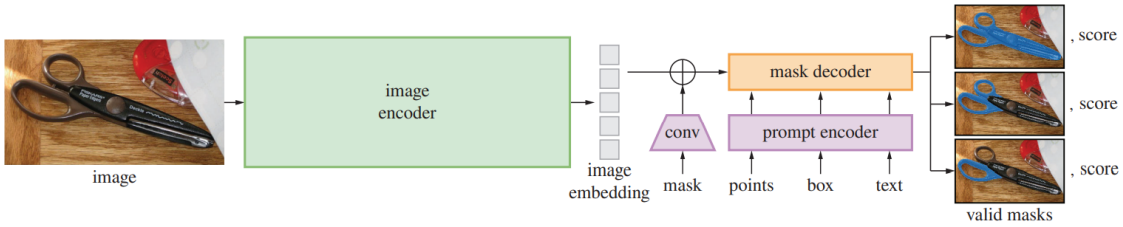


Figure 4: SAM architecture overview. Image taken from [2]

As reported in the original paper of SAM, the three main components of SAM are: an image encoder, a prompt encoder and a mask decoder.
A pre-trained Vision Transformer was used for the decoder to produce image embeddings. These features are then passed into a decoder head that also accepts prompts embeddings generated by the prompt encoder. Prompts could be a rough mask, labeled points, or text.
"The mask decoder efficiently maps the image embedding, prompt embeddings, and an output token to a mask. This design, employs a modification of a Transformer decoder block followed by a dynamic mask prediction head.[...] After running two blocks, we upsample the image embedding and an MLP maps the output token to a dynamic linear classifier, which then computes the mask foreground probability at each image location".[2].

Figure 5: Original and segmented image obtained from SAM

4 Objective: optimizing Semantic Segmentation through SAM model integration

Semantic segmentation models such as DPT can classify the regions into which an image is divided by associating a label with a region of an image, but they lack accuracy, especially at the edges, so the regions that are identified are not always very accurate.

SAM, on the other hand, can provide very accurate masks, but it does not assign a category to each mask and often segments objects too much, resulting, for example, in masks for parts of objects that are not significant

So the goal is to improve the semantic segmentation obtained using the DPT model by using the masks provided by SAM, so as to achieve more accurate semantic segmentation.

4.1 Dataset

The dataset used in our experiments is PASCAL VOC 2012. It is a dataset widely used as a benchmark for object detection and semantic segmentation. It contains 20 object categories plus the background.
To test our results, we used the validation set which contains 1449 images, and for each image is available pixel-level segmentation annotation. These were used to calculate the metrics used to evaluate segmentation.

4.2 Procedure Followed

First we cloned the **DPT** repository [3], set the dependencies as described in the repository and downloaded the model weights (DPT Hybrid).
We then gave each image as input to the DPT semantic segmentation model, so that for each image we obtained the **prediction matrix**, contained in the variable 'prediction' of the file `run_segmentation.py` that is, a matrix where each element of the matrix is an integer that corresponds to a class associated with that pixel.
We ran the semantic segmentation model by giving as input all the images in the validation set and stored locally all the numpy arrays corresponding to the predictions obtained for each image.

Then the images were also given as input to the SAM model, so that a number of SAM-generated masks were obtained for each image.

To use the SAM model, in order to generate masks for each image, we first downloaded the 'segment_anything' module and then to generate the masks we used the `generate` method of the `SamAutomaticMaskGenerator` class.

