Ian Morgan-Graham
Dr. Hari Kalva
COT 5930 – Digital Image Processing SPRING 2024
April 14th, 2024
Project: Image/Video Annotation and Segmentation with YOLOv8

I. INTRODUCTION

YOLO, short for "You Only Look Once," is a state-of-the-art, real-time object detection system. It's known for its speed and accuracy in detecting objects in images and videos. YOLO approaches object detection as a single regression problem, directly predicting bounding boxes and class probabilities for those boxes. This allows YOLO to detect objects in real-time with a single pass through the neural network, making it significantly faster than traditional object detection approaches. The purpose of this project was to gain some experience with image and video annotation and segmentation using the YOLO object detection algorithm. The YOLO algorithm was originally written in C or C++. This project utilized the Python derived version of YOLO.

The main steps in this project were:

-- obtain 10 suitable aerial drone clips of coastline footage.
-- sample 5 second clips from each of the original clips.
-- extract a random sample frame/image from each of the 5 second coastline clips.
-- annotate objects in each of the randomly extracted sample frames/images.
-- use the annotated sample images to train YOLO for annotation and segmentation.
-- use trained YOLO model to annotate objects in each 5 second clip.
-- use trained YOLO model to segment objects in each 5 second clip.
-- use trained YOLO model to segment unseen video clips.

II. DATA ACQUISITION AND PRE-PROCESSING

Open source aerial drone footage of various coastlines was obtained from two websites -- https://pixabay.com and https://www.pexels.com. An effort was made to obtain a broad range of coastlines – beach-less and very rocky coastlines, coastlines with buildings or other human structures, coastlines with wide flat beaches, and coastlines with different levels of shoreline foliage. The 10 coastline video clips were downloaded. These original video clips varied in length from 12 to 36 seconds. Since processing these long video clips would consume considerable time and processing resources, the decision was made to use only 5 seconds from each long video clip. The open source video editing program Kdenlive (version 21.12.3) (Figure 1 below) was used to isolate 5 second sections from each long clip.

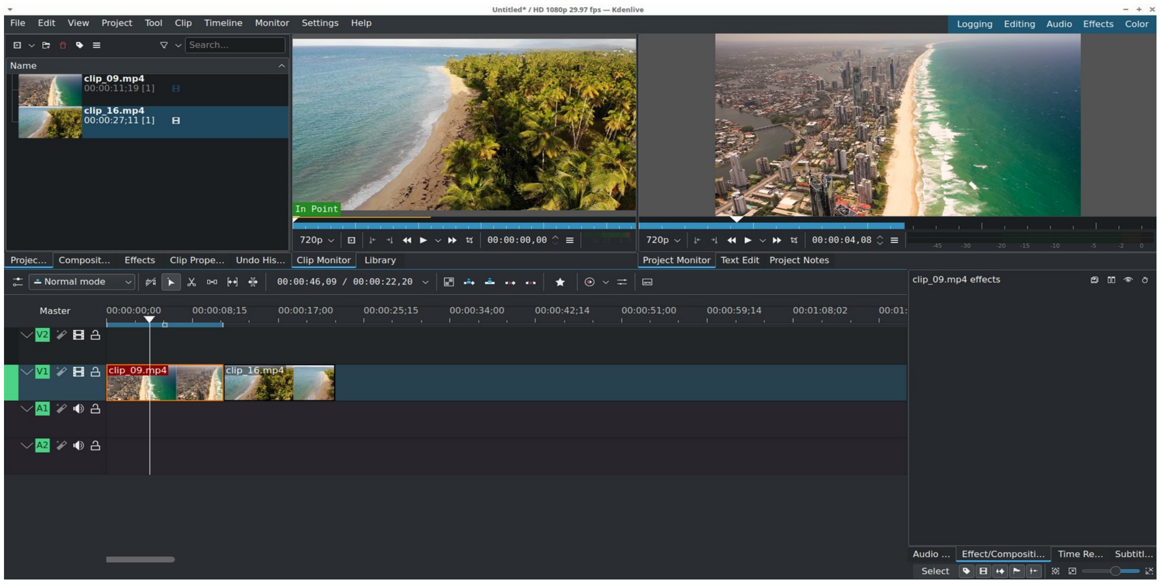

Untitled* / HD 1080p 29.97 fps — Kdenlive
File Edit View Project Tool Clip Timeline Monitor Settings Help
Logging Editing Audio Effects Color
clip_09.mp4 00:00:11;19 [1]
clip_16.mp4 00:00:27;11 [1]
In Point
00:00:00,00
00:00:04,08
Project Monitor Text Edit Project Notes
Clip Monitor Library
Normal mode
00:00:46,09 / 00:00:22,20
clip_09.mp4 effects
Master
V2 V1 A1 A2
clip_09.mp4 clip_16.mp4

Figure 1. Using Kdenlive video editor to collect 3 second clips

The charcteristics of these ten, 5 second duration video clips was extracted using a function, “extract_video_info()”, in the attached file, “YOLO Annotation Segmentation.ipynb” . The clips are compared below in table 1.

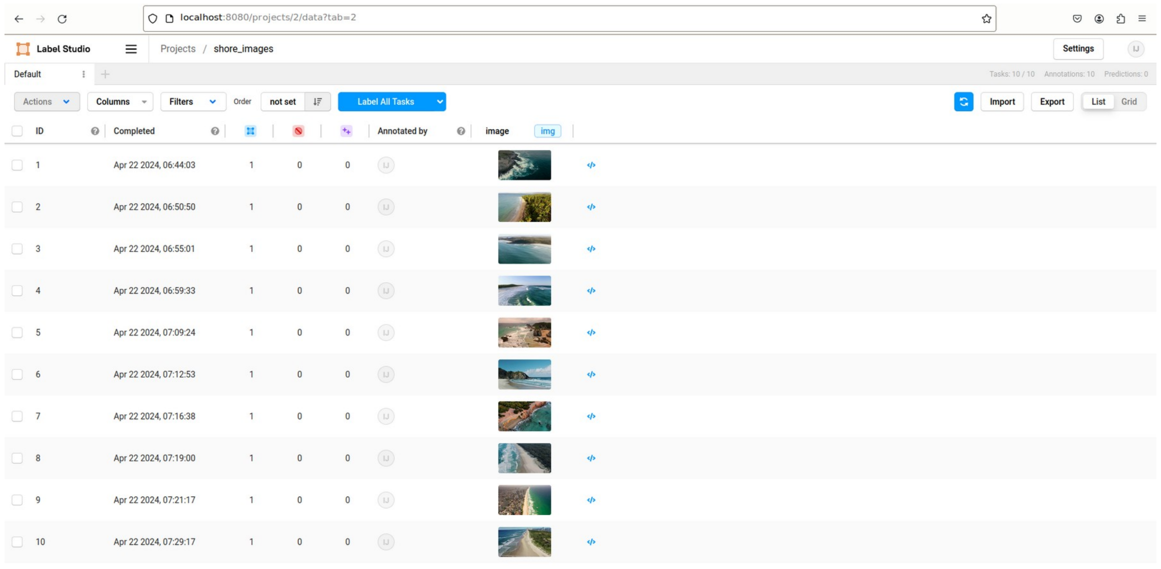
clip name	duration (secs)	fps	total frames	frame width	frame height
clip_01.mp4	5.07	30	152	1920	1080
clip_02.mp4	5.07	30	152	1920	1080
clip_03.mp4	5.09	24	122	1920	1080
clip_04.mp4	5.09	24	122	1920	1080
clip_05.mp4	5.07	30	152	1920	1080
clip_06.mp4	5.07	30	152	1920	1080
clip_07.mp4	5.07	30	152	1920	1080
clip_08.mp4	5.07	30	152	1920	1080
clip_09.mp4	5.07	30	152	1920	1080
clip_10.mp4	5.07	30	152	1920	1080

Table 1. Characteristics of 5 second duration clips.

Overall, the 5 second clips are fairly similar with some variation in frames per second (24 to 30) and total frame (122 to 152). These 5 second video clips are located in root directory folder, “videos_folder”.

III. METHOD AND APPROACH

Once the ten, 5 second clips were constructed, the next step was extracting a frame from each clip to use for image annotation and subsequent model training. A function, “random_frame_to_image()”, in the attached .ipynb file was used to extract the frames. The extracted frames are located in the root directory folder, “images_folder”. Annotation of these extracted frame/images was done using the online, open source tool, Label Studio (https://labelstud.io/). Figures 2, 3a, and 3b show various stages of image annotation using Label Studio.



ID	Completed				Annotated by	image
1	Apr 22 2024, 06:44:03	1	0	0	IJ	
2	Apr 22 2024, 06:50:50	1	0	0	IJ	
3	Apr 22 2024, 06:55:01	1	0	0	IJ	
4	Apr 22 2024, 06:59:33	1	0	0	IJ	
5	Apr 22 2024, 07:09:24	1	0	0	IJ	
6	Apr 22 2024, 07:12:53	1	0	0	IJ	
7	Apr 22 2024, 07:16:38	1	0	0	IJ	
8	Apr 22 2024, 07:19:00	1	0	0	IJ	
9	Apr 22 2024, 07:21:17	1	0	0	IJ	
10	Apr 22 2024, 07:29:17	1	0	0	IJ	

Figure 2. Extracted frame/images uploaded into Label Studio for annotation

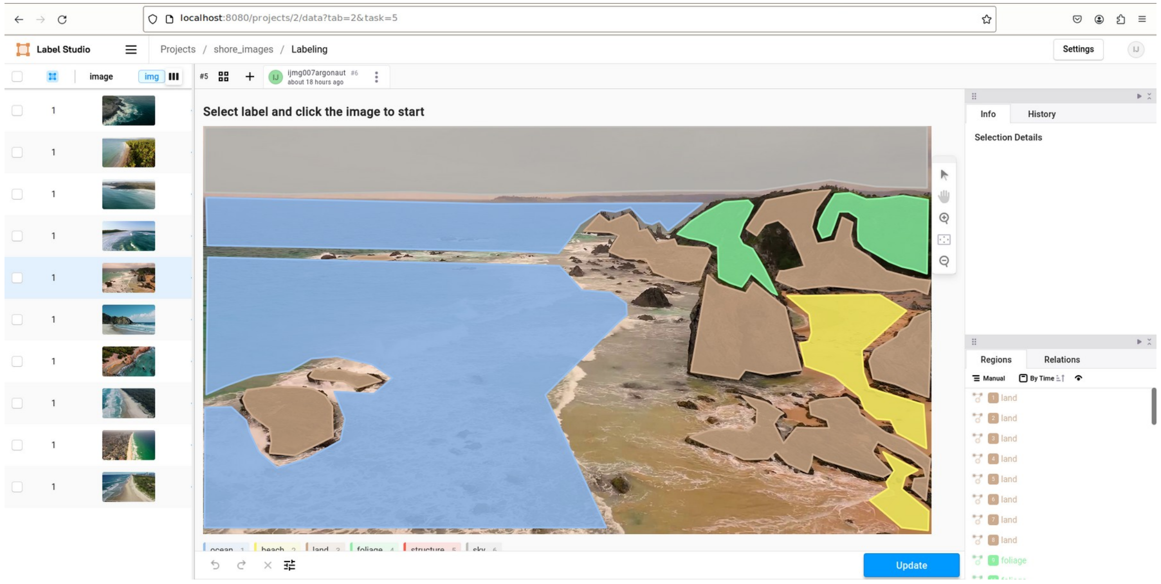


Figure 3a. Polygon annotation of a coastline with ocean (blue), land (brown), sky (gray), foliage (green), and beach (yellow).

