17.07 – RESEARCHING EXISTING DATASETS ON LITERATURE

DATASET PROVIDERS:
OPENAIRE. : BAD SAT IMG
KAGGLE : earthquake damage detection , disaster aerial imagery
GOOGLE DATASET post-earthquake debris drone imagery dataset, satellite imagery disaster detection labeled dataset

xBD dataset

MAIN DATASET REPOSITORY:
https://github.com/satellite-image-deep-learning/datasets?tab=readme-ov-file

SATELLITE:
MAXAR:
https://github.com/opengeos/maxar-open-data

TURKEY EARTHQUAKE SATELLITE IMG TUE-CD MSI-NET :
https://github.com/RSMagneto/MSI-Net?tab=readme-ov-file

*** xBD - XVIEW2 (Needs to register):

https://xview2.org/dataset
PAPER: https://arxiv.org/pdf/1911.09296

*** Rescuenet

https://www.kaggle.com/datasets/yaroslavchyrko/rescuenet/data
PAPER: https://arxiv.org/pdf/2202.12361

* Kaggle Türkiye Earthquake

https://www.kaggle.com/datasets/buraktaci/turkiye-earthquake-2023
Low res

18.07.2024- GATHERING AND EXPLORING DATASETS

THERE ARE 2 MAIN DATASETS

1- RESCUENET https://arxiv.org/pdf/2202.12361
2- XBD https://xview2.org/dataset

Github repository: https://gitlab.bewelltech.com.tr/Intern4/visdrone-object-detection

RescueNet: A High Resolution UAV Semantic Segmentation Dataset for Natural Disaster Damage Assessment

Recent advancements in computer vision and deep learning techniques have facilitated notable progress in scene understanding, thereby assisting rescue teams in achieving precise damage assessment. In this paper, we present RescueNet, a meticulously curated high-resolution post-disaster dataset that includes detailed classification and semantic segmentation annotations. This dataset aims to facilitate comprehensive scene understanding in the aftermath of natural disasters. RescueNet comprises post-disaster images collected after Hurricane Michael, obtained using Unmanned Aerial Vehicles (UAVs) from multiple impacted regions. The uniqueness of RescueNet lies in its provision of high-resolution post-disaster imagery, accompanied by comprehensive annotations for each image. Unlike existing datasets that offer annotations limited to specific scene elements such as buildings, RescueNet provides pixel-level annotations for all classes, including buildings, roads, pools, trees, and more. Furthermore, we evaluate the utility of the dataset by implementing state-of-the-art segmentation models on RescueNet, demonstrating its value in enhancing existing methodologies for natural disaster damage assessment. The dataset can be downloaded from https://www.dropbox.com/scl/fo/ntgeyhxe2mzd2wuh7he7x/AHJ-cNzQL-Eu04HS6bvBgcw?rlkey=6vxiaqve9gp6vzvzh3t5mz0vv&e=1&dl=0

Table 1. Overview of existing natural disaster datasets.

Dataset	Size	Resolution	Image Type	Task	# of Annotated Classes
ABCD[14]	22171	Varies	Satellite	Classification	2
Chen *et al.*[8]	-	Varies	Satellite	Object Detection	2
fMoW[21]	$\sim$ 1 million	Varies	Satellite	Classification	63
xBD[9]	22068	1024x1024	Satellite	Instance Segmentation, Classification	4
ISBDA[13]	1030	-	Aerial (Social Media)	Object Detection	3
AIDER[18]	2545	-	UAV	Classification	4
FloodNet[10]	2343	3000x4000	UAV	Semantic Segmentation, Classification	9
RescueNet	**4494**	**3000x4000**	**UAV**	**Semantic Segmentation, Classification**	**10**

Table 2. Number of polygons of different buildings based on their damage levels.

Damage Level	Number of Polygons
No Damage	4011
Medium Damage	3119
Major Damage	1693
Total Destruction	2080

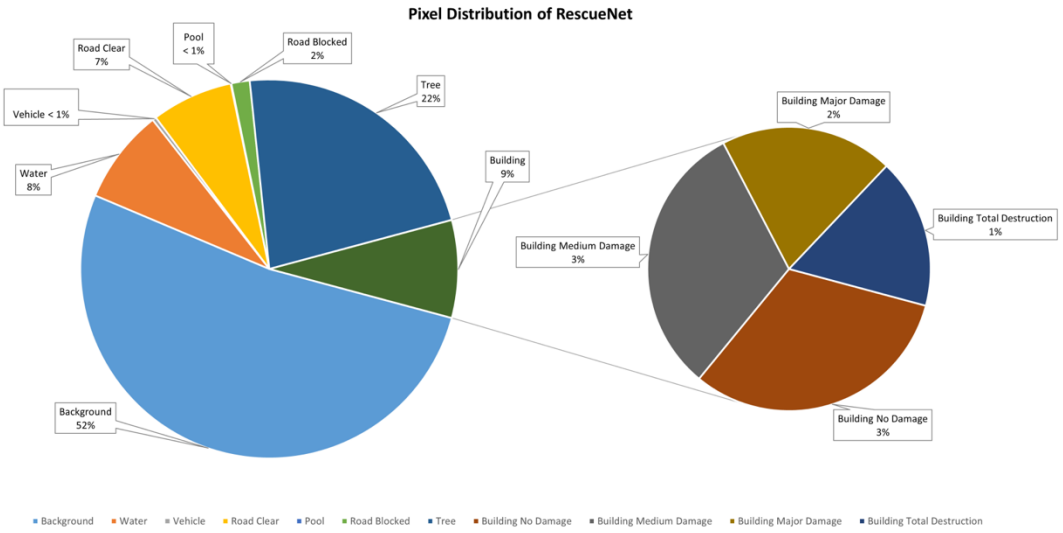


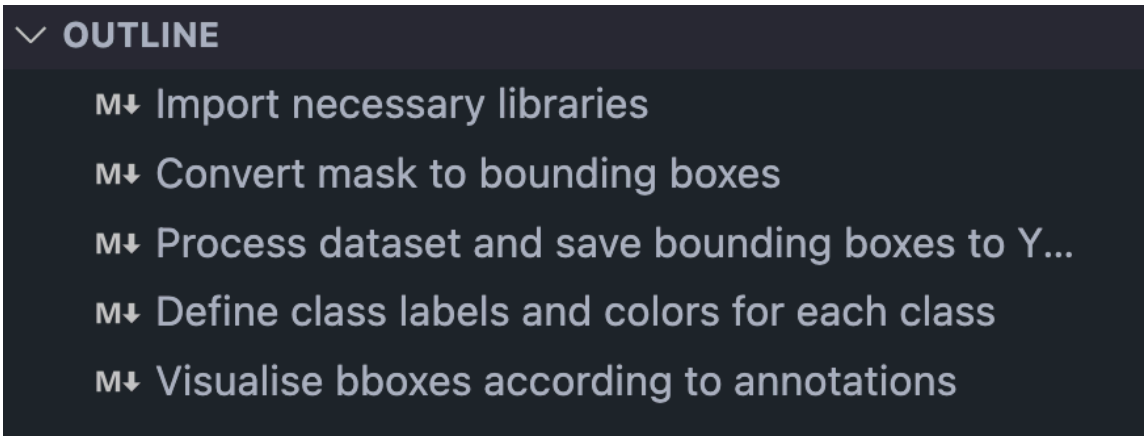
Figure 2. Pixel distribution of different classes in RescueNet.