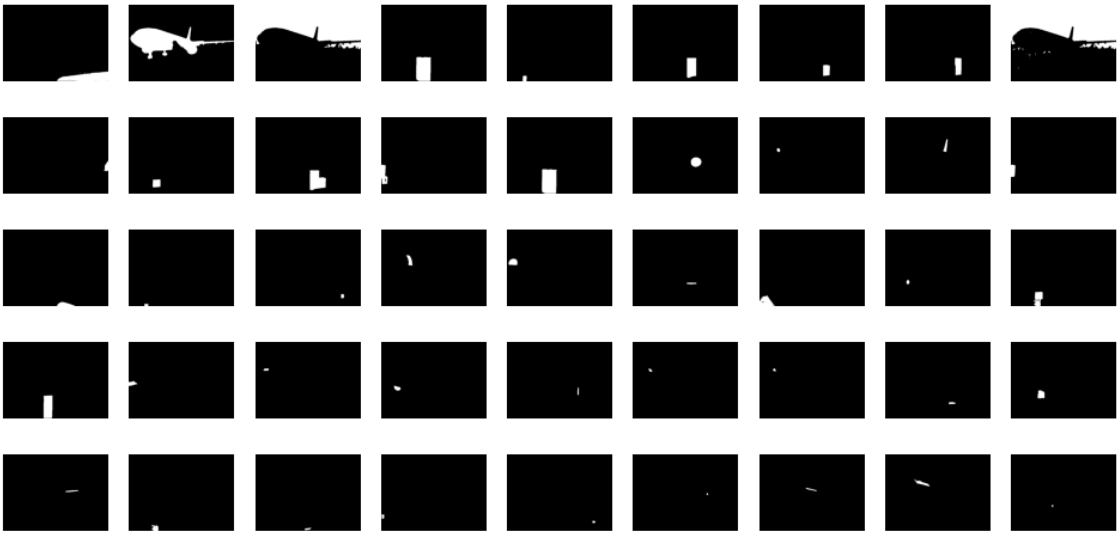


Figure 6: some masks obtained applying SAM to an image

Masks are arrays where each element is a binary value, True indicating that that pixel belongs to the highlighted mask or False otherwise.

At this point, considering a single image, for each mask we considered the corresponding pixels in the prediction matrix and calculated the majority class. In this way we associated each mask with the corresponding class.
Next, for each True pixel of each mask, we assigned the value of that mask's class to the corresponding pixel in the prediction matrix, thus eventually obtaining an adjusted version of the prediction matrix.
This procedure is described below.

Algorithm 1 Merging algorithm

```
1:
2: procedure MERGE(mask, prediction, class_value)
3:     for pixel in mask do
4:         if pixel == True then
5:             prediction[pixel] ← class_value
6:         end if
7:     end for
8:     return prediction
9: end procedure
10:
11: masks                                  ▷ List of all masks of an image obtained from SAM
12: prediction                                        ▷ Predictions matrix obtained from DPT
13: for mask in masks do
14:     class_frequencies ← empty dictionary     ▷ it will contain for each class (key) the
    number of occurrences of pixels of that class in the mask
15:     for pixel in mask do
16:         if pixel = True then
17:             value ← prediction[pixel]
18:             class_frequencies[value] ← class_frequencies[value] + 1
19:         end if
20:     end for
21:     max_value ← max(class_frequencies.values())
22:     class_value ← next(key for key, value in class_frequencies.items() if value == max_value)
23:     prediction ← Merge(mask, prediction, class_value)
24: end for
25:
```