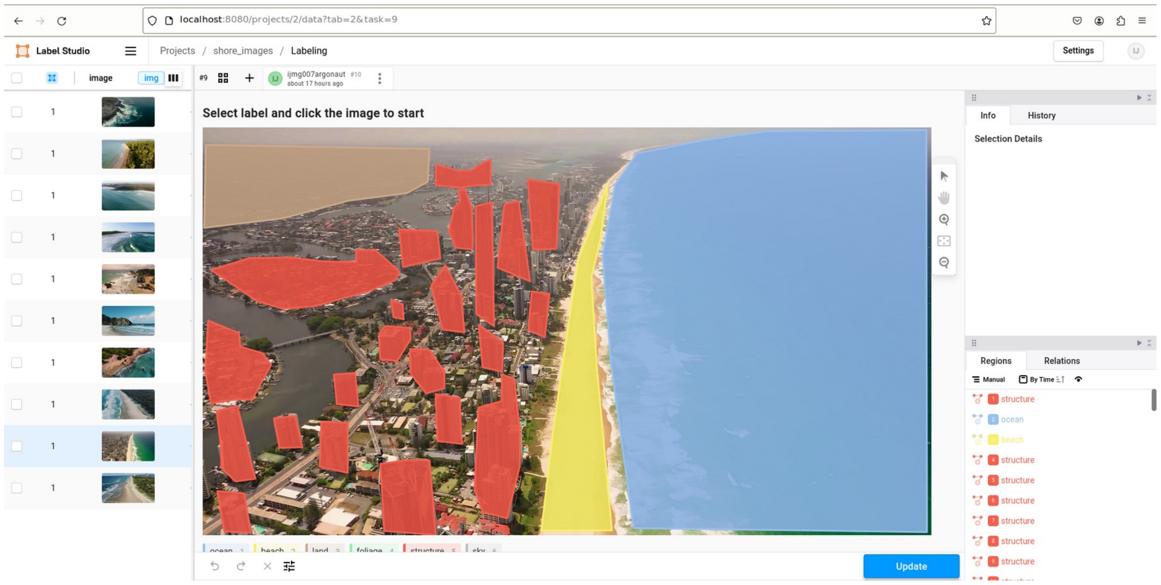

localhost:8080/projects/2/data?tab=2&task=9
Label Studio
Projects / shore_images / Labeling
Settings
Select label and click the image to start
Info
History
Selection Details
Regions
Relations
structure
ocean
beach
Update

Figure 3b. Polygon annotation of a coastline with ocean (blue), land (brown), sky (gray), structures (red), and beach (yellow).

After annotation with Label Studio, the annotated files were exported as a .zip file in YOLO compatilble .txt file formats. The exported files included images renamed post-annotation by LabelStudio, the correspondingly named .txt label files describing the location and class/category of each annotated object on an image, and a classes.txt file listing the classes/categories to which annotated objects can be assigned. Figures 4 through 6 along with Table 2 summarize these files and the their relationships.

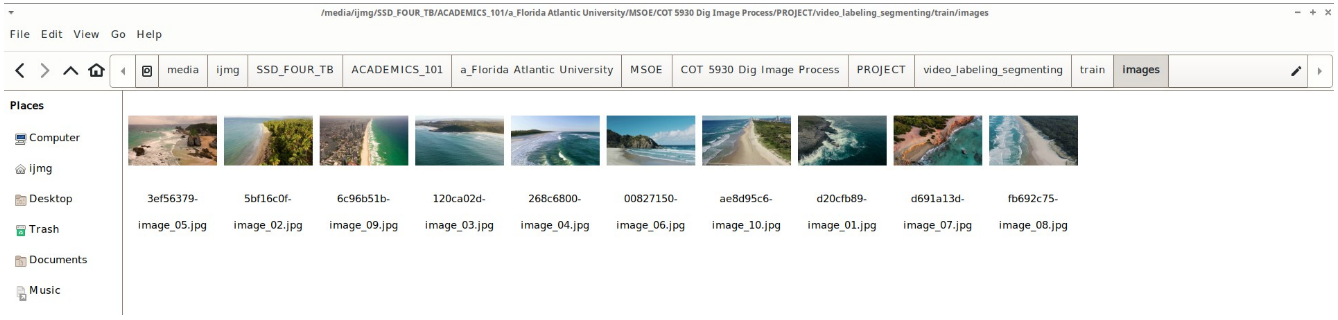

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/train/images
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting train images
Places
Computer
ijmg
Desktop
Trash
Documents
Music
3ef56379-image_05.jpg
5bf16c0f-image_02.jpg
6c96b51b-image_09.jpg
120ca02d-image_03.jpg
268c6800-image_04.jpg
00827150-image_06.jpg
ae8d95c6-image_10.jpg
d20cfb89-image_01.jpg
d691a13d-image_07.jpg
fb692c75-image_08.jpg

Figure 4. Images renamed post-annotation by Label Studio (in training "train" folder).

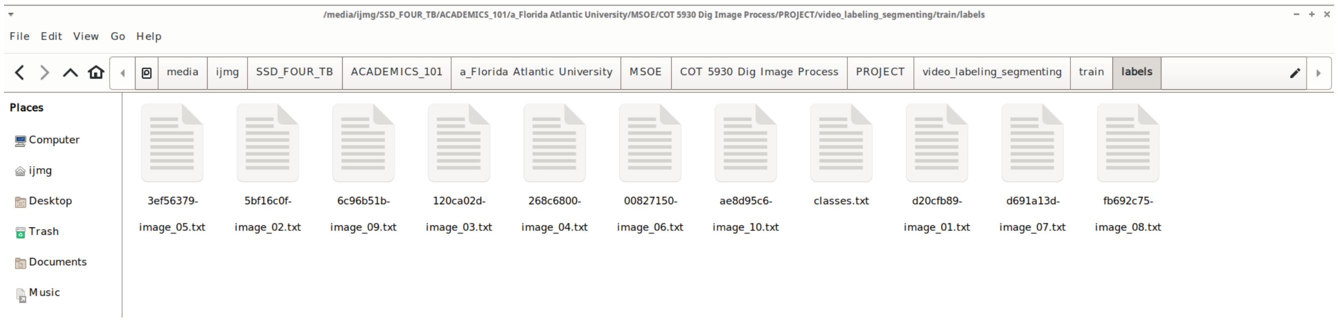
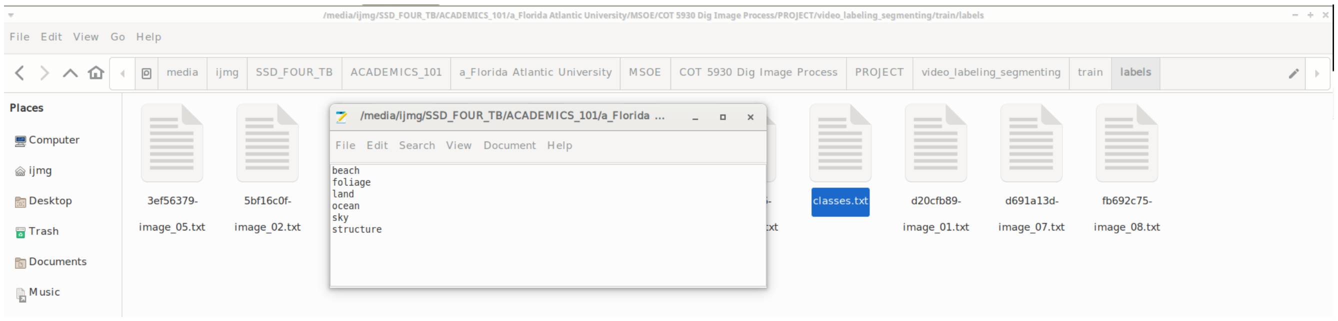

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/train/labels
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting train labels
Places
Computer
ijmg
Desktop
Trash
Documents
Music
3ef56379-image_05.txt
5bf16c0f-image_02.txt
6c96b51b-image_09.txt
120ca02d-image_03.txt
268c6800-image_04.txt
00827150-image_06.txt
ae8d95c6-image_10.txt
classes.txt
d20cfb89-image_01.txt
d691a13d-image_07.txt
fb692c75-image_08.txt

Figure 5. Correspondingly named .txt label files describing the location and class/category of each annotated object on an image. Also note the classes.txt file fourth from right (in training "train" folder).


