Data Information

The dataset comprises RGB images, 3D bounding boxes, instance segmentation masks, and point cloud data. The primary objective of this project is to perform instance segmentation, focusing primarily on using the mask data and RGB images. While the dataset offers diverse data modalities, the instance segmentation task relies heavily on the mask data to differentiate individual objects and on the RGB images for visual context, ensuring precise object identification and separation within each image.

Models selection

Based on the available data, the YOLOv8 and SAM models were selected, with a detailed comparison of their performance provided in the following section. YOLOv8 is designed for fast segmentation, making it efficient for tasks where speed is essential. In contrast, the SAM model, which is larger and trained on a wide and diverse dataset, produces high-quality segmentations and is effective at recognizing and accurately segmenting a range of objects across different scenarios. This balance between speed and accuracy allows for an informed choice based on specific project needs.

DataLoading

Separate data loader classes were developed specifically for loading the images and masks. These classes handle the organization and loading of data directly from designated folders and have the capability to display the loaded images and masks as well. Once the data has been successfully loaded, it is divided into three sets: 80% of the data is allocated for training the model, 10% is used for testing, and the remaining 10% is set aside for validation. This structured split ensures a balanced approach for training, assessing model accuracy, and fine-tuning performance.

Yolov8 Segmentation

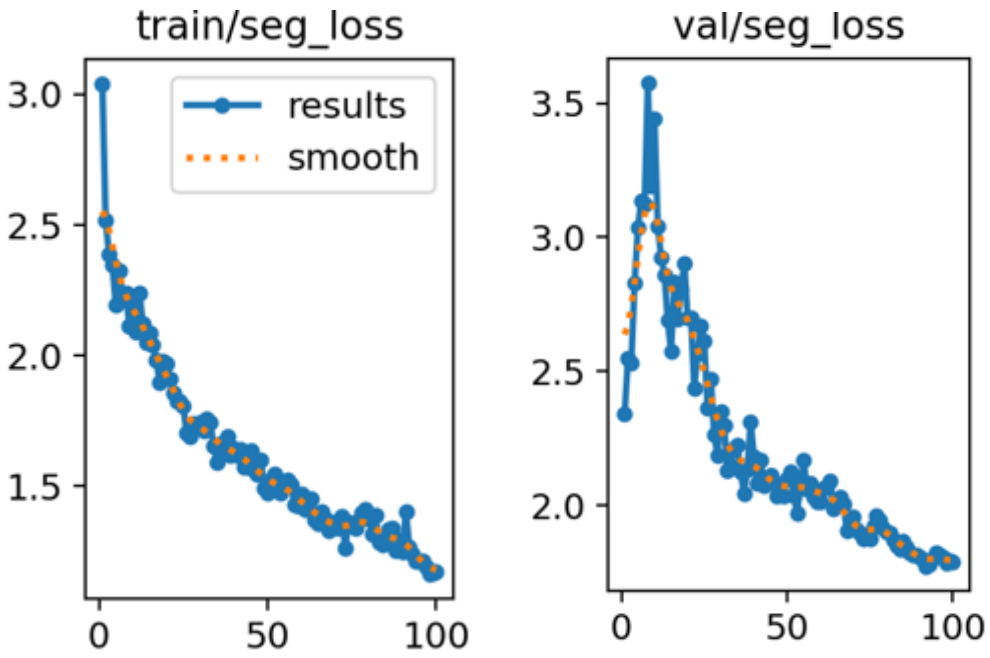
Preprocessing

The YOLOv8 segmentation model is designed to work with input images of various sizes, eliminating the need to resize them during the training process. However, for training, each image requires the contours of the objects to be stored in a text file, formatted in a normalized way. To achieve this, the boundary points of all objects are extracted from the segmentation masks, then normalized and saved in the required text file format. Furthermore, the images are arranged in a structured folder format that aligns with the model's specific training requirements, ensuring seamless compatibility with YOLOv8's processing needs.