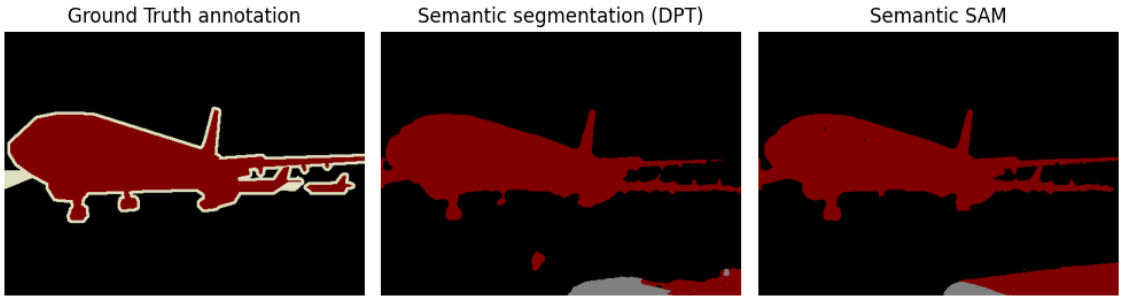
4.3 Results

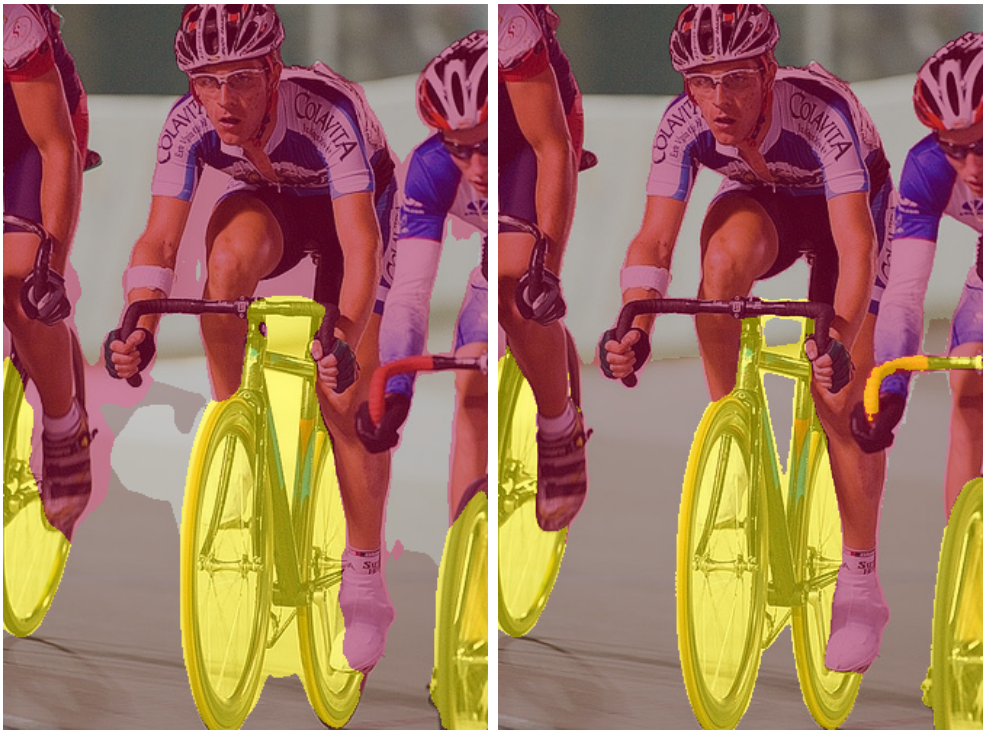
Below we can see the comparison of the results obtained from segmentation with DPT and the hybrid DPT + SAM approach, and we also compare the ground truth annotations provided in the test set and the predictions obtained with the two approaches.

It is important to note that DPT was trained with the ADE20K dataset, which is different from the dataset we used to test our results, and some classes present in PASCAL VOC 2012 are not present in ADE20K, so DPT cannot predict these classes.
Therefore, the value of pixels classified with a label not present in PASCAL VOC 2012 was set to 0.

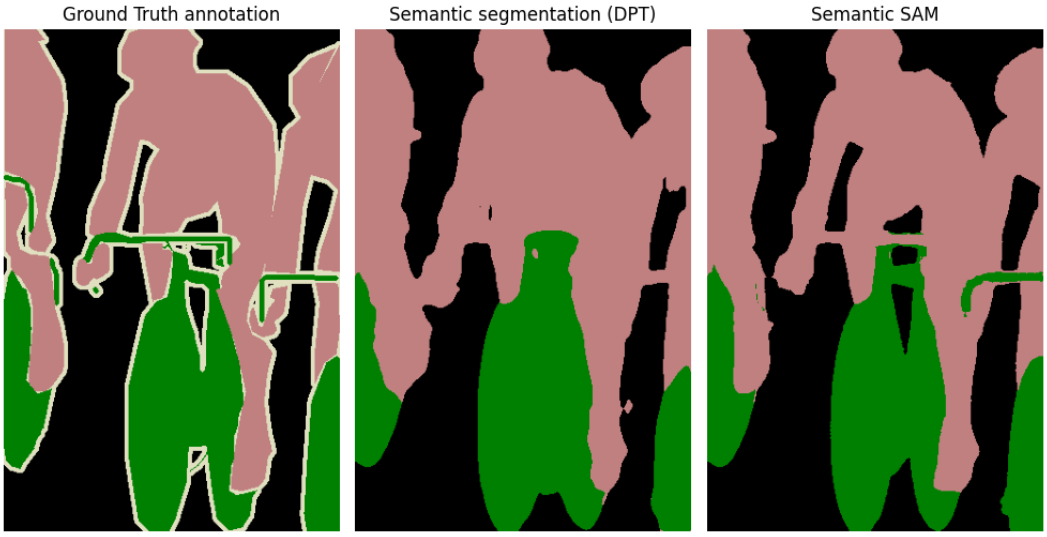
(a) Semantic Segmentation (DPT)