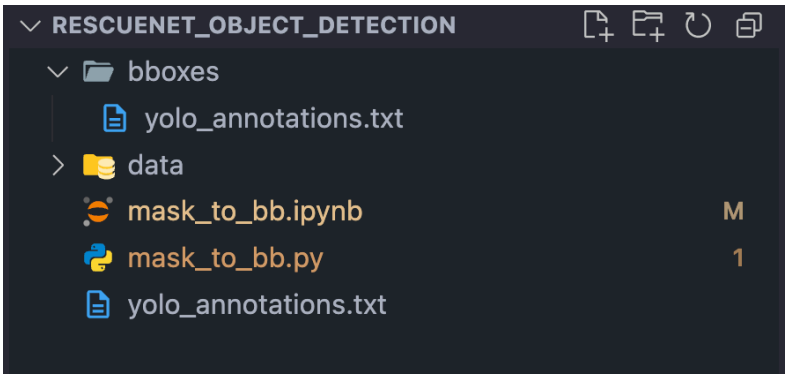
VERSION NO. = RescueNet_v1.0
##----------
Features:

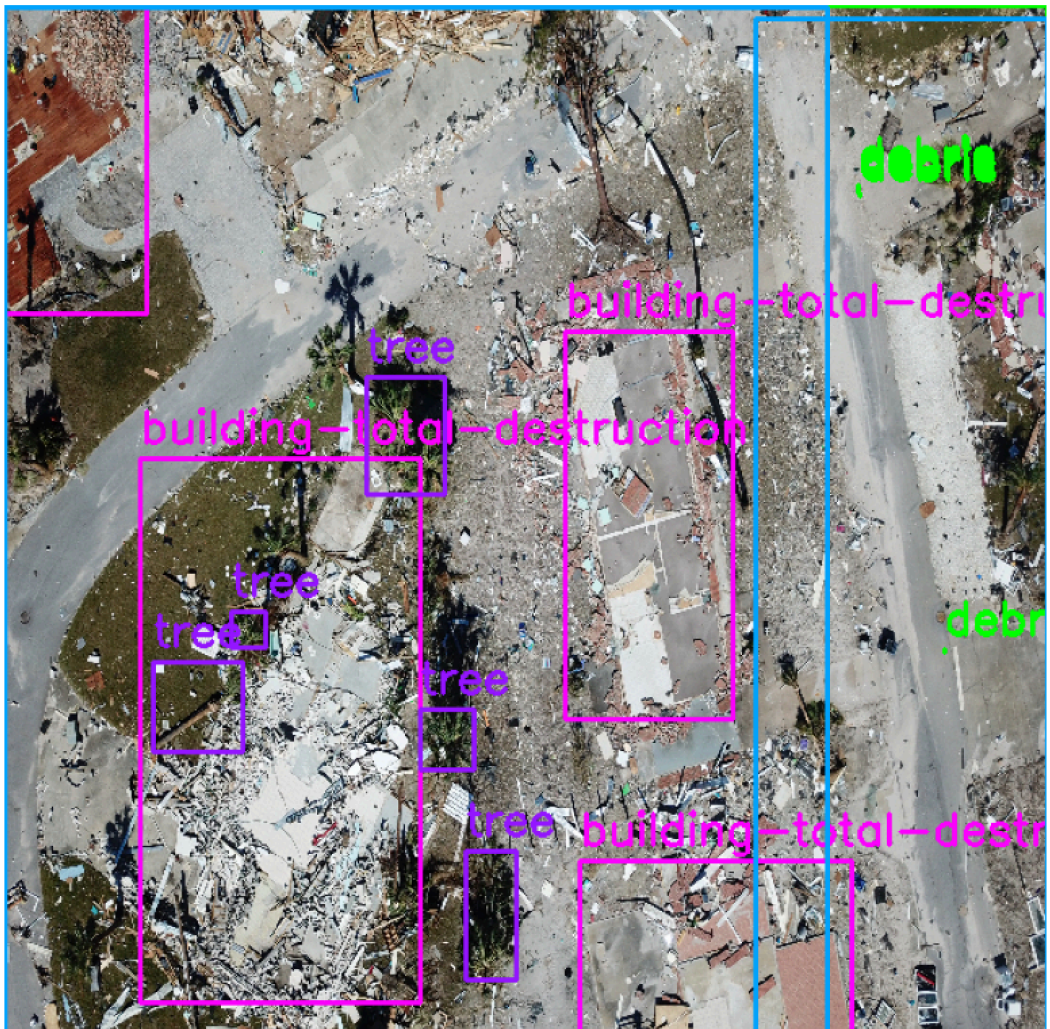
Total class: 11
'Background':0,
'Debris':1,
'Water':2,
'Building_No_Damage':3,
'Building_Minor_Damage':4,
'Building_Major_Damage':5,
'Building_Total_Destruction':6,
'Vehicle':7,
'Road':8,
'Tree':9,
'Pool':10,
'Sand':11).

Total image: 4494 (Train: 3595, Val: 449, Test: 450)

The class labels are annotated in mask format. For object detection bboxes needs to be digged from label images. So the ultimate goal of this day is **converting masks to bounding boxes.**


OUTLINE
Import necessary libraries
Convert mask to bounding boxes
Process dataset and save bounding boxes to Y...
Define class labels and colors for each class
Visualise bboxes according to annotations


RESCUENET_OBJECT_DETECTION
bboxes
yolo_annotations.txt
data
mask_to_bb.ipynb
M
mask_to_bb.py
1
yolo_annotations.txt

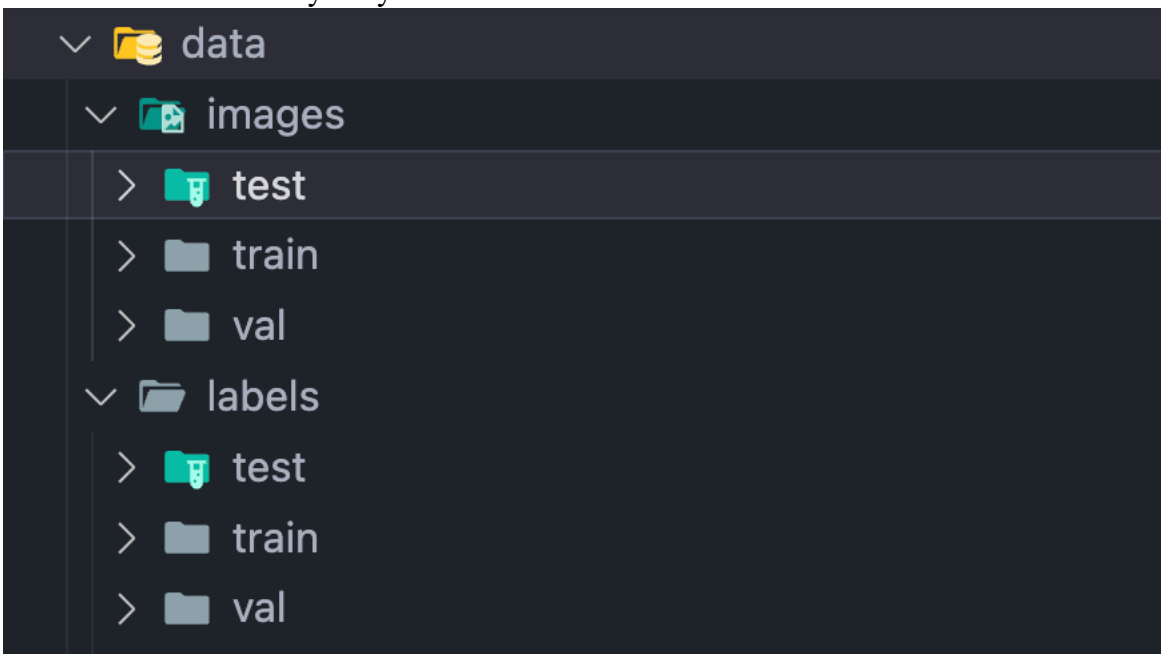

tree
tree
tree
tree
tree

Digging the rest of data
train val test
80 – 10 – 10
Annotations have been created in a txt file.
Data is set. creating corresponding and seperate bboxes txt files will be done on next day

19.07- TESTING ON YOLOV5 AND YOLOV8 MODELS

-creating labels for yolo with python script. see yolo_annotation_handler.py

Creating clean annotation.txt files foreach image for YOLO with help of python script.
New data structure ready for yolo models



dataset.yaml:

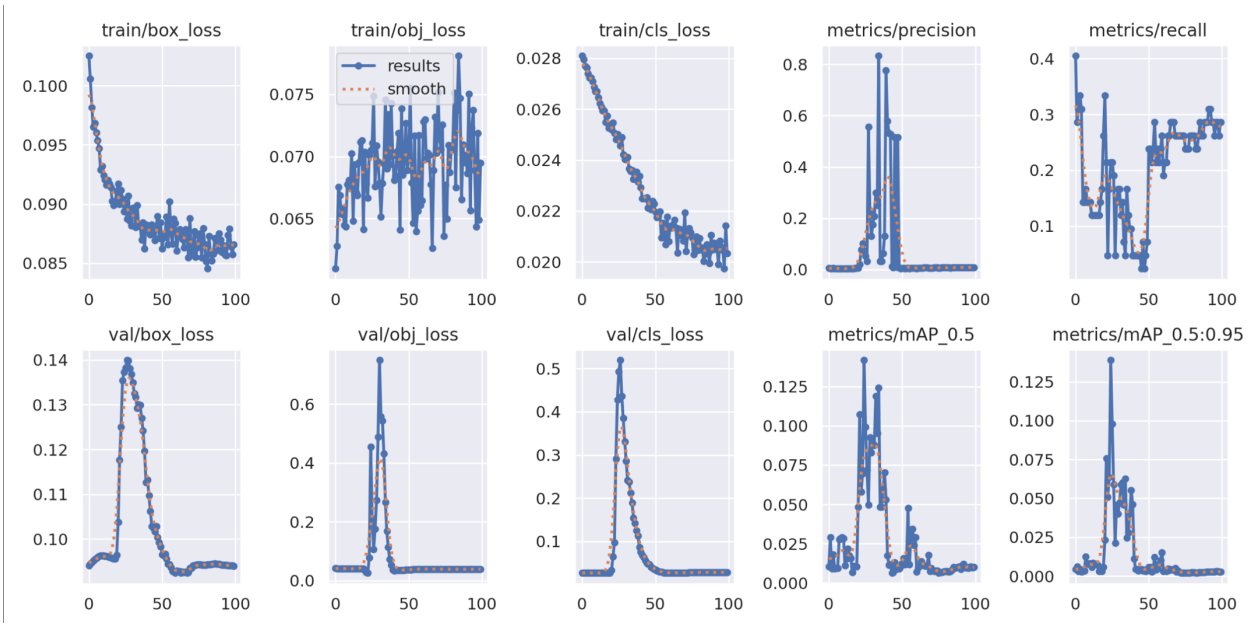
```yaml
path: /content/drive/MyDrive/data

train: images/train
val: images/val
test: images/test

nc: 2

names: ['damaged-building','building-no-damage']
```

yolo v5 and yolov8 models are tested in colab env. outputs can be found under yolos directory

data uploaded to drive and colab enviroment with T4 GPU is prepared.
Results are saved under yolos directory

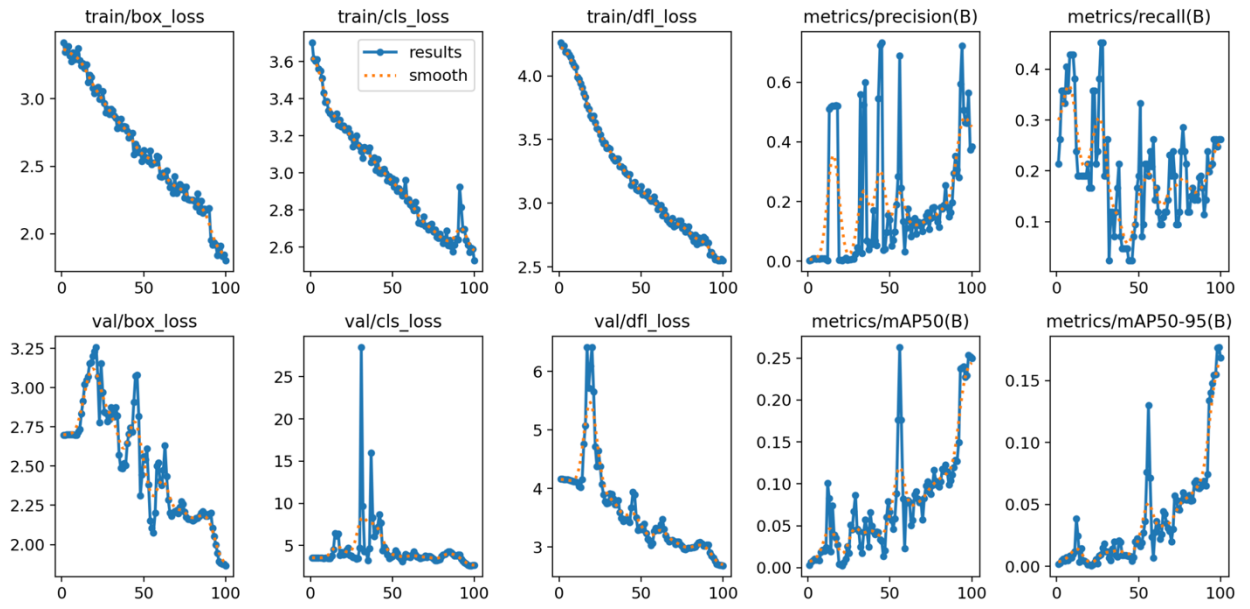
Yolov5:

```
!python train.py --img 640 --batch 16 --epochs 100 --data dataset.yaml --weights '' --
cfg models/yolov5s.yaml --device 0 --cache
```



Yolov8:

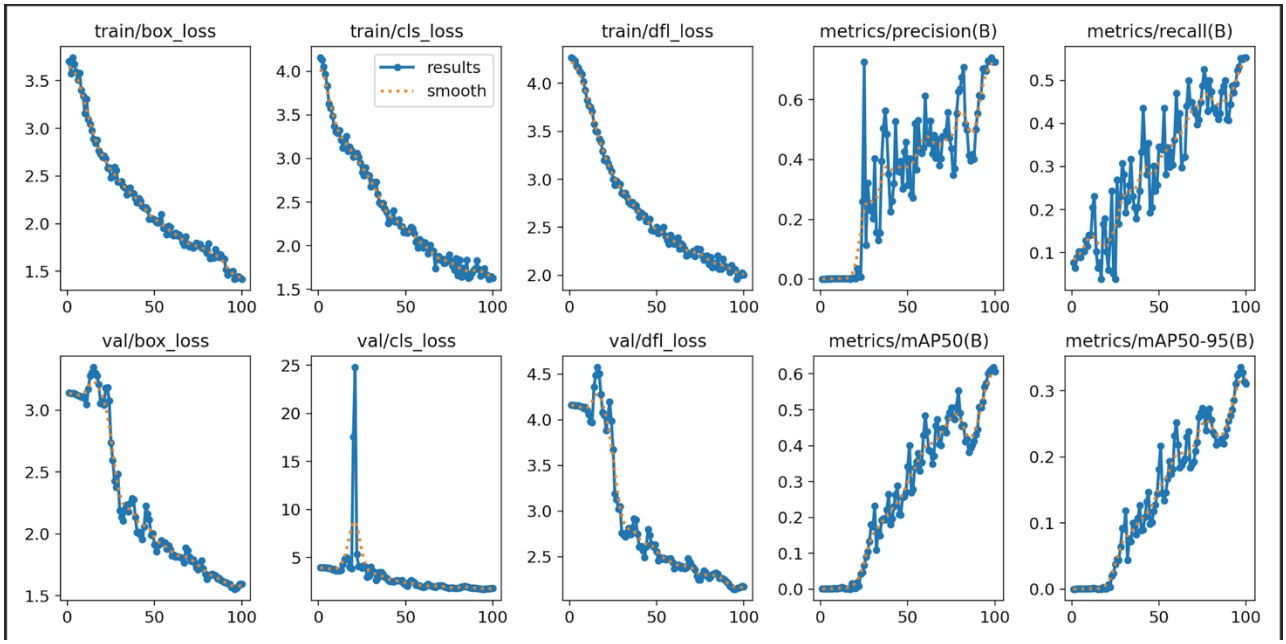
```
# Initialize the model
model = YOLO('yolov8n.yaml')  # Using the 'n' (nano) version of YOLOv8 for fast
training. You can use 's', 'm', 'l', 'x' for larger models.

# Train the model
model.train(data='/content/drive/MyDrive/data/dataset.yaml', epochs=100, imgsz=640,
batch=16, cache = True)
```



22.07- FIXING THE BBOX PROBLEMS AND RETESTING ON THE YOLOV8S MODEL

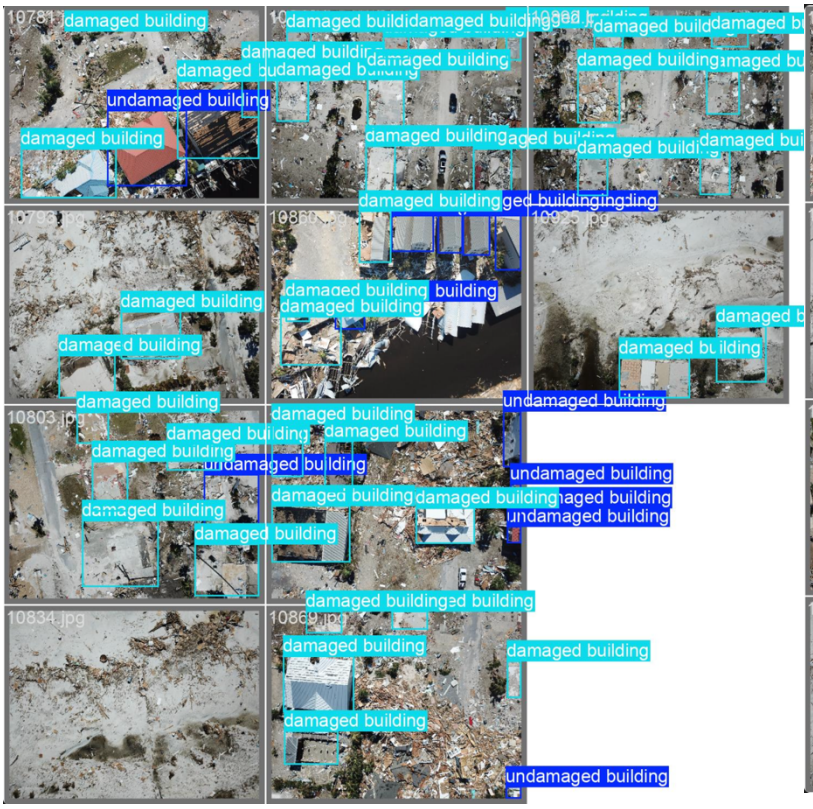
Bounding boxes were problematic(some of the bounding boxes doesnt appear in new annotation txt file) and a new script has been written to achieve more accurate bboxes.
See “bbox_fix.ipynb” inside preprocess directory.