Training

The model underwent training for a total of 100 epochs, utilizing a learning rate of 0.01 and the Adam optimizer to optimize performance. Due to constraints in Google colab computing resources, hyperparameter tuning was not performed during this training iteration. The accompanying training and validation plots illustrate that the difference between the validation loss and training loss is quite small, and both losses exhibit stability without diverging significantly. This stability suggests that the model is effectively learning from the training data without overfitting, meaning it is generalizing well.

Training and validation loss for the YOLOv8 segmentation model



Evaluation

The evaluation of the model was carried out using the Intersection over Union (IoU) metric and by counting the number of generated masks. Since class labels were not provided, a combined ground truth mask was created by merging all individual masks corresponding to each object within an image. For instance, if there were four objects in the ground truth image, each associated with its own mask, these masks were merged to form a single ground truth mask that included all objects.

Similarly, the same method was applied to the model's predictions. Individual predicted masks were combined into one overall predicted mask, and the IoU was calculated by comparing this combined mask with the ground truth mask. This method allowed us to gain insights into the model's accuracy in segmenting objects correctly.

In addition to IoU, the model's ability to differentiate between objects was assessed using the mean absolute error (MAE). For example, if there were four ground truth masks but only three predicted masks, the MAE was computed between these two sets to evaluate the extent of discrepancy in object separation. The model achieved an IoU of 0.714 on a test set that constituted 10% of the data held out for evaluation. Furthermore, the mean absolute error was recorded at 3.88, indicating the degree to which the predicted masks deviated from the actual masks. This comprehensive evaluation process highlights both the model's segmentation capabilities and its proficiency in distinguishing between individual objects.

Segment Anything Model

For instance segmentation with the Segment Anything Model (SAM), a two-stage approach is utilized. The first stage involves using an object detection model to identify the objects present in an image. Once the objects have been detected, the SAM is applied to perform the segmentation of these identified objects.

In this particular setup, the YOLOv8 model is used for the initial object detection phase. YOLOv8 excels at quickly and accurately locating objects within an image, making it an ideal choice for this task. After the objects have been successfully detected, the SAM base model is employed to segment the detected objects into distinct regions. This combination of YOLOv8 for detection and SAM for segmentation allows for effective and precise instance segmentation, facilitating a clearer understanding of the objects within the image. This method leverages the strengths of both models to achieve enhanced performance in segmenting instances accurately.

Preprocessing

When training the YOLOv8 detector model, it is important to note that input images can come in various sizes. However, each image must be accompanied by bounding boxes that accurately represent the objects within it. These bounding boxes are generated from the existing masks and subsequently saved in text files, which are essential for the training process.

In a similar manner, the Segment Anything Model (SAM) also requires bounding boxes to determine which objects need to be segmented. For training the SAM model, it is necessary to have a reference image alongside its corresponding mask. To prepare for this training, a combined mask is created by merging all individual object masks present in the image. This combined mask provides a comprehensive representation of all objects within that image.

Once the combined mask is generated, both the mask and the reference image are resized to a uniform dimension of 256x256 pixels. By following these procedures, the training for both the YOLOv8 detector model and the SAM model can be conducted successfully, ensuring that the models can accurately identify and segment objects within the images.