(b) Semantic segmentation + SAM

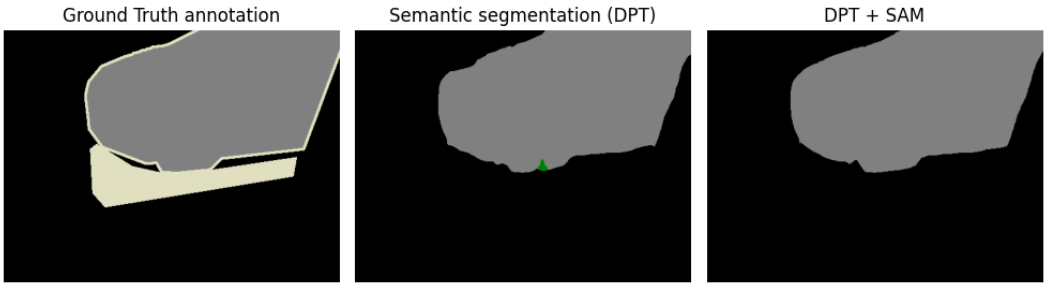




(a) Semantic Segmentation (DPT) (b) Semantic segmentation + SAM



(a) Semantic Segmentation (DPT) (b) Semantic segmentation + SAM



As can be seen from the images above, the use of the hybrid approach between DPT and SAM allowed for more accurate semantic segmentation.

4.4 Evaluation

To quantify the results obtained and understand not only from a visual point of view whether the hybrid approach actually performs better than semantic segmentation alone with DPT, we evaluated the results using the following metrics: IOU, mIOU, and DICE coefficient.

The IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. The mIoU instead is computed by taking the IoU for each class present in the image and averaging them.

The DICE coefficient is calculated by taking twice the area of the intersection, divided by the sum of the total number of pixels in both images.

For both DICE and simple IOU, we did not consider the background.

$$IoU = \frac{area\ of\ intersection}{area\ of\ union}$$

$$DICE\ Coefficient = \frac{2 \times area\ of\ intersection}{total\ num.\ pixels\ in\ both\ images}$$

Given the mismatch between the classes in the PASCAL VOC 2012 dataset and those that could be predicted by DPT, arrangements have been made in order to be able to calculate the metrics correctly. We decided not to consider in the metrics calculation all images containing those classes that could not be predicted by DPT. For this reason, metrics are calculated on a subset of 624 images.
Even the class 'background' actually would not be predictable by DPT, since it also recognizes classes such as 'sky' or 'wall,' so for the remaining images, labeled pixels that do not fall within the 20 of PASCAL VOC were associated with the background class

As we can also see from the results obtained, for all the metrics, SAM integration brought improved performance on the semantic segmentation task, with an mIoU of 0.743 versus 0.692 for semantic segmentation alone with DPT.

	mIoU	IoU	DICE Coefficient
DPT	0.692	0.646	0.734
DPT + SAM	**0.743**	**0.667**	**0.747**

Table 1: Semantic segmentation results

5 Converting bounding boxes into segmentation masks

Given that SAM can take bounding boxes as input and given that ground truth annotations consider individual objects and not all areas of the image, we tried to perform the semantic segmentation task by giving bounding boxes as input to SAM and assigning all pixels in the mask returned by SAM the bounding box label.

The pre-trained object detection model **YOLO-NAS** was used to obtain the bounding boxes. It was developed by DECI in May 2023 and trained on the MS COCO dataset, It delivers state-of-the-art performance in terms of mAP and speed, outperforming other models like YOLOv7 and YOLOv8. YOLO-NAS is 0.5 mAP point more accurate and 10-20% faster than equivalent variants of YOLOv8 and YOLOv7.
To use the model, we used the SuperGradients library, based on PyTorch.

