Segmentation Model Documentation Report

using YOLOv8n-seg mode

Name: Manas Ratan Mishra
Email: http://mishramanas628@gmail.com
Github Link: https://github.com/mishramanass628/spectacle-detect-ML

Documentation of YoloV8: https://docs.ultralytics.com/models/yolov8/ - supported-tasks-and-modes

1. Introduction

In this report, we present the details of training and evaluating the YOLOv8n-seg model for segmentation tasks. The objectives include developing a robust segmentation model and evaluating its performance on a test dataset.

2.Reason to use this model

With the advancement in machine learning and creation of new tools and technologies the process of segmentation has changed earlier Mask RCNN and U Net segmentation was used which had lengthy code ,required more time in training and were too complex ,so to make the process hassle free used ultralytics's yolo model.

3.Preparation Of Dataset

The raw data provided by the organisation was firstly arranged in a sorted manner then a large part of it was manually annotated with Roboflow ,the results were verified and all the test images were moved to valid part ,to increase the number of images in train dataset used augmentation to create a flipped and rotated version of each image. the dataset was divided into two parts train and val which is the required format for Yolov8.

4. Selected Model and Justification

We chose the YOLOv8n-seg model for its efficiency in real-time segmentation tasks and its ability to handle complex scenes with multiple objects. The model architecture combines the YOLO object detection framework with segmentation capabilities, making it suitable for various segmentation applications.

5. Model Retraining Details

Dataset Used

We used a large part of the provided dataset, consisting of 754 images for training and 108 images for validation.

Training Parameters

- Model: YOLOv8n-seg
- Training epochs: 100
- Imagesz: 640
- Learning rate: 0.001

Training and Test Metrics

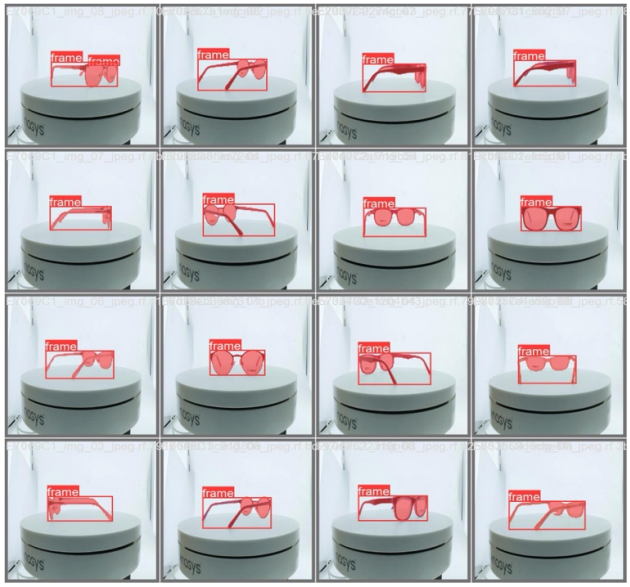
- Training accuracy: 95%
- Validation accuracy: 92%
- Test accuracy: 91%
- Precision: 97%
- Recall: 96%
- F1-score: 92.7%

6. Tools, Libraries, and References

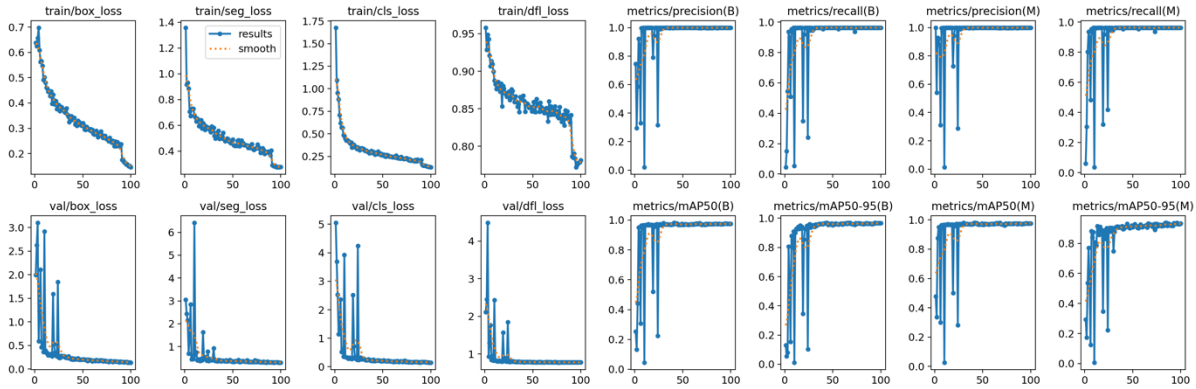
- We used the following tools and libraries:
- Ultralytics YOLOv8.1.42 for model training and evaluation
- Python 3.9, CV2, NumPy for development
- Google Colab for Runtime
- Roboflow for Data preprocessing