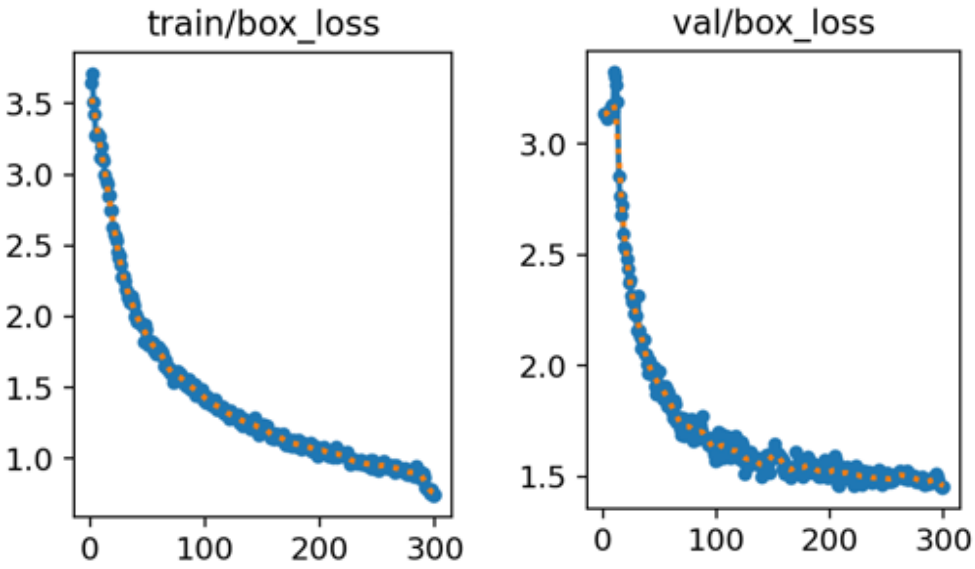
Training

The training process for the YOLOv8 model was for 300 epochs, utilizing a learning rate of 0.01 and the Adam optimizer for efficient training. After completing the training of the YOLOv8 model, the SAM model was subsequently trained for 5 epochs, using a smaller learning rate of 0.0001, while also employing the Adam optimizer.

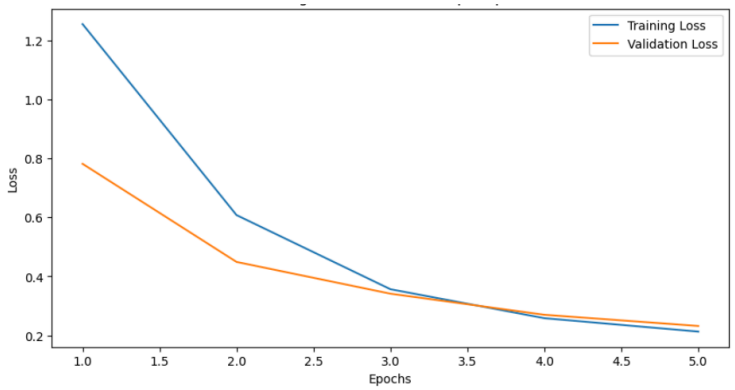
It is important to note that hyperparameter tuning was not performed during this iteration, nor were the number of training epochs increased, primarily due to limitations in Google colab computing power. These constraints necessitated a focus on a more straightforward training approach without extensive optimization efforts.

The training and validation results are illustrated in the plots provided below. Upon examining these plots, it is evident that the difference between the training loss and validation loss is minimal. Additionally, both losses remain stable throughout the training process without showing signs of divergence. This stability suggests that the model is effectively learning without overfitting to the training data. Overall, these observations indicate a successful training regime, allowing both models to learn from the data while maintaining their ability to generalize to unseen examples.

Training and validation loss for the YOLOv8 detection model



Training and validation loss for the SAM model



Evaluation

The evaluation method utilized for the SAM model mirrored the approach previously outlined. Ultimately, the SAM model achieved an Intersection over Union (IoU) score of 0.804, indicating its effectiveness in accurately segmenting objects. Additionally, the model recorded a mean absolute error of 1.2, reflecting its precision in differentiating between objects within the images. These results highlight the model's strong performance in instance segmentation tasks.

Inference Optimization

The YOLO models were successfully converted into TensorRT format, which optimizes them for high-performance inference on compatible hardware. This conversion process enhances the model's efficiency and speed during deployment. The following code snippet illustrates the specific steps taken to achieve this conversion, ensuring that the models can leverage TensorRT's capabilities for improved processing time and resource utilization.

```
from ultralytics import YOLO

# Load the YOLOv8 model

model = YOLO("yolov8n.pt")


# Export the model to TensorRT format

model.export(format="engine")# creates 'yolov8n.engine'


# Load the exported TensorRT model

tensorrt_model = YOLO("yolov8n.engine")
```