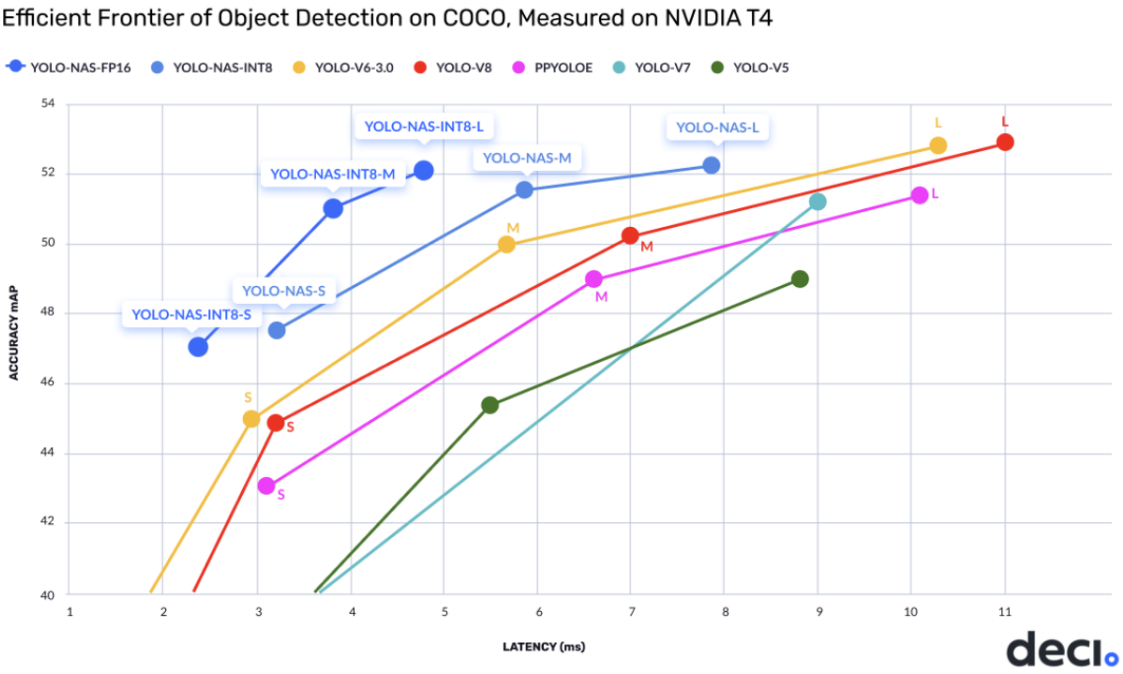


Figure 10: Picture taken from [4]