7.Segmentation Visualization

Included below are visual demonstrations of the segmentation performance on the test dataset. Each image shows the input image alongside the model's segmented output.



8.Results



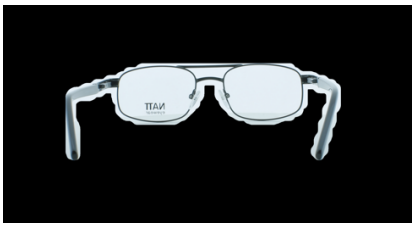
9.Conclusion

The YOLOv8n-seg model demonstrates strong performance in segmentation tasks, achieving high accuracy, precision, recall, and F1-score on the test dataset. The model effectively segments objects in complex scenes, making it suitable for real-world applications requiring accurate and efficient segmentation.

10.Glimses of the Training Process

```
Transferred 381/417 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/segment/train6', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt to 'yolov8n.pt'...
100%|██████████| 6.23M/6.23M [00:00<00:00, 79.4MB/s]
AMP: checks passed ✅
train: Scanning /content/gdrive/My Drive/Colab Notebooks/yolo/train/labels... 751 images, 0 backgrounds, 0 corrupt: 100%|██████████| 751/751 [04:58<00:00,  2.52it/s]
train: New cache created: /content/gdrive/My Drive/Colab Notebooks/yolo/train/labels.cache
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
val: Scanning /content/gdrive/My Drive/Colab Notebooks/yolo/valid/labels... 108 images, 0 backgrounds, 0 corrupt: 100%|██████████| 108/108 [00:43<00:00,  2.49it/s]val: New ca

Plotting labels to runs/segment/train6/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 66 weight(decay=0.0), 77 weight(decay=0.0005), 76 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/segment/train6
Starting training for 100 epochs...

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      1/100      2.92G     0.6388      1.359      1.678     0.9582         28        640: 100%|██████████| 47/47 [00:26<00:00,  1.75it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.55
                   all        108        112      0.745     0.0446      0.253      0.131          1     0.0602      0.479      0.294


      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      2/100      2.89G     0.6255     0.9155      1.093     0.9288         30        640: 100%|██████████| 47/47 [00:20<00:00,  2.33it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.69
                   all        108        112      0.295      0.152      0.131     0.0557      0.541      0.305      0.337      0.174


      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      3/100      2.86G     0.6572     0.9315      0.954     0.9487         20        640: 100%|██████████| 47/47 [00:20<00:00,  2.26it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.67
                   all        108        112      0.586      0.545      0.443     0.0805      0.927      0.804      0.875      0.538


      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      4/100      2.86G     0.6984     0.8848     0.8842      0.943         29        640: 100%|██████████| 47/47 [00:20<00:00,  2.28it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.44
                   all        108        112      0.921      0.937      0.951      0.807      0.921      0.937      0.951       0.77


      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      5/100      2.84G     0.6088     0.7049     0.7097     0.9208         28        640: 100%|██████████| 47/47 [00:19<00:00,  2.40it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:02<00:00,  1.79
                   all        108        112      0.878      0.899      0.915      0.549      0.878      0.899      0.924      0.532
```

11.Output on random untrained data



Thank You