/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/train/labels

File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting train labels
Places
Computer
ijmg
Desktop
Trash
Documents
Music
/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida ...
File Edit Search View Document Help

```
beach
foliage
land
ocean
sky
structure
```

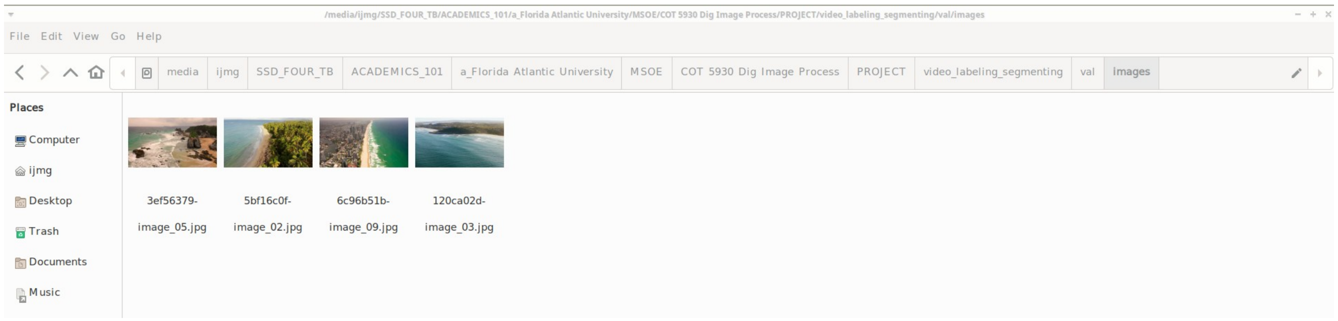
3ef56379-image_05.txt
5bf16c0f-image_02.txt
classes.txt
d20cfb89-image_01.txt
d691a13d-image_07.txt
fb692c75-image_08.txt

Figure 6. Contents of classes.txt file listing the classes/categories to which annotated objects can be assigned (in training "train" folder).

Renamed Image Containing Annotated Objects	Label File Linking Object Annotation and Category in Renamed Image File
d20cfb89-image_01.jpg	d20cfb89-image_01.txt
5bf16c0f-image_02.jpg	5bf16c0f-image_02.txt
120ca02d-image_03.jpg	120ca02d-image_03.txt
268c6800-image_04.jpg	268c6800-image_04.txt
3ef56379-image_05.jpg	3ef56379-image_05.txt
00827150-image_06.jpg	00827150-image_06.txt
d691a13d-image_07.jpg	d691a13d-image_07.txt
fb692c75-image_08.jpg	fb692c75-image_08.txt
6c96b51b-image_09.jpg	6c96b51b-image_09.txt
ae8d95c6-image_10.jpg	ae8d95c6-image_10.txt

Table 2. Annotation file relationships (applies to both training and validation folders)

The files shown in Figures 4 through 6 along with Table 2 show the arrangement of the annotated files in the root folder "train" containing the subfolders, "images" (Figure 4) and "labels" (Figure 5). Similar to the training folder, "train", the root directory also contains another folder for validation, "val", that contains its own subfolders, "images" and "labels". However, these validation subfolders only contain four of the renamed training images and four of the corresponding labels .txt files as well as a copy of the classes.txt file. This is shown in Figures 7 and 8.


/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/val/images

File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting val images
Places
Computer
ijmg
Desktop
Trash
Documents
Music

3ef56379-image_05.jpg
5bf16c0f-image_02.jpg
6c96b51b-image_09.jpg
120ca02d-image_03.jpg

Figure 7. Images renamed post-annotation by Label Studio (in validation "val" folder).

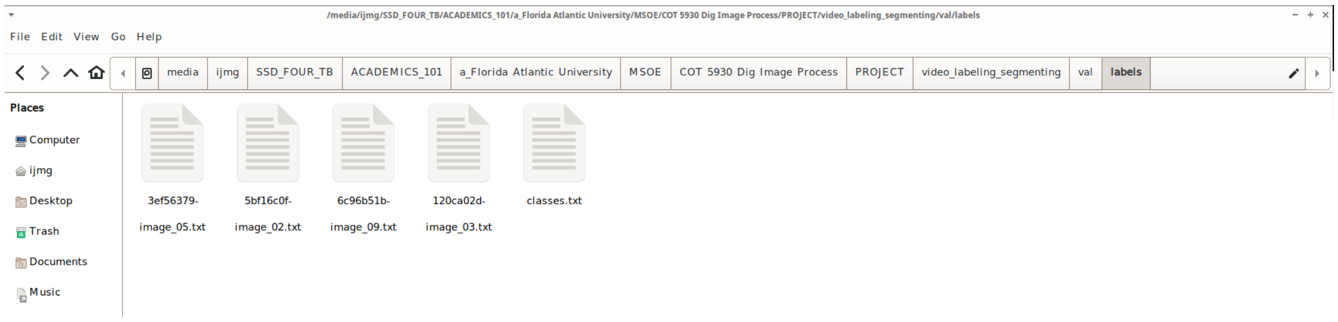

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/val/labels
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting val labels
Places
Computer
ijmg
Desktop
Trash
Documents
Music
3ef56379-image_05.txt
5bf16c0f-image_02.txt
6c96b51b-image_09.txt
120ca02d-image_03.txt
classes.txt

Figure 8. Correspondingly named .txt label files describing the location and class/category of each annotated object on an image. Also not the classes.txt file fourth from right (in validation "val" folder).

This organization of training and validation data is required by the YOLO framework for both label training and segmentation training.

At this stage (Figure 9), the ten 5 second images are located in the root folder, "videos_folder". The ten extracted frames/images are located in the root folder, "images_folder". The training and validation data are in their appropriate folders in the root directory, "train" and "val", respectively. The root directory also contains yet another copy of the classes.txt file. Also present is the ipynb file containing the Python code orchestrating the project and a .docx/.pdf file pairing representing the work-in-progress of this report. One final file required by the YOLO framework, "data_custom.yaml", contains the complete directory paths to the training and validation folders, the number of annotation classes/categories ("nc"), and an exact ordered list ("names") of the annotation classes/categories (Figure 10). Note that this ordered list must contain the annotation classes/categories in the same order as shown in the classes.txt files mentioned previously.

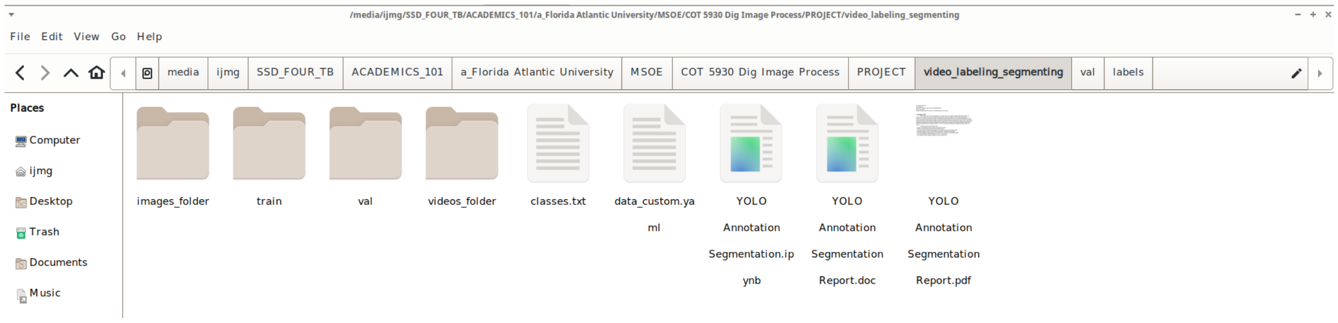
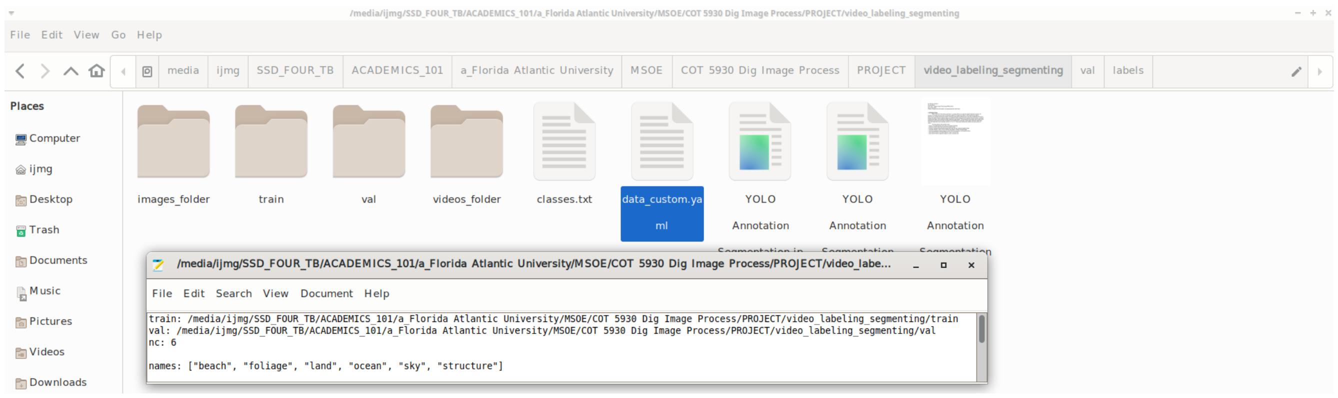

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting val labels
Places
Computer
ijmg
Desktop
Trash
Documents
Music
images_folder
train
val
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation Segmentation.ipynb
YOLO Annotation Segmentation Report.doc
YOLO Annotation Segmentation Report.pdf

Figure 9. Root directory, "video_labeling_segmenting" with content files and folders.