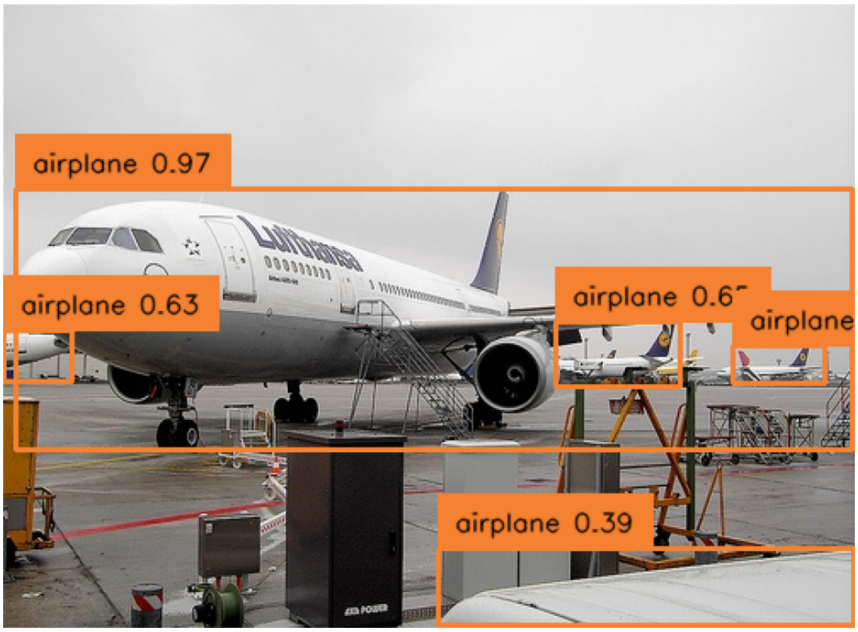


Figure 11: Bounding boxes obtained from YOLO

Figure 12: Masks obtained from SAM for each bounding box

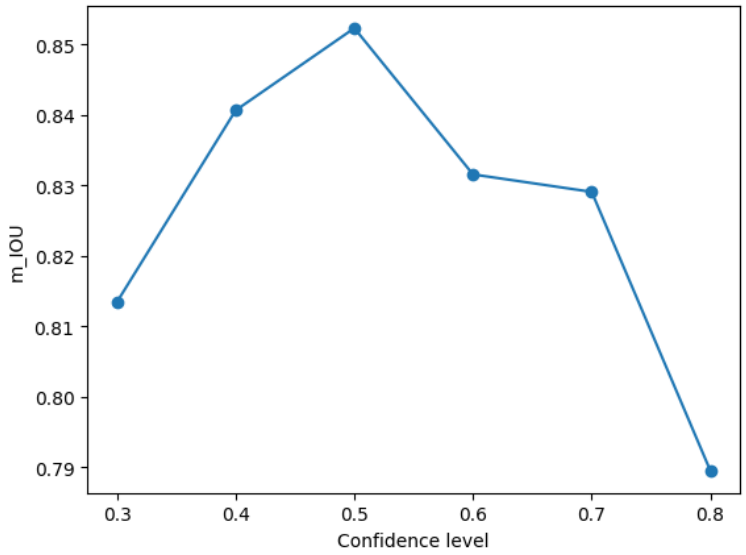
The parameter "multimask_output" was set to False, so that the model returns only one mask for each bounding box, where this mask is the one associated with the highest score. Scores gives the model's own estimation of the quality of these masks. It's an estimation of the IoU.

Algorithm 2 Merging algorithm (YOLO + SAM version)

```
prediction ← empty matrix of the same height and width of the image
masks                                  ▷ List of all masks of an image obtained from SAM
labels                  ▷ list of labels, one for each bounding box detected from YOLO
i ← 0
for mask in masks do
    for pixel in mask do
        if pixel == True then
            prediction[pixel] ← labels[i]
        end if
    end for
    i ← i + 1
end for
```

We can see that YOLO, given an input image, gives us the bounding boxes and for each

bounding box gives us the label and a confidence level.
Confidence is a number between 0 and 1, where 1 indicates maximum confidence in correctly identifying the object and 0 indicates minimum confidence. In practical terms, the confidence value indicates the estimated probability that the object detected by the model is actually what the model thinks it is.
We can set a confidence level as a threshold, such that YOLO returns only bounding boxes associated with a confidence value greater than this threshold. So the higher the threshold, the fewer bounding boxes returned for a given image.
We then calculated the m_IoU on part of the test set by trying different confidence levels in order to obtain the confidence level that gives us the best performance on our dataset. As we can see from the plot below, we chose 0.5 as the confidence level to generate the final segmentations on which we then calculated the metrics.