YOLOV8S Model test outputs on Google Colab notebook (you can see the full run outputs on yolos/yolov8_fixed_annotation folder):



LABEL

PREDICTION



These results have been obtained with only 100 images. More images will make the model more accurate about predictions.

TODO **EXTEND THE DATASET!

QuickQuakeBuildings: Post-earthquake SAR-Optical Dataset for Quick Damaged-building Detection

https://arxiv.org/abs/2312.06587
https://github.com/ya0-sun/posteq-saropt-buildingdamage?tab=readme-ov-file

Deep Learning tools in Building Detection from Drone & Satellite imagery

https://www.linkedin.com/pulse/deep-learning-tools-building-detection-from-drone-satellite-/

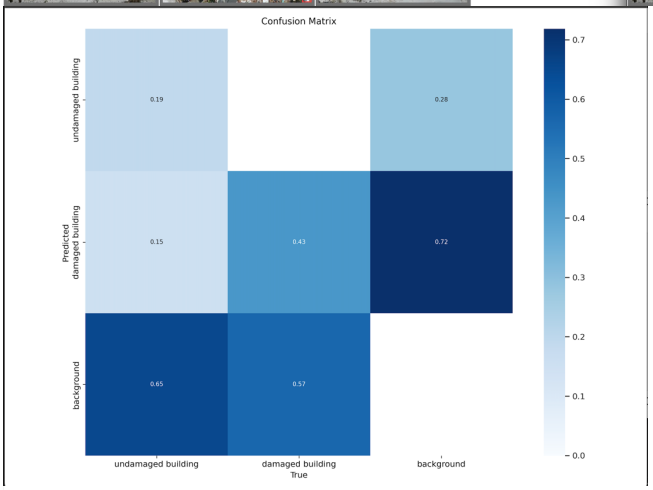
Faster_RCNN_for_DOTA

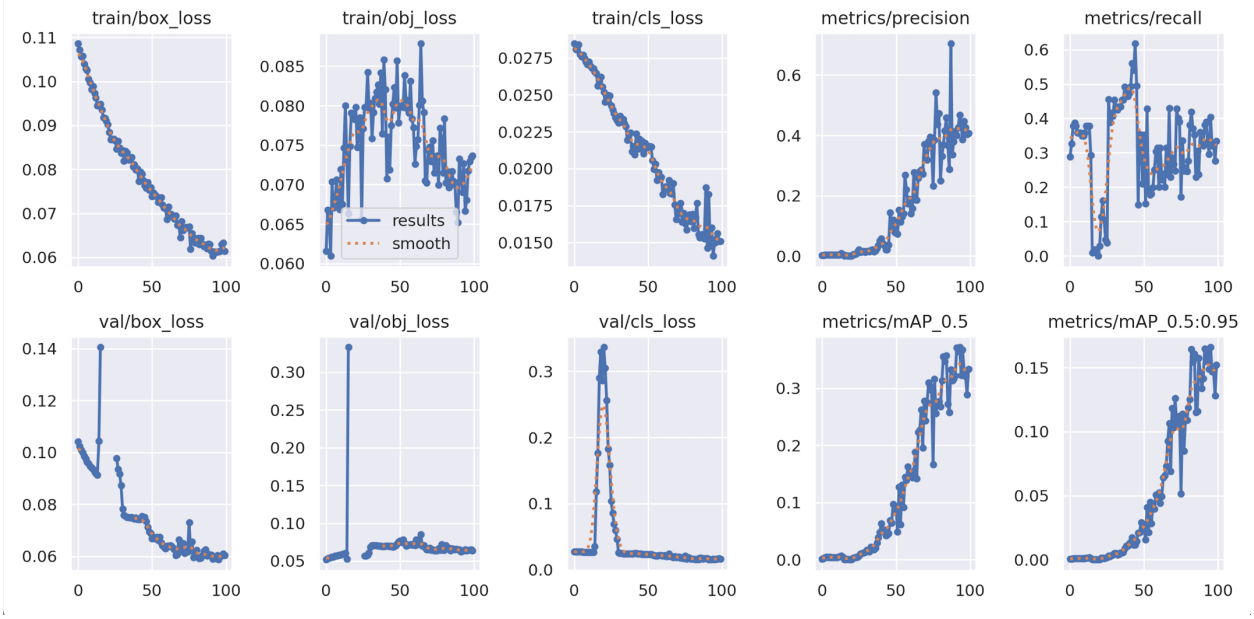
https://modelzoo.co/model/faster_rcnn_for_dota

23.07- TESTS TESTS TESTS

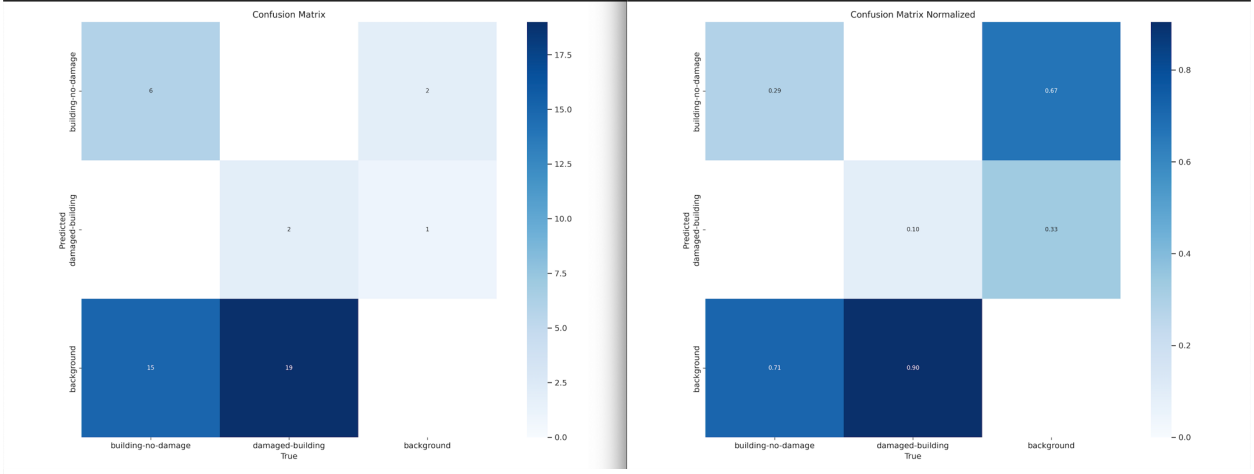
YOLOV10 BASE TEST
EXTEND THE DATASET:
+22 Train / +5 test/ +5 val
Total: train: 102 val: 15 test: 15
YOLOV10S BS:16 – 32
YOLOV8S BS:16 – 32
YOLOV5M BS:16