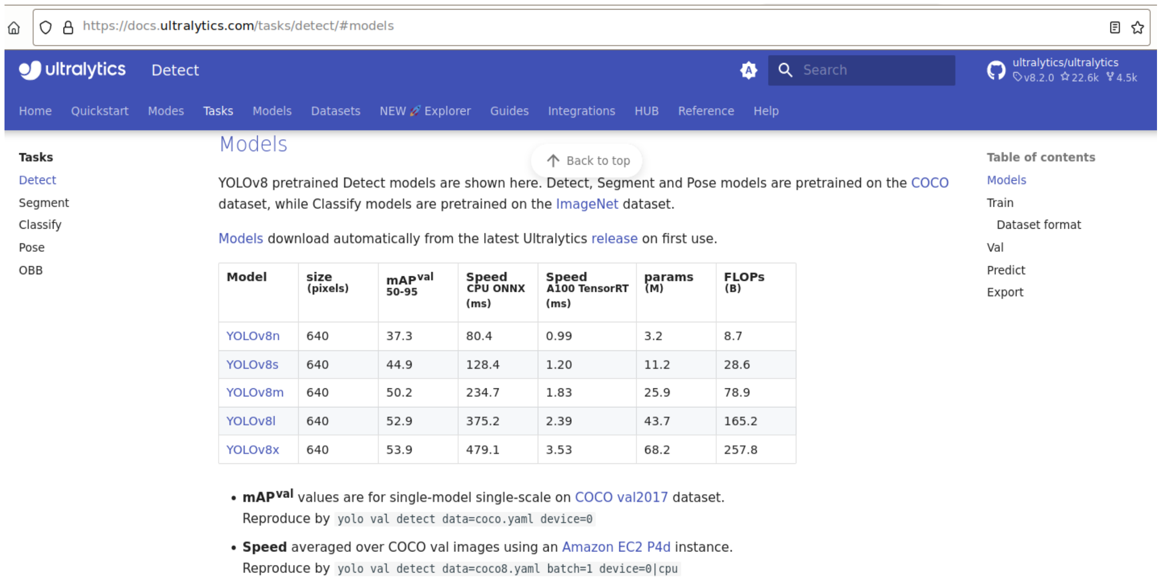

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting val labels
Places
Computer
ijmg
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
images_folder
train
val
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation
YOLO Annotation
YOLO Annotation
/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labe...
File Edit Search View Document Help

```
train: /media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/train
val: /media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/val
nc: 6

names: ["beach", "foliage", "land", "ocean", "sky", "structure"]
```

Figure 10. Root directory, "video_labeling_segmenting" with contents of "data_custom.yaml" file shown.

The next step is to download and install the Ultralytics library along with its dependencies from the Python Package Index (PyPI). Ultralytics is a deep learning research group and open-source project that develops state-of-the-art natural language processing and computer vision algorithms like YOLO. This was done with the code, "!pip install ultralytics" in the .ipynb file. With the Ultralytics library in place the next step is to select appropriate YOLO labeling and YOLO segmenting models from the Ultralyitcs documentation. Figure 11 shows the available labeling or detection models. Figure 12 shows the available segmentation models. The letter "v" indicates version (here version 8) and the letters "n", "s", "m", "l", "x" correspond to "nano", "small", "medium", "large", and "extra-large" representing various model sizes and associated complexity/training times. For this project, the YOLOv8m and YOLOv8m-seg models were used. In both cases models download automatically from the latest Ultralytics release when first used in code after Ultralytics library installation. These models are PyTorch based and when used in code carry the extension, ".pt", hence "yolov8m.pt" for labeling/detection and "yolov8m-seg.pt" for segmentation.


https://docs.ultralytics.com/tasks/detect/#models
ultralytics
Detect
Search
ultralytics/ultralytics v8.2.0 22.6k 4.5k
Home Quickstart Modes Tasks Models Datasets NEW Explorer Guides Integrations HUB Reference Help
Tasks
Detect
Segment
Classify
Pose
OBB
Back to top
Table of contents
Models
Train
Dataset format
Val
Predict
Export

Models

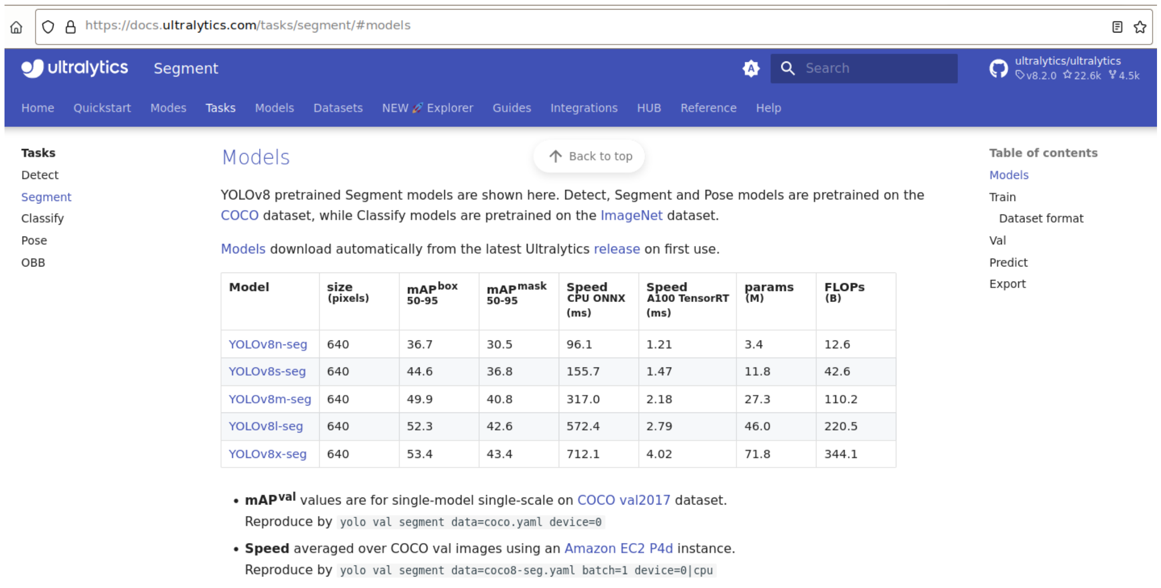
YOLOv8 pretrained Detect models are shown here. Detect, Segment and Pose models are pretrained on the COCO dataset, while Classify models are pretrained on the ImageNet dataset.

Models download automatically from the latest Ultralytics release on first use.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

- **mAP^{val}** values are for single-model single-scale on COCO val2017 dataset.
  Reproduce by `yolo val detect data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an Amazon EC2 P4d instance.
  Reproduce by `yolo val detect data=coco8.yaml batch=1 device=0|cpu`

Figure 11. Available labeling or detection yolov8 models.


https://docs.ultralytics.com/tasks/segment/#models
ultralytics
Segment
Search
ultralytics/ultralytics v8.2.0 22.6k 4.5k
Home Quickstart Modes Tasks Models Datasets NEW Explorer Guides Integrations HUB Reference Help
Tasks
Detect
Segment
Classify
Pose
OBB
Back to top
Table of contents
Models
Train
Dataset format
Val
Predict
Export

Models

YOLOv8 pretrained Segment models are shown here. Detect, Segment and Pose models are pretrained on the COCO dataset, while Classify models are pretrained on the ImageNet dataset.

Models download automatically from the latest Ultralytics release on first use.

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

- **mAP^{val}** values are for single-model single-scale on COCO val2017 dataset.
  Reproduce by `yolo val segment data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an Amazon EC2 P4d instance.
  Reproduce by `yolo val segment data=coco8-seg.yaml batch=1 device=0|cpu`

Figure 12. Available segmentation yolov8 models.

At this stage training of the YOLO labeling/detection model can be accomplished with the code:

```
###########################################################################################
# TRAIN YOLO OBJECT LABELING MODEL ########################################################
###########################################################################################
!yolo task=detect mode=train epochs=100 data=data_custom.yaml model=yolov8m.pt imgsz=640 batch=8
```

where "task" instructs the model to detect annotations, "mode" instructs the model to train with the data presented, "epochs" indicates the number of training epochs, "data" uses the previously mentioned .yaml file to find the training and validation data folders, "model" specifies the model to be used, "imgsz" sets the image size, and "batch" sets the batch size during training epochs. Training is faster on a GPU. When only a CPU is available (as in this case), training takes about 2 hours.

After the training code has run, the root directory (Figure 13) contains two new items (red boxes).

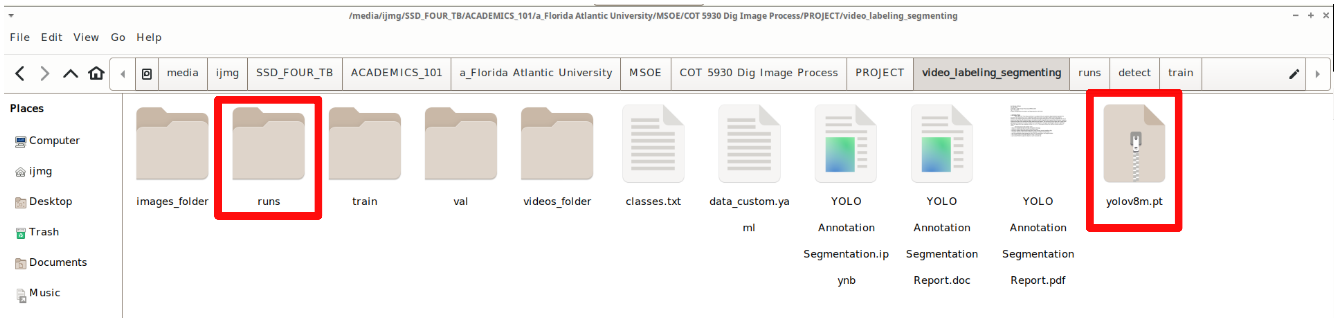

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting runs detect train
Places
Computer
ijmg
Desktop
Trash
Documents
Music
images_folder
runs
train
val
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation Segmentation.ipynb
YOLO Annotation Segmentation Report.doc
YOLO Annotation Segmentation Report.pdf
yolov8m.pt

Figure 13. Root directory after training code.

The yolov8m.pt was installed as mentioned earlier by running the above training code. The runs folder was was created from running the above training code. The runs folder contains the directory path runs/detect/train/weights. The contents of the train folder is shown below in Figure 14 and the weights folder in Figure 15. The train folder contains various outputs from the training process including a confusion matix, precision-recall curves for the object classes, training loss curves and training image samples. Most important is the weight folder which contains the last set of training model weights and best set of training model weights. The best set of training weights (best.pt) is the version of the yolov8m.pt model trained by the custom training data. This best.pt file is copied to the root directory and renamed as yolov8m_custom.pt to be used for future image and video annotation (Figure 16).