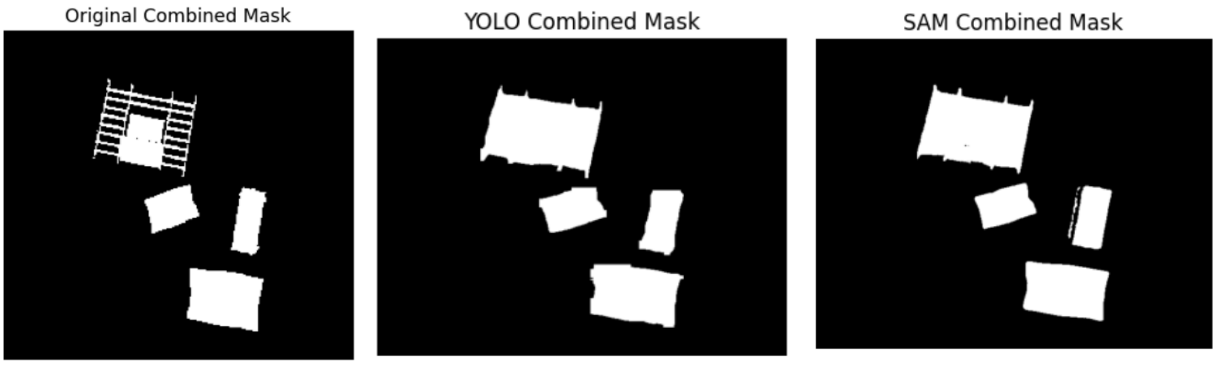
Comparison

Instance Segmentation on a test image

The images presented below demonstrate that both the SAM and YOLO models effectively recognize each individual object within the scenes. Each model assigns unique masks to the detected objects, allowing for clear differentiation and separation between them. This capability highlights the strength of both models in accurately performing instance segmentation tasks, as they can distinctly outline and categorize various objects.

Combined mask

The masks illustrate that both the YOLO and SAM models faced challenges in achieving fine-grained segmentation. Nevertheless, the SAM model demonstrated a superior performance by producing sharper segmentations characterized by more clearly defined boundaries. This distinction in segmentation quality indicates that while both models can identify and separate objects, the SAM model is better equipped for precise contours of each object. This capability is particularly important in applications requiring high accuracy in object recognition and classification, where the clarity of boundaries can significantly impact the overall effectiveness of the segmentation results.



Performance

The SAM model outperforms the YOLO segmentation model, as demonstrated in the comparison table below. The SAM model achieves a higher Intersection over Union (IoU) score, highlighting its superior segmentation accuracy and capability to distinguish object boundaries

effectively. Additionally, the SAM model shows a lower mean absolute error, indicating it can more precisely differentiate between objects in an image than the YOLO segmentation model. These metrics suggest that SAM provides more refined segmentation, offering sharper boundaries and better separation across various objects within each scene. However, when it comes to speed, the YOLO model has an advantage due to its smaller parameter count, allowing it to perform faster inferences. This trade-off makes YOLO suitable for applications requiring speed, while SAM is preferable for high-precision segmentation.

	IoU	Mean absolute error
Yolov8 Segmentation	0.714	3.88
SAM	0.804	1.2

Code Structure

The `DataPreprocess.py` file is responsible for handling the processes of reading and displaying data within the project. It ensures that the data is correctly accessed and can be visualized for further analysis. The `helper.py` file contains a collection of essential helper functions that assist in various operations throughout the code, streamlining tasks that would otherwise require repetitive coding.

The `Yolo_process.ipynb` notebook is specifically designed for preprocessing data in preparation for the YOLO model, ensuring that the input meets the model's requirements. In contrast, the `train_sam.ipynb` notebook focuses on training the SAM model, providing the necessary steps and configurations for effective learning.

For training the YOLOv8 segmentation model, the `train_yolo_seg.py` script is utilized, while the `train_yolo.ipynb` notebook is dedicated to training the YOLOv8 detector model. Together, these scripts and notebooks facilitate the training of both segmentation and detection models.

Additionally, the `Infer.ipynb` notebook serves to visualize the performance of the trained models, allowing to assess their effectiveness in segmentation and object detection tasks. Finally, the `test.ipynb` notebook is employed for evaluating the models' performance, providing metrics and insights that help understand their capabilities and limitations.