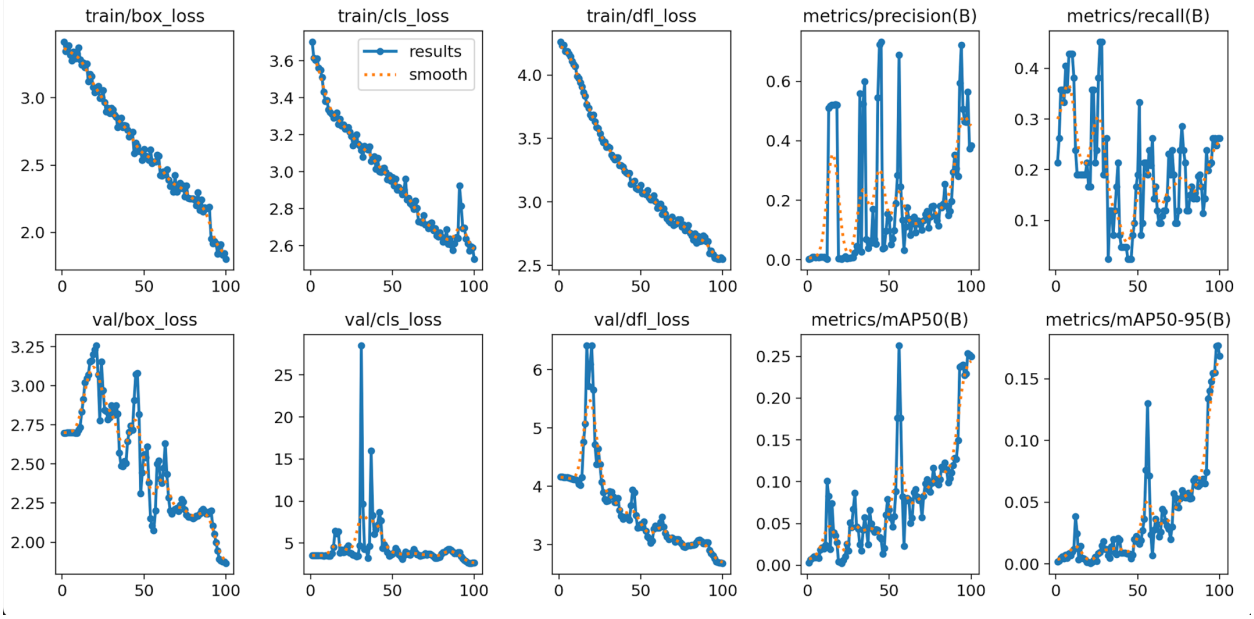
YOLOV5M BS:16



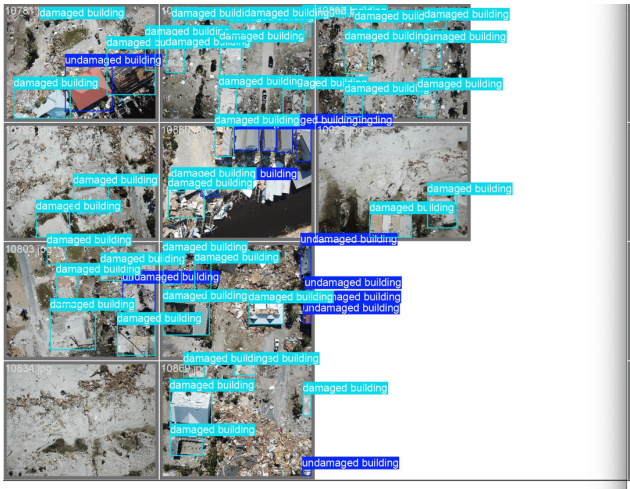


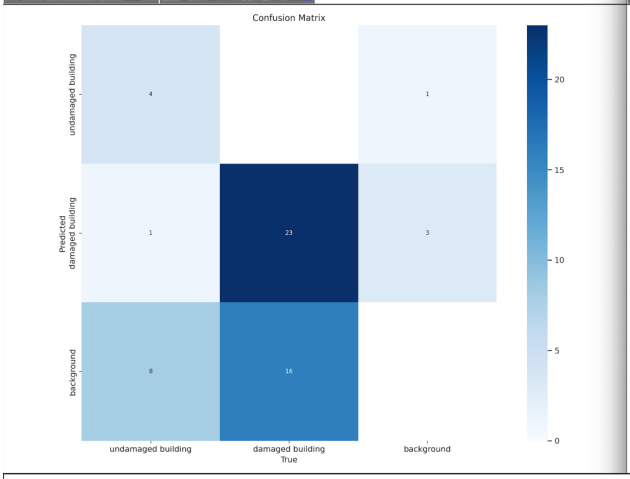
yolov8n_bs16_annotation_fail

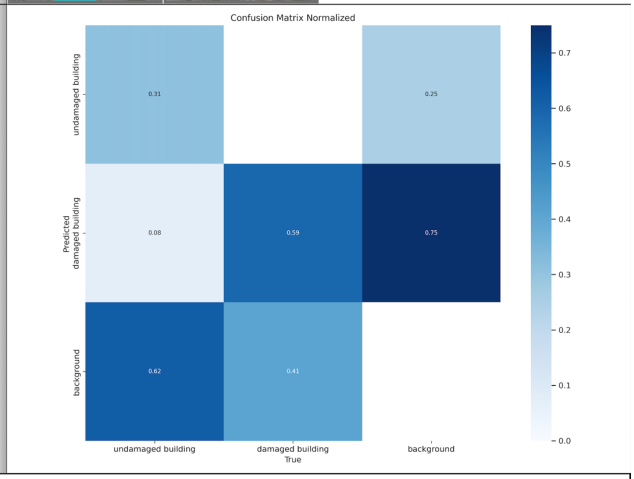


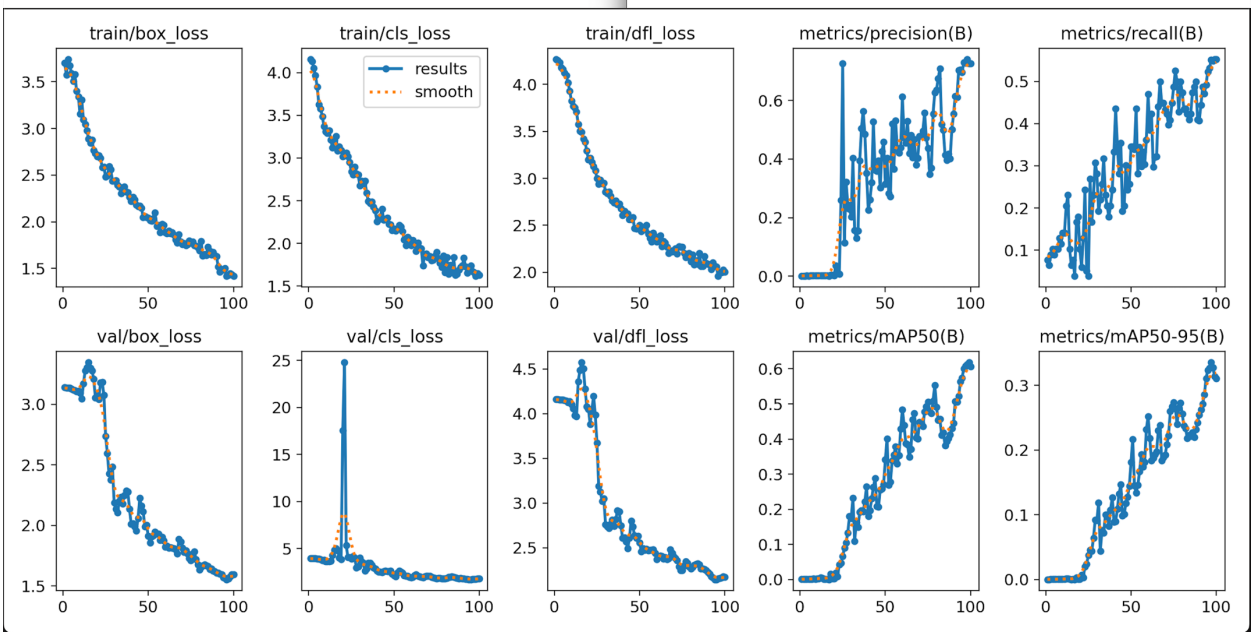