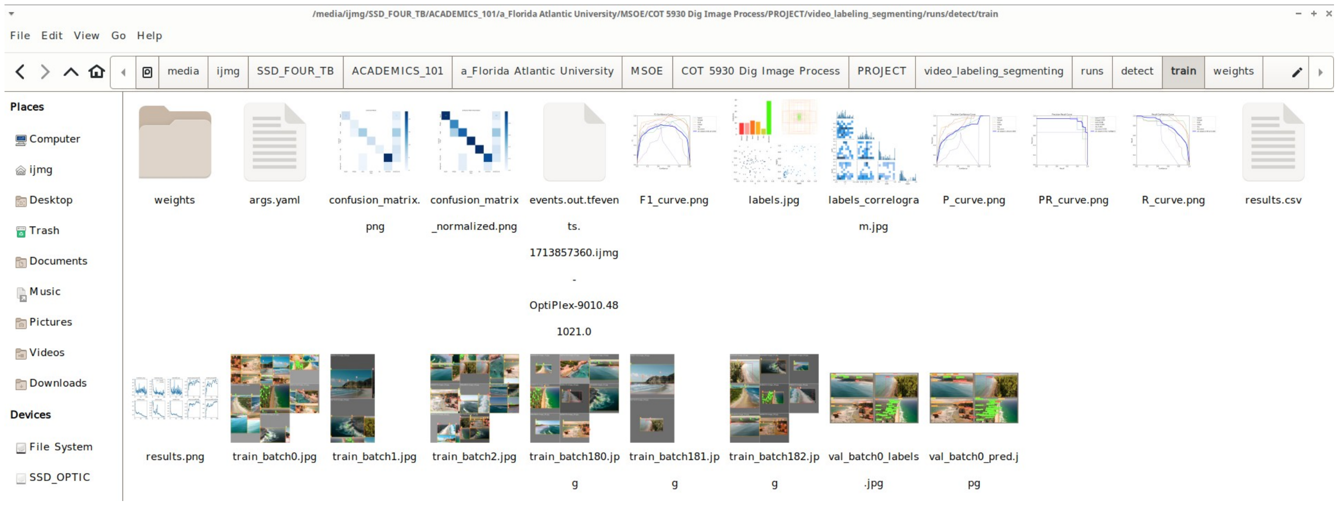

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/runs/detect/train
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting runs detect train weights
Places
Computer
ijmg
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
SSD_OPTIC
weights
args.yaml
confusion_matrix.png
confusion_matrix_normalized.png
events.out.tfevents.1713857360.ijmg-OptiPlex-9010.481021.0
F1_curve.png
labels.jpg
labels_correlogram.jpg
P_curve.png
PR_curve.png
R_curve.png
results.csv
results.png
train_batch0.jpg
train_batch1.jpg
train_batch2.jpg
train_batch180.jpg
train_batch181.jpg
train_batch182.jpg
val_batch0_labels.jpg
val_batch0_pred.jpg

Figure 14. Contents of runs/detect/train/ folder.

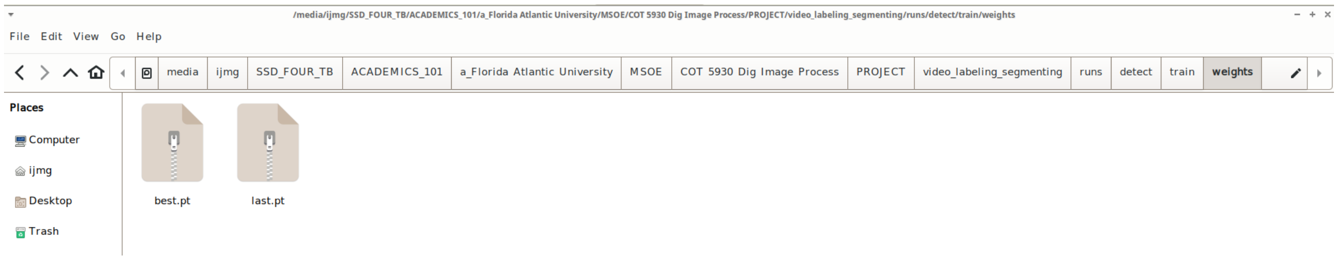

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting/runs/detect/train/weights
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting runs detect train weights
Places
Computer
ijmg
Desktop
Trash
best.pt
last.pt

Figure 15. Contents of runs/detect/train/weights folder showing best.pt custom trained YOLO model.

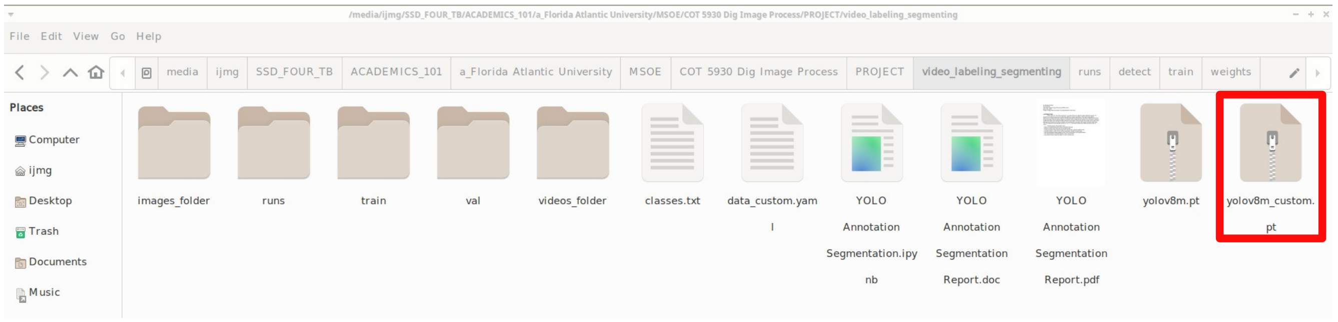
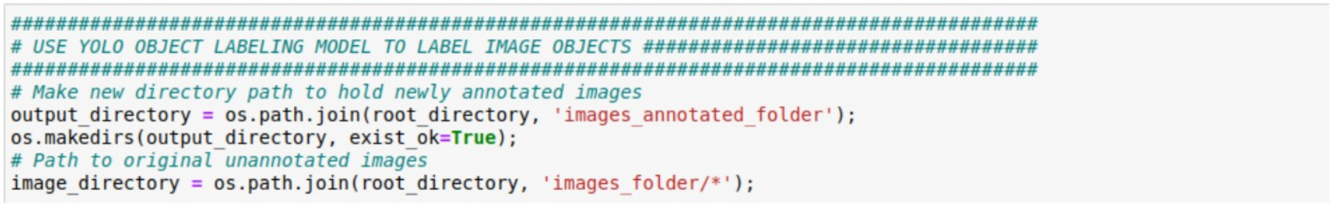
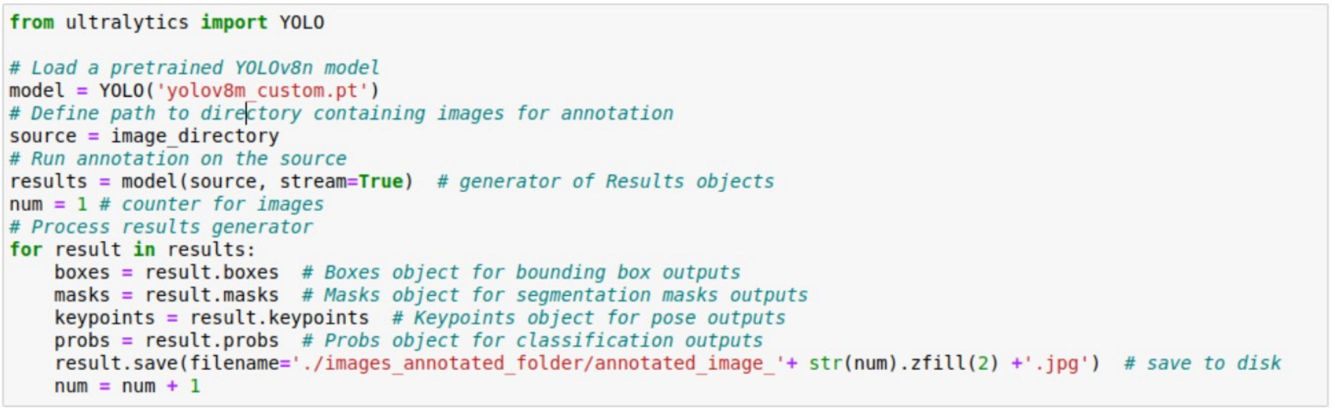

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida Atlantic University MSOE COT 5930 Dig Image Process PROJECT video_labeling_segmenting runs detect train weights
Places
Computer
ijmg
Desktop
Trash
Documents
Music
images_folder
runs
train
val
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation Segmentation.ipynb
YOLO Annotation Segmentation Report.doc
YOLO Annotation Segmentation Report.pdf
yolov8m.pt
yolov8m_custom.pt

Figure 16. Root directory showing new custom trained YOLO model, yolov8m_custom.pt (red box).

Next, the custom trained model, yolov8m_custom.pt, was used to annotate the images extracted from each 5 second clip then each of the 5 second clips. For the image annotations, the following code sections were used:



```
############################################################################################
# USE YOLO OBJECT LABELING MODEL TO LABEL IMAGE OBJECTS ###################################
############################################################################################
# Make new directory path to hold newly annotated images
output_directory = os.path.join(root_directory, 'images_annotated_folder');
os.makedirs(output_directory, exist_ok=True);
# Path to original unannotated images
image_directory = os.path.join(root_directory, 'images_folder/*');
```



```
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8m_custom.pt')
# Define path to directory containing images for annotation
source = image_directory
# Run annotation on the source
results = model(source, stream=True)  # generator of Results objects
num = 1 # counter for images
# Process results generator
for result in results:
    boxes = result.boxes  # Boxes object for bounding box outputs
    masks = result.masks  # Masks object for segmentation masks outputs
    keypoints = result.keypoints  # Keypoints object for pose outputs
    probs = result.probs  # Probs object for classification outputs
    result.save(filename='./images_annotated_folder/annotated_image_'+ str(num).zfill(2) +'.jpg')  # save to disk
    num = num + 1
```

```python
# Display annotated images
import os
import cv2
import matplotlib.pyplot as plt

# A function to display annotated images
def display_sample_images(directory, num_images=5):
    # Get list of image files in the directory
    image_files = [f for f in os.listdir(directory) if f.endswith('.jpg')]

    # Display each image
    for image_file in image_files:
        image_path = os.path.join(directory, image_file)
        image = cv2.imread(image_path)  # Read image using OpenCV
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert from BGR to RGB
        plt.imshow(image)
        plt.title(image_file)
        plt.axis('off')
        plt.show()

# Display annotated images
image_directory = os.path.join(root_directory, 'images_annotated_folder')
num_images=10
display_sample_images(image_directory, num_images)
```

These code sections created the root directory folder, images_annotated_folder (Figure 17) filled with annotated images shown in Figure 18 and sample-displayed in the .ipynb file.