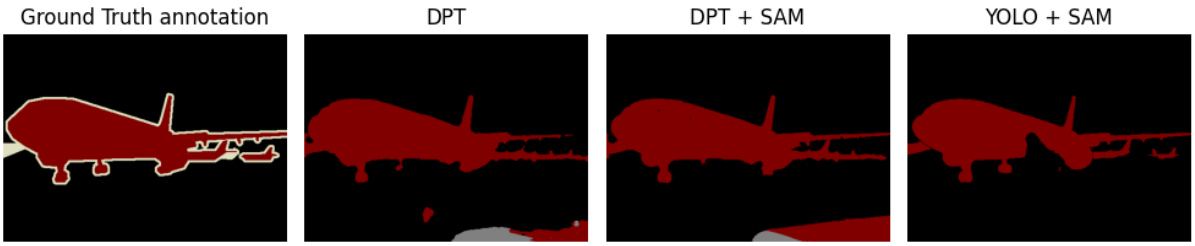


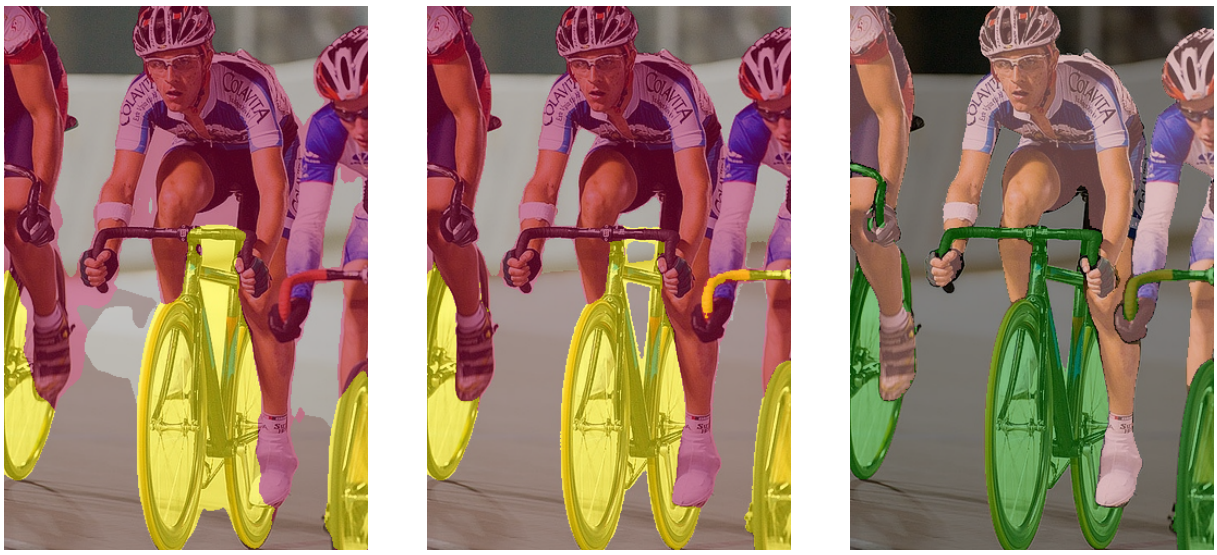
5.1 Results

(a) DPT

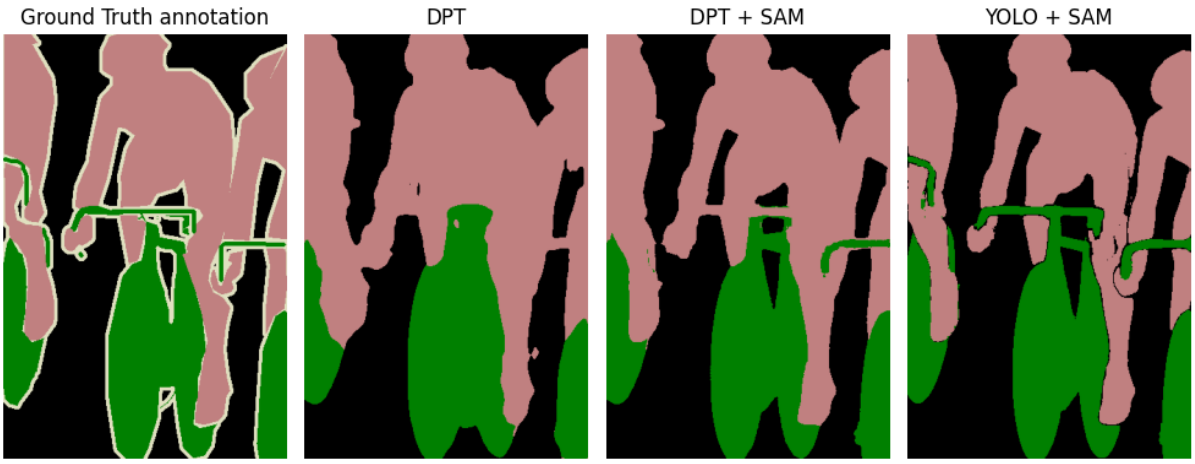
(b) DPT + SAM

(c) YOLO + SAM

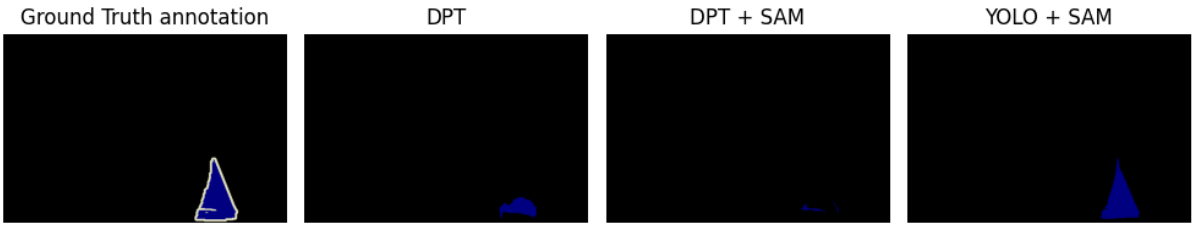




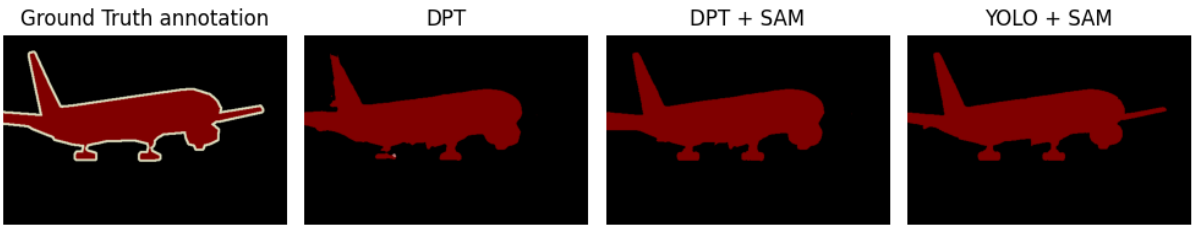
(a) DPT (b) DPT + SAM (c) YOLO + SAM



(a) DPT (b) DPT + SAM (c) YOLO + SAM



(a) DPT (b) DPT + SAM (c) YOLO + SAM



	mIoU	**IoU**	**DICE Coefficient**
DPT	0.692	0.646	0.734
DPT + SAM	0.743	0.667	0.747
YOLO + SAM	**0.839**	**0.744**	**0.81**
YOLO + SAM (full val set)	**0.851**	**0.769**	**0.834**

Table 2: Summary of semantic segmentation results on PASCAL VOC 2012 with all approaches tested

With this approach of providing bounding boxes as input to SAM, we achieved a substantial improvement, even compared to the hybrid approach using semantic segmentation with DPT.
From the images we can appreciate that the target objects in the ground truth annotations are contoured much better than the previous approaches, and this result is also confirmed by the computed metrics. In fact, the mIoU increased from the previous 0.743 to 0.839. Furthermore, since all objects belonging to all 20 PASCAL-VOC classes are detectable by YOLO-NAS, the metrics were also calculated on the entire validation set, resulting in an mIOU of 0.851 in line with the best results obtained on the PASCAL VOC 2012 dataset for the semantic segmentation task (the state of the art is represented by [5] with a mIoU of 90%)

6 Conclusions

In this work, we tried to combine a semantic segmentation model such as DPT with SAM to improve segmentation results.

Actually using the masks provided by SAM helped to achieve more accurate segmentation, thus improving the results obtained with the DPT model.

However, at least for the semantic segmentation task on the PASCAL VOC 2012 dataset, we found it better to provide as input to SAM the bounding boxes obtained from another

object detection model, and then assign the corresponding label to all pixels belonging to the mask provided by SAM for that bounding box.

References

[1] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. "Vision Transformers for Dense Prediction". In: *CoRR* abs/2103.13413 (2021). arXiv: 2103.13413. URL: https://arxiv.org/abs/2103.13413 (page 2).

[2] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV] (pages 3, 4).

[3] *DPT Repository*. URL: https://github.com/isl-org/DPT (page 5).

[4] *YOLO-NAS Repository*. URL: https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md (page 11).

[5] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. *Rethinking Pre-training and Self-training*. 2020. arXiv: 2006.06882 [cs.CV] (page 16).