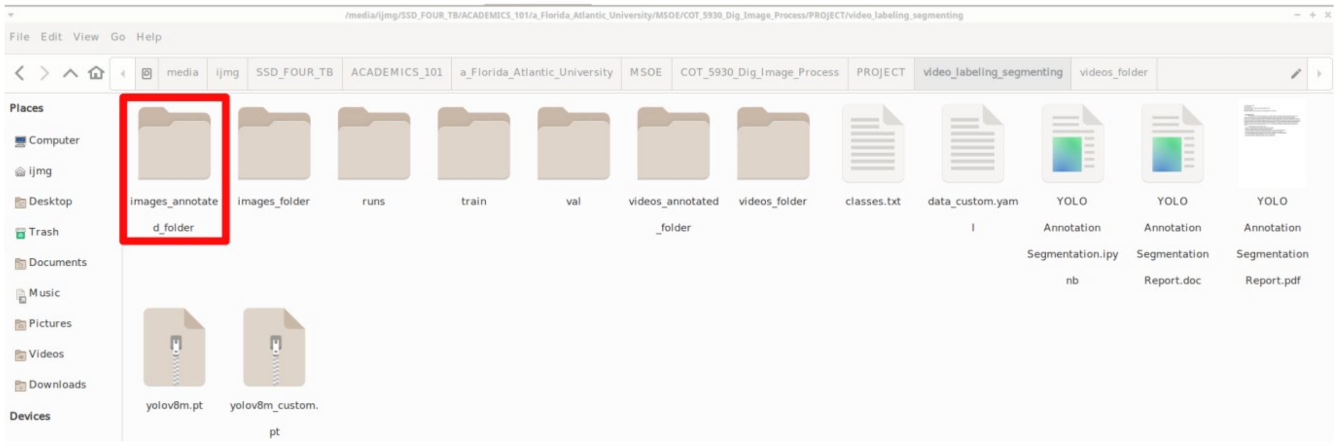

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting videos_folder
Places
Computer
ijmg
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
images_annotated_folder
images_folder
runs
train
val
videos_annotated_folder
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation Segmentation.ipynb
YOLO Annotation Segmentation Report.doc
YOLO Annotation Segmentation Report.pdf
yolov8m.pt
yolov8m_custom.pt

Figure 17. Root directory showing new images_annotated_folder (red box).

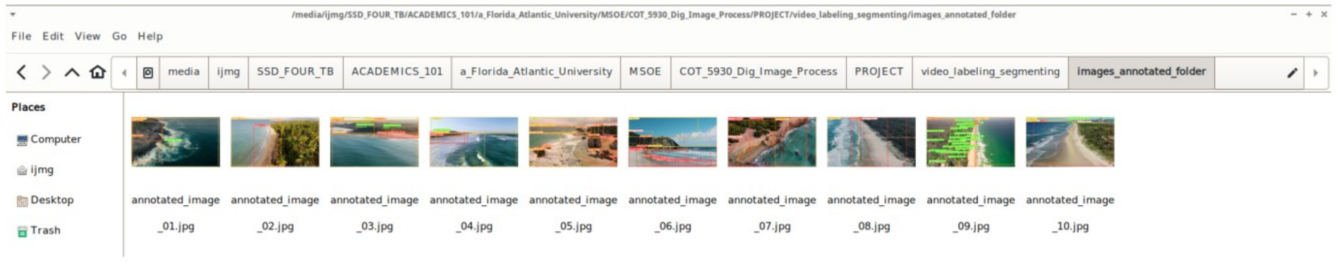

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting/images_annotated_folder
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting images_annotated_folder
Places
Computer
ijmg
Desktop
Trash
annotated_image_01.jpg
annotated_image_02.jpg
annotated_image_03.jpg
annotated_image_04.jpg
annotated_image_05.jpg
annotated_image_06.jpg
annotated_image_07.jpg
annotated_image_08.jpg
annotated_image_09.jpg
annotated_image_10.jpg

Figure 18. Annotated images within images_annotated_folder (red box) in Figure 17.

For the video annotations, the following code sections were used:

```python
###########################################################################################
# USE YOLO OBJECT LABELING MODEL TO LABEL VIDEO OBJECTS ###################################
###########################################################################################
# Make new directory path to hold newly annotated videos
output_directory = os.path.join(root_directory, 'videos_annotated_folder');
os.makedirs(output_directory, exist_ok=True);
# Path to original unannotated videos
video_directory = os.path.join(root_directory, 'videos_folder');
```

```python
# List all files in the video directory
video_files = sorted(os.listdir(video_directory))
# Print the list of video files
print("Files in the directory:")
for video_file in video_files:
    print(video_file)
```

```python
# A function to join .jpg images into an .mp4 file
def join_images_to_video(path_to_images, video_file):
    # List all .jpg files in the directory containing .jpg images
    images = [img for img in os.listdir(path_to_images) if img.endswith(".jpg")]

    # Sort the images based on their filenames
    images.sort()

    # Determine the width and height of the first image
    img = cv2.imread(os.path.join(path_to_images, images[0]))
    height, width, _ = img.shape

    # Define the output video file name
    video_name = os.path.join(path_to_images, 'annotated_' + video_file)

    # Define the video codec
    codec = cv2.VideoWriter_fourcc(*'mp4v')

    # Create a video writer object
    fps = 30 # Frames per second
    video_writer_object = cv2.VideoWriter(video_name, codec, fps, (width, height))

    # Iterate through each image and add it to the video
    for image in images:
        img_path = os.path.join(path_to_images, image)
        frame = cv2.imread(img_path)
        video_writer_object.write(frame)
        # Delete the image file after it has been added to the video
        os.remove(img_path)

    # Release the video writer
    video_writer_object.release()

    print(f"Video saved as {video_name}")
```

```python
# Load a pretrained YOLOv8n model
model = YOLO('yolov8m_custom.pt')
for video_file in video_files:
    # Define path to directory containing videos for annotation
    source = os.path.join(video_directory, video_file)
    # Run annotation on the source
    results = model(source, stream=True)
    num = 1 # counter for images
    for result in results:
        boxes = result.boxes  # Boxes object for bounding box outputs
        masks = result.masks  # Masks object for segmentation masks outputs
        keypoints = result.keypoints  # Keypoints object for pose outputs
        probs = result.probs  # Probs object for classification outputs
        # YOLO is only able to save annotated .jpg images (??)
        result.save(filename='./videos_annotated_folder/annotated_video_image_'+ str(num).zfill(3) + '.jpg' )
        num = num + 1
    # Join the annotated images into final annotated .mp4 video clip
    #   then delete the separate annotated images from the directory
    path_to_images = './videos_annotated_folder/'
    join_images_to_video(path_to_images, video_file)
```

These code sections created the root directory folder, videos_annotated_folder (Figure 19) filled with annotated video clips shown in Figure 20. This code section also made use of a function, join_images_to_video() to address the unexpected inability of the custom trained YOLO model to directly construct annotated .mp4 files. Instead, the model constructed a series of annotated .jpg images which were then joined into an .mp4 file using the function, join_images_to_video().

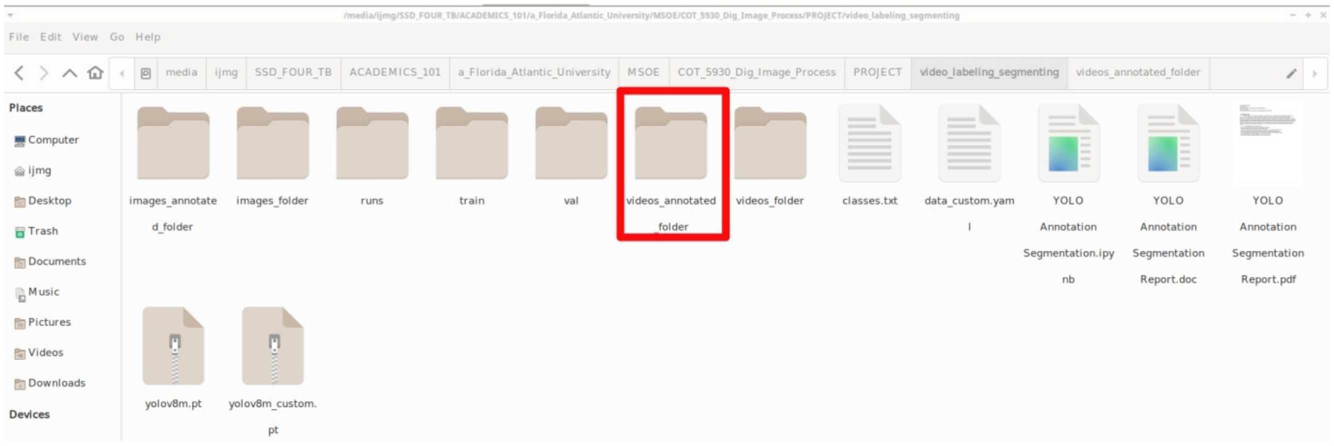

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting videos_annotated_folder
Places
Computer
ijmg
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
images_annotated_folder
images_folder
runs
train
val
videos_annotated_folder
videos_folder
classes.txt
data_custom.yaml
YOLO Annotation Segmentation.ipynb
YOLO Annotation Segmentation Report.doc
YOLO Annotation Segmentation Report.pdf
yolov8m.pt
yolov8m_custom.pt

Figure 19. Root directory showing new videos_annotated_folder (red box).

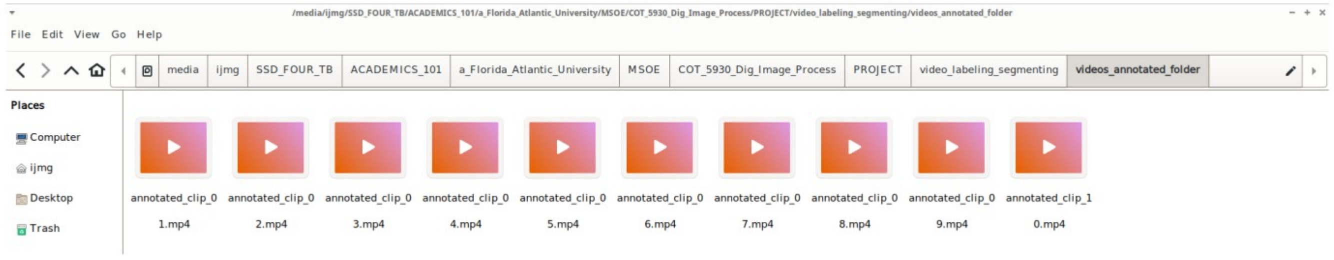

/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting/videos_annotated_folder
File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting videos_annotated_folder
Places
Computer
ijmg
Desktop
Trash
annotated_clip_01.mp4
annotated_clip_02.mp4
annotated_clip_03.mp4
annotated_clip_04.mp4
annotated_clip_05.mp4
annotated_clip_06.mp4
annotated_clip_07.mp4
annotated_clip_08.mp4
annotated_clip_09.mp4
annotated_clip_10.mp4

Figure 20. Annotated 5 second video files within videos_annotated_folder (red box) in Figure 19.

At this stage training of the YOLO segmentation model can be accomplised with the code:

```
##########################################################################################
# TRAIN YOLO OBJECT SEGMENTATION MODEL ###################################################
##########################################################################################
!yolo task=segment mode=train epochs=100 data=data_custom.yaml model=yolov8m-seg.pt imgsz=640 batch=8
```

where "task" instructs the model to segment annotations, "mode" instructs the model to train with the data presented, "epochs" indicates the number of training epochs, "data" uses the previously mentioned .yaml file to find the training and validation data folders, "model" specifies the model to be used, "imgsz" sets the image size, and "batch" sets the batch size during training epochs. Training is faster on a GPU. When only a CPU is available (as in this case), training takes about 2 hours.

After the training code has run, the root directory (Figure 21) will contain the model yolov8m-seg.pt (indicated by the red box). The yolov8m-seg.pt model was installed by running the above training code. The runs folder will now contain an additional directory path runs/segment/train2/weights. The contents of the train2 folder (numbered as 2 due to an aborted and deleted train1 run) is shown below in Figure 22 and the weights folder in Figure 23. As before, the train folder contains various outputs from the training process including a confusion matix, precision-recall curves for the object classes, training loss curves and training image samples. Most important is the weight folder which contains the last set of training model weights and best set of training model weights. The best set of training weights is the version of the yolov8m-seg.pt model trained by the custom training data. This best.pt file is again copied to the root directory but this time renamed as yolov8m-seg_custom.pt to be used for future image and video segmentation (Figure 21, green box).

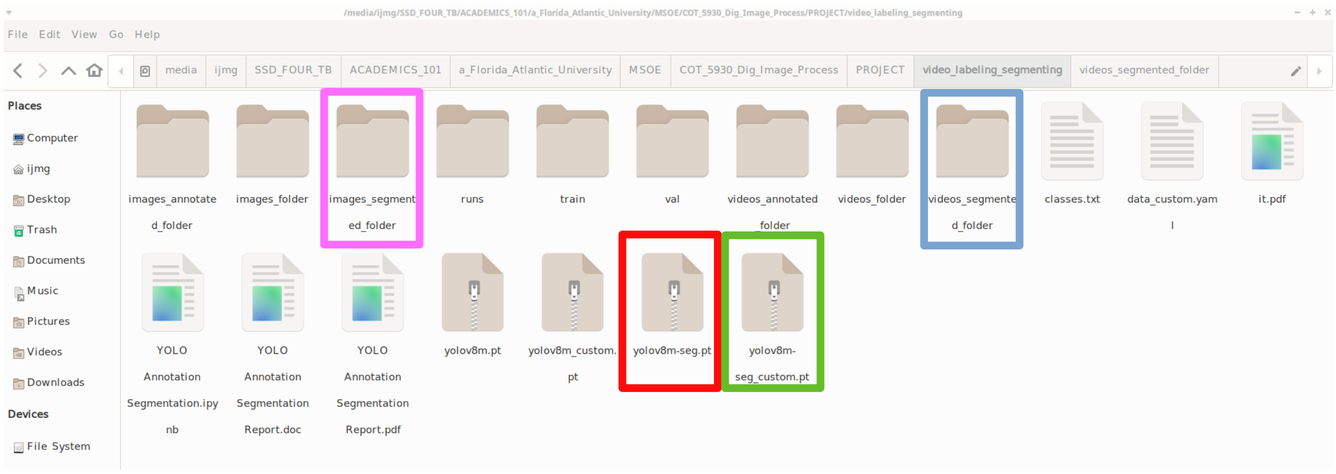
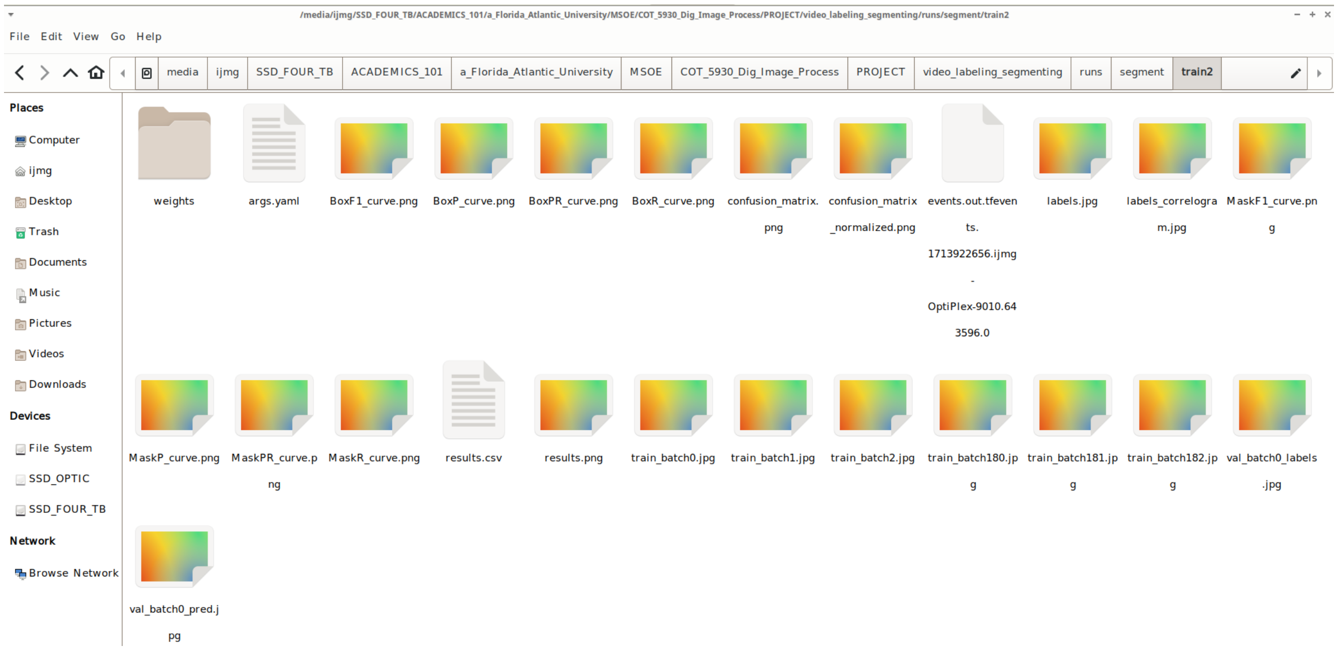


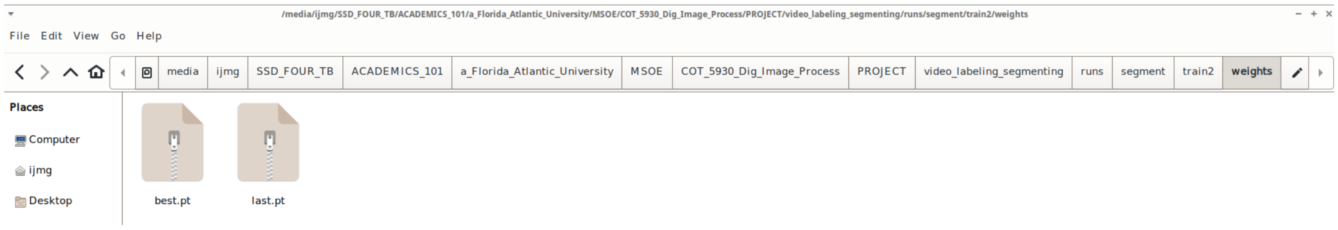
Figure 21. Root directory with yolov8m-seg.pt model (red box), yolov8m-seg_custom.pt model (green box), images_segmented_folder (purple box), and videos_segmented_folder (light blue box) identified.


/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting/runs/segment/train2

File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting runs segment train2
Places
Computer
ijmg
Desktop
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
SSD_OPTIC
SSD_FOUR_TB
Network
Browse Network

weights, args.yaml, BoxF1_curve.png, BoxP_curve.png, BoxPR_curve.png, BoxR_curve.png, confusion_matrix.png, confusion_matrix_normalized.png, events.out.tfevents.1713922656.ijmg-OptiPlex-9010.643596.0, labels.jpg, labels_correlogram.jpg, MaskF1_curve.png, MaskP_curve.png, MaskPR_curve.png, MaskR_curve.png, results.csv, results.png, train_batch0.jpg, train_batch1.jpg, train_batch2.jpg, train_batch180.jpg, train_batch181.jpg, train_batch182.jpg, val_batch0_labels.jpg, val_batch0_pred.jpg

Figure 22. Contents of runs/segment/train2 folder


/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting/runs/segment/train2/weights

File Edit View Go Help
media ijmg SSD_FOUR_TB ACADEMICS_101 a_Florida_Atlantic_University MSOE COT_5930_Dig_Image_Process PROJECT video_labeling_segmenting runs segment train2 weights
Places
Computer
ijmg
Desktop

best.pt, last.pt

Figure 23. Contents of runs/segment/train2/weights folder

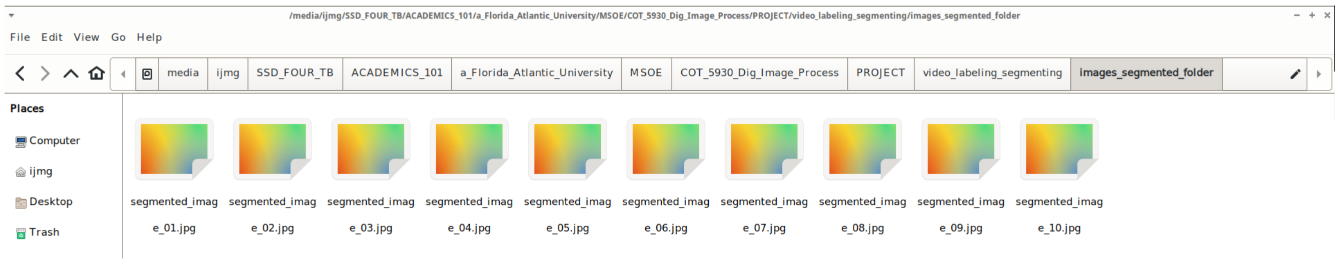
Next, the custom trained model, yolov8m-seg_custom.pt, was used to segment the images extracted from each 5 second clip then each of the 5 second clips. For the image segmentations, this was done using the code sections,

```
##############################################################################################
# USE YOLO OBJECT SEGMENTATION MODEL TO SEGMENT IMAGE OBJECTS ##############################
##############################################################################################
# Make new directory path to hold newly segmented images
output_directory = os.path.join(root_directory, 'images_segmented_folder');
os.makedirs(output_directory, exist_ok=True);
# Path to original unannotated images
image_directory = os.path.join(root_directory, 'images_folder/*');
```

```python
from ultralytics import YOLO

# Load a custom YOLOv8n model
model = YOLO('yolov8m-seg_custom.pt')
# Define path to directory containing images for annotation
source = image_directory
# Run annotation on the source
results = model(source, stream=True)  # generator of Results objects
num = 1 # counter for images
# Process results generator
for result in results:
    boxes = result.boxes  # Boxes object for bounding box outputs
    masks = result.masks  # Masks object for segmentation masks outputs
    keypoints = result.keypoints  # Keypoints object for pose outputs
    probs = result.probs  # Probs object for classification outputs
    result.save(filename='./images_segmented_folder/segmented_image_'+ str(num).zfill(2) +'.jpg')
    num = num + 1
```

These code sections created the root directory folder, images_segmented_folder (Figure 21, purple box) filled with segmented images shown in Figure 24 and sample-displayed in the .ipynb file.



/media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida_Atlantic_University/MSOE/COT_5930_Dig_Image_Process/PROJECT/video_labeling_segmenting/images_segmented_folder

File Edit View Go Help

media | ijmg | SSD_FOUR_TB | ACADEMICS_101 | a_Florida_Atlantic_University | MSOE | COT_5930_Dig_Image_Process | PROJECT | video_labeling_segmenting | images_segmented_folder

Places
Computer
ijmg
Desktop
Trash

segmented_image_01.jpg
segmented_image_02.jpg
segmented_image_03.jpg
segmented_image_04.jpg
segmented_image_05.jpg
segmented_image_06.jpg
segmented_image_07.jpg
segmented_image_08.jpg
segmented_image_09.jpg
segmented_image_10.jpg

Figure 24. Annotated images within images_segmented_folder (purple box) in Figure 21.

For the video segmentations, the following code sections were used:

```python
#########################################################################################
# USE YOLO OBJECT SEGMENTATION MODEL TO SEGMENT VIDEO OBJECTS ###########################
#########################################################################################
# Make new directory path to hold newly segmented videos
output_directory = os.path.join(root_directory, 'videos_segmented_folder');
os.makedirs(output_directory, exist_ok=True);
# Path to original unannotated videos
video_directory = os.path.join(root_directory, 'videos_folder');
```

```python
# List all files in the video directory
video_files = sorted(os.listdir(video_directory))
# Print the list of video files
print("Files in the directory:")
for video_file in video_files:
    print(video_file)
```

```python
# A function to join .jpg images into an .mp4 file
def join_images_to_video(path_to_images, video_file):
    # List all .jpg files in the directory containing .jpg images
    images = [img for img in os.listdir(path_to_images) if img.endswith(".jpg")]

    # Sort the images based on their filenames
    images.sort()

    # Determine the width and height of the first image
    img = cv2.imread(os.path.join(path_to_images, images[0]))
    height, width, _ = img.shape

    # Define the output video file name
    video_name = os.path.join(path_to_images, 'annotated_' + video_file)

    # Define the video codec
    codec = cv2.VideoWriter_fourcc(*'mp4v')

    # Create a video writer object
    fps = 30 # Frames per second
    video_writer_object = cv2.VideoWriter(video_name, codec, fps, (width, height))

    # Iterate through each image and add it to the video
    for image in images:
        img_path = os.path.join(path_to_images, image)
        frame = cv2.imread(img_path)
        video_writer_object.write(frame)
        # Delete the image file after it has been added to the video
        os.remove(img_path)

    # Release the video writer
    video_writer_object.release()

    print(f"Video saved as {video_name}")
```

```python
# Load a custom YOLOv8n model
model = YOLO('yolov8m-seg_custom.pt')
for video_file in video_files:
    # Define path to directory containing videos for segmentation
    source = os.path.join(video_directory, video_file)
    # Run annotation on the source
    results = model(source, stream=True)
    num = 1 # counter for images
    for result in results:
        boxes = result.boxes  # Boxes object for bounding box outputs
        masks = result.masks  # Masks object for segmentation masks outputs
        keypoints = result.keypoints  # Keypoints object for pose outputs
        probs = result.probs  # Probs object for classification outputs
        # YOLO is only able to save annotated .jpg images (??)
        result.save(filename='./videos_segmented_folder/segmented_video_image_'+ str(num).zfill(3) + '.jpg' )
        num = num + 1
    # Join the annotated images into final annotated .mp4 video clip
    #   then delete the separate annotated images from the directory
    path_to_images = './videos_segmented_folder/'
    join_images_to_video(path_to_images, video_file)
```

These code sections created the root directory folder, videos_segmented_folder (Figure 21, light blue box) filled with segmented images shown in Figure 25. This code section also made use of a slightly modified version of the function, join_images_to_video() to address the unexpected inability of the custom trained YOLO model to directly construct annotated .mp4 files. Instead, the model constructed a series of annotated .jpg images which were then joined into an .mp4 file using the function, join_images_to_video().

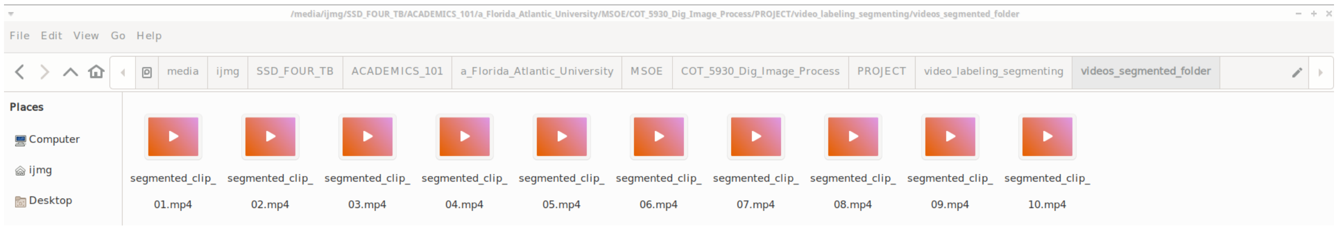


Figure 25. Segmented videos within videos_segmented_folder (light blue box) in Figure 21.

IV. RESULTS / CONCLUSION

Two models were successfully trained, a custom segmentation model (yolov8m-seg_custom.pt) and a custom annotation model (yolov8m_custom.pt). Both worked fairly well with images and video clips. Segmented training images are located in the folder, images_segmented_folder. Segmented training videos are located in the folder, videos_segmented_folder. Annotated training images are located in the folder, images_annotated_folder. Annotated training videos are located in the folder, videos_annotated_folder. The final task was to test the new custom segmentation model (yolov8m-seg_custom.pt) on similar but unseen coastline video clips. Twenty-three additional 5 second coastline clips were obtained and placed in the root directory folder, videos_test_folder. They were then fed into the yolov8m-seg_custom.pt model and segmented output clips stored in the root directory folder, videos_test_segmented_folder. When evaluating the segmented output video clips it is helpful to adjust video playback speed to about 60% of normal. Reviewing the yolov8m-seg_custom.pt model's attempts to segment completely new and unseen video clips revealed several patterns:

1.) Segmentation is more accurate with fewer object classes present in a video frame.
2.) Segmentation is more accurate when the coastline runs in a near vertical, linear pattern in a video frame.
3.) Segmentation is more accurate when the captured drone footage was not from too great an altitude in a video frame.
4.) Segmentation of islands and peninsulas can sometimes be difficult.
5.) Segmentation/separation of land and foliage can sometimes be difficult with low lying plant growth or large, isolated trees.
6) Segmentation/separation of large, calm areas of ocean from beach space can sometimes be difficult.

This project was conducted using simple CPU computational resources. Ideally GPU resources should be used for more efficient use of time and the ability to train the models with larger data sets. An unexpected benefit of the project was an introduction to the Label Studio video/image annotation tool. The tool is online, open source, useful for video/image and non-video/image labeling, and seems to have a strong community of users.