yolov8n_fixed_annotation

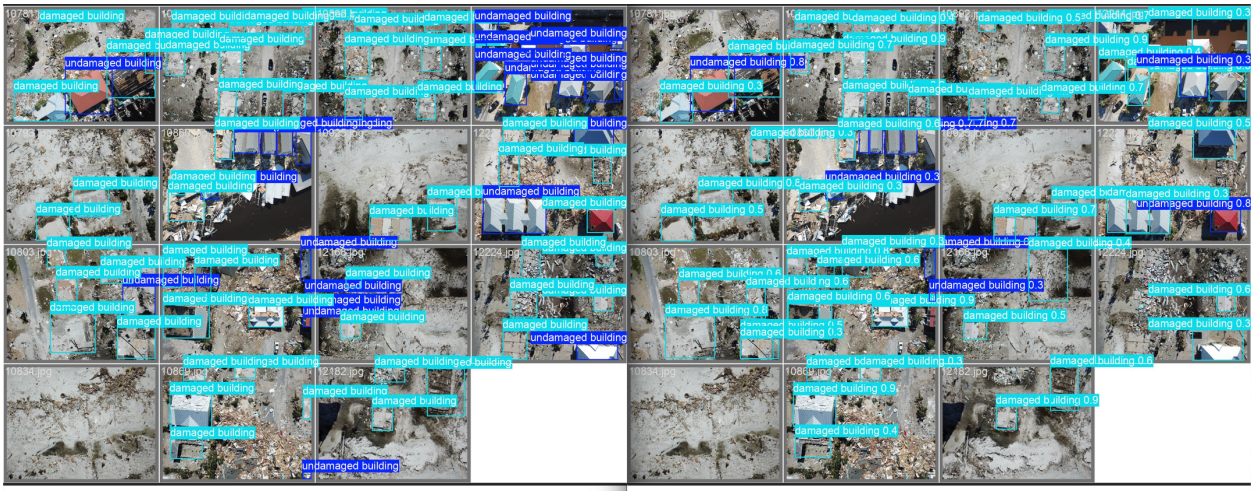


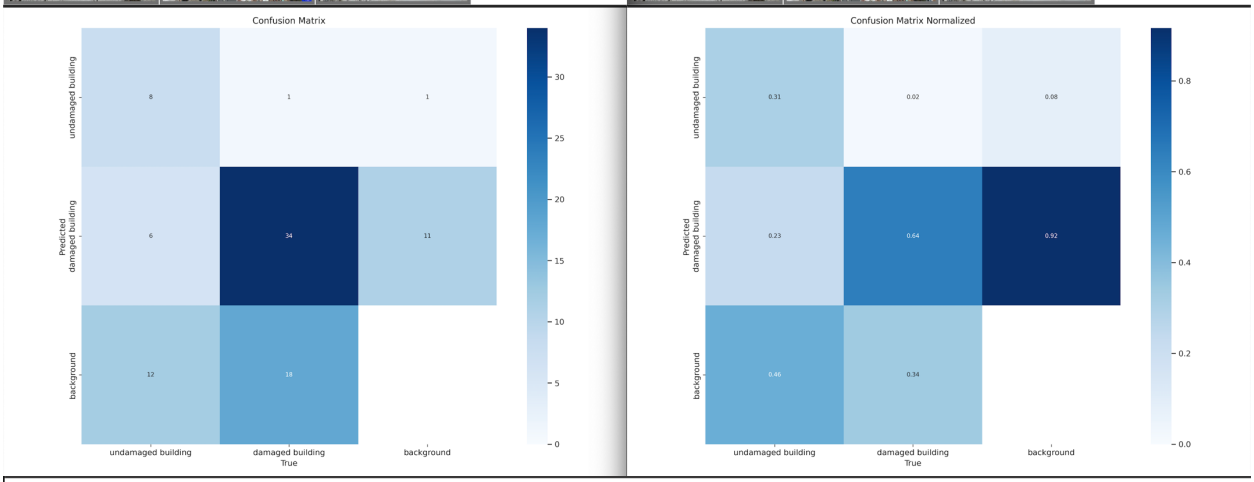


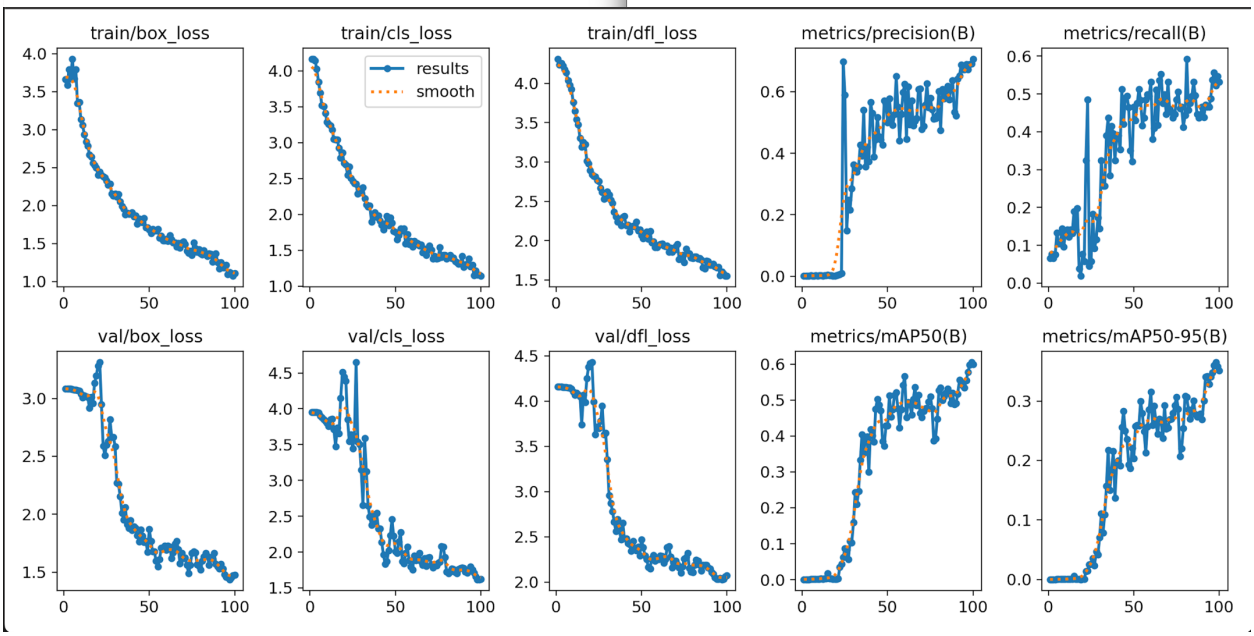




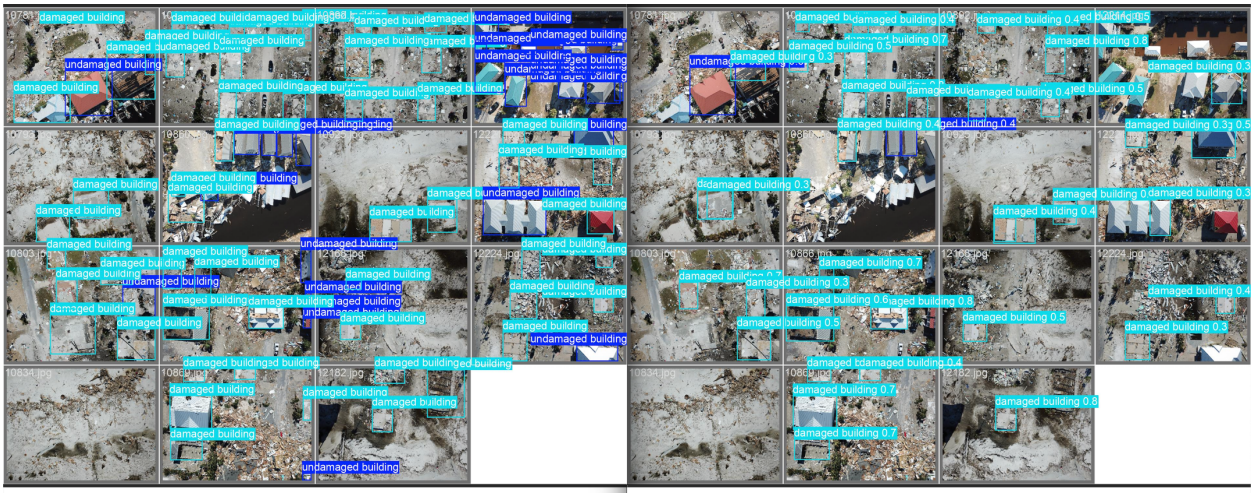
yolov8s_bs32_extended:

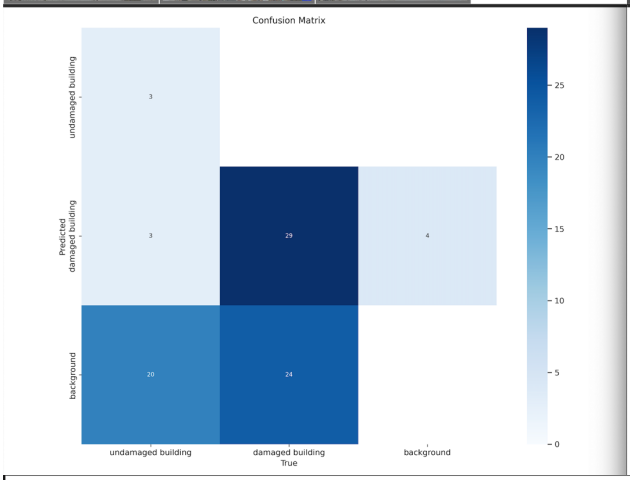


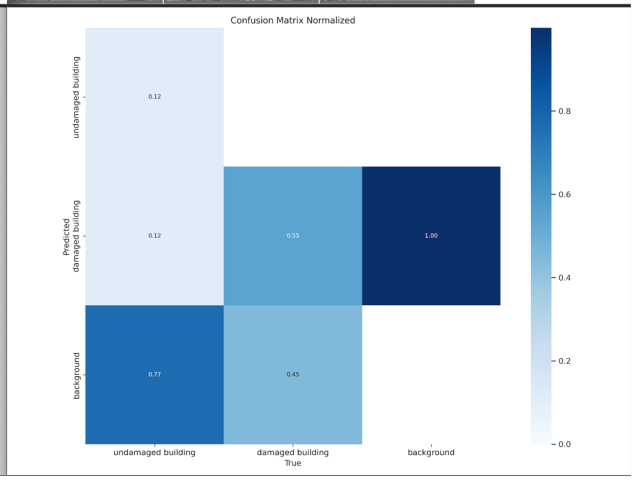


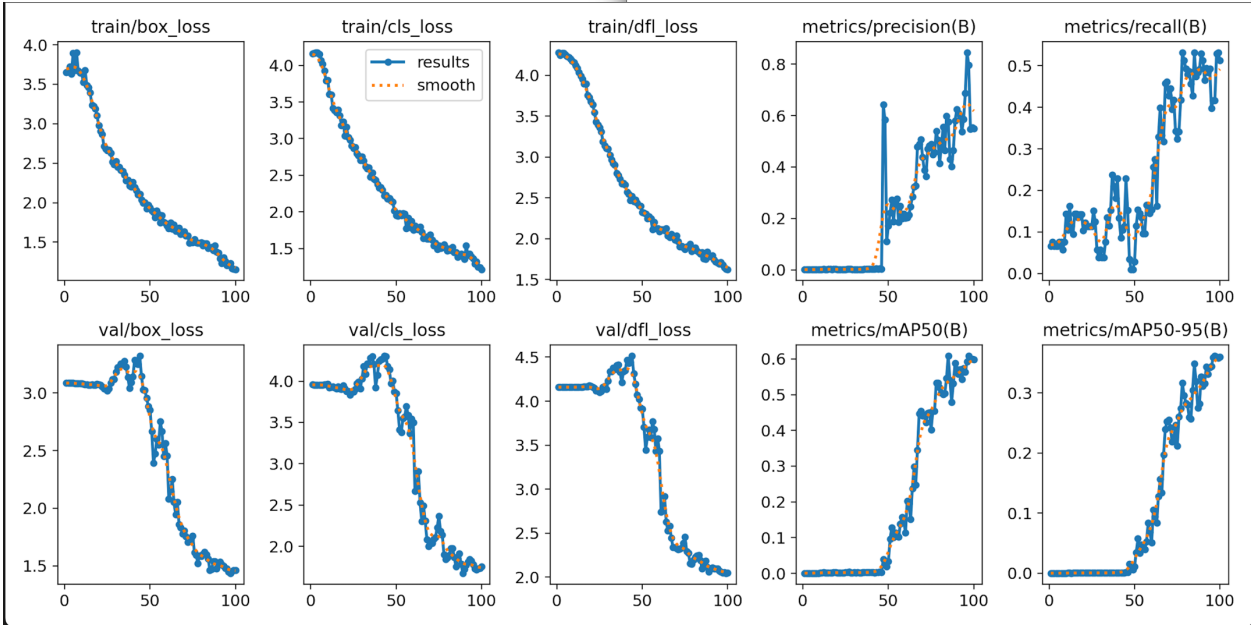


yolov8s_bs64_extended:

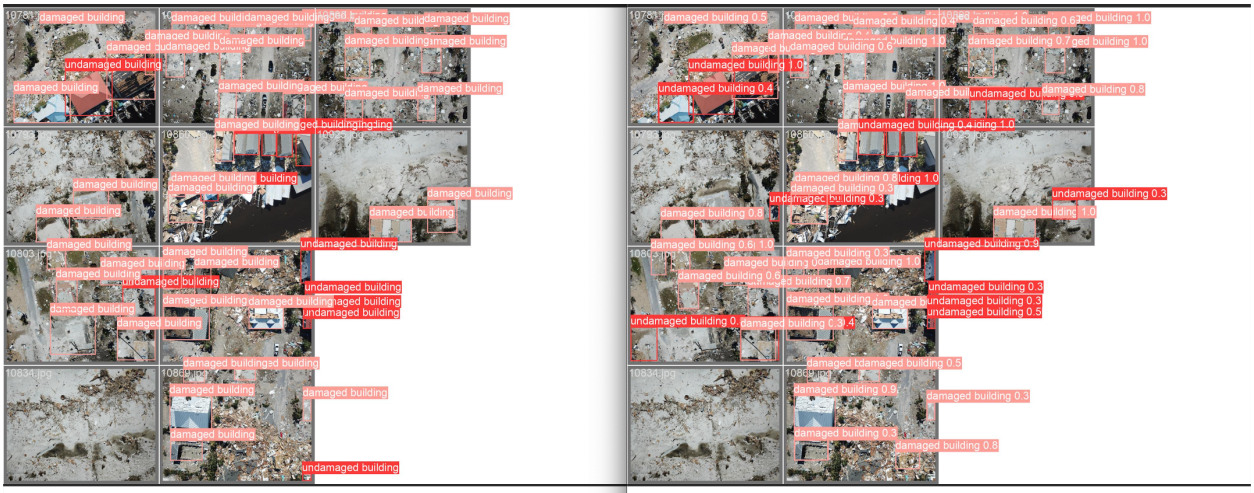


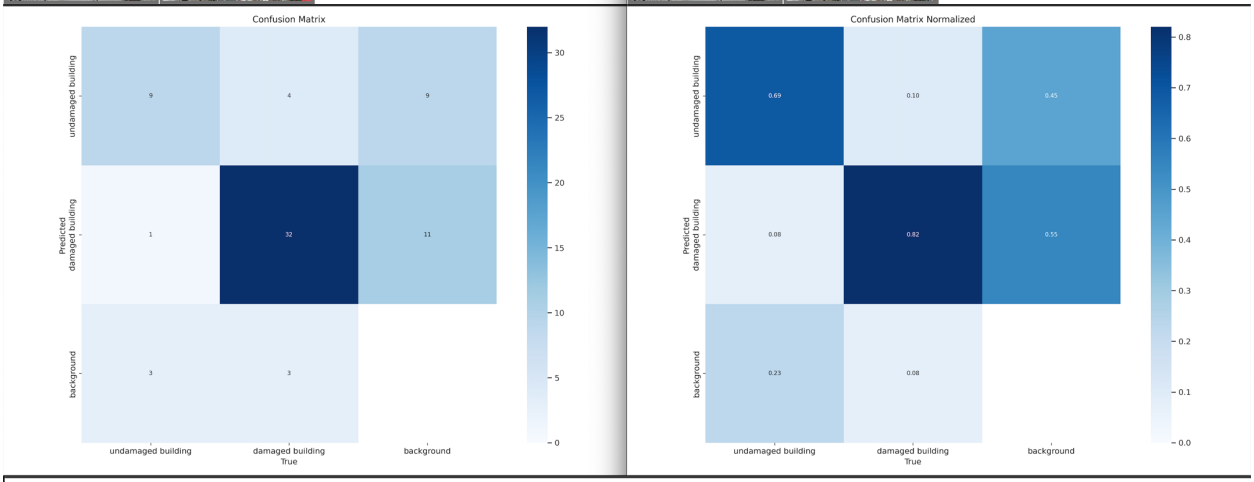


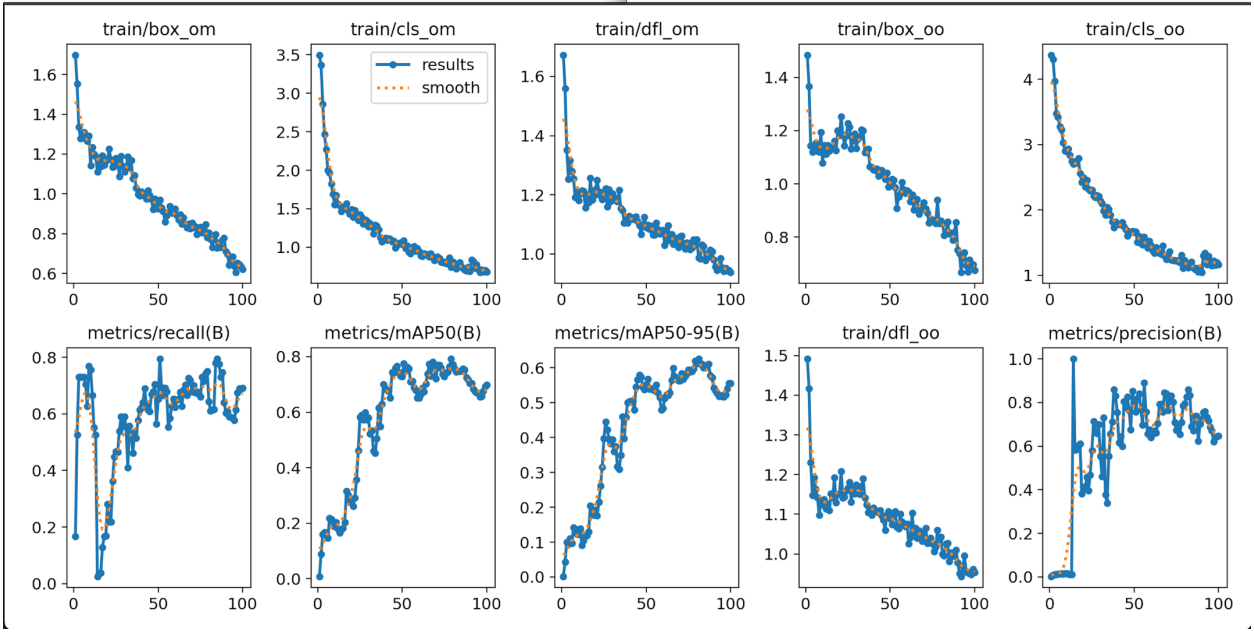




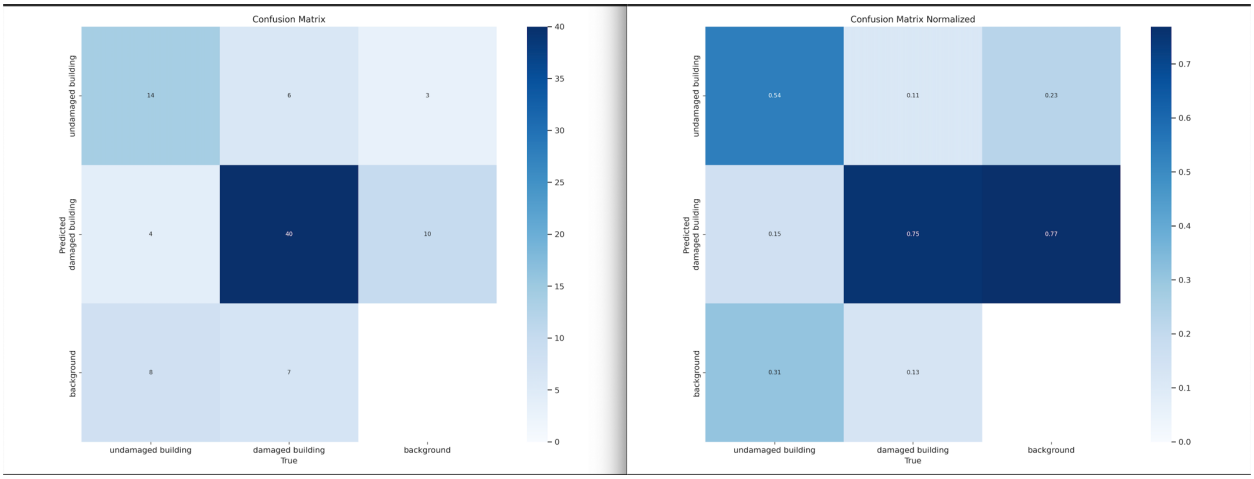
Yolov10_bs16_non_extended:

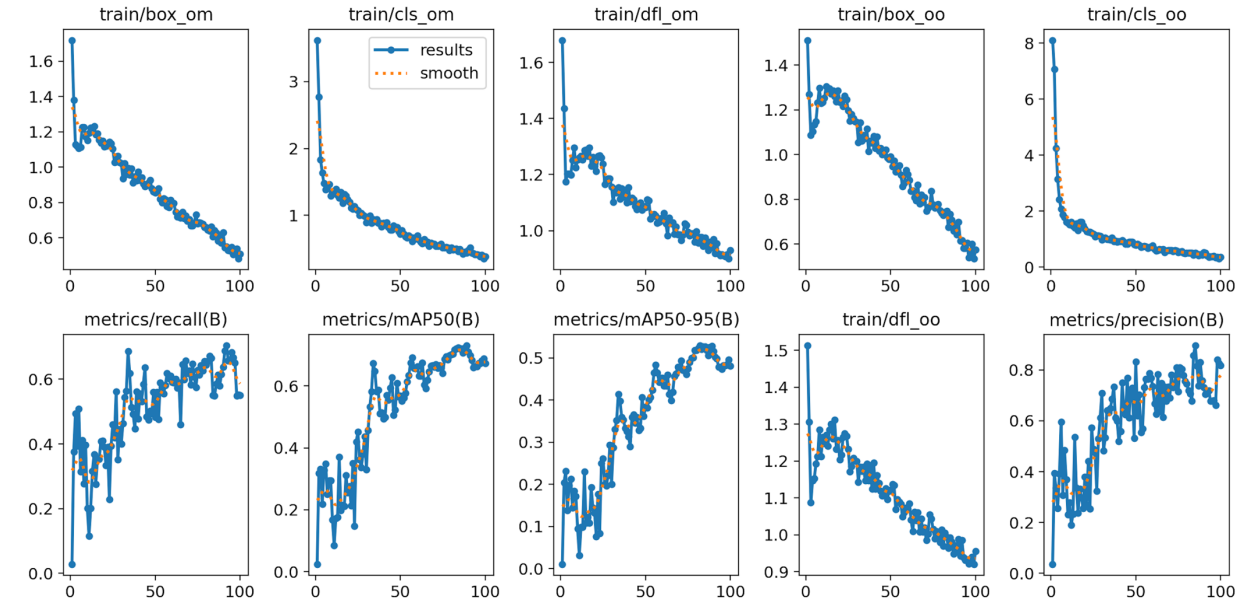




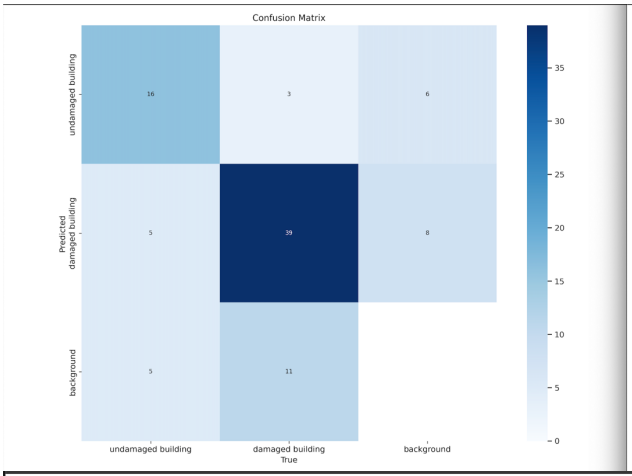


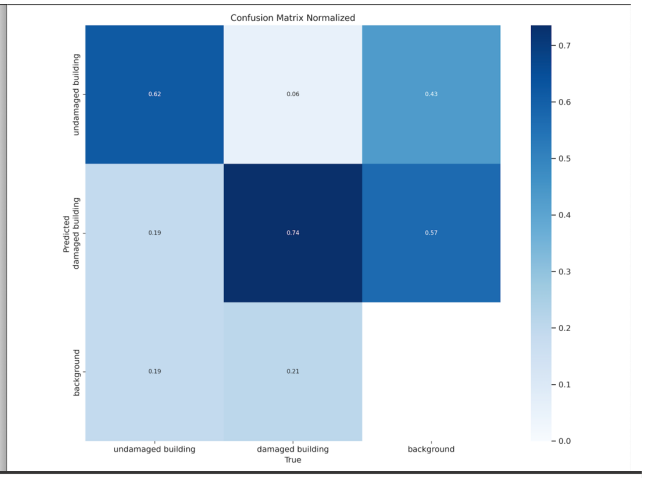
yolov10s_bs_16_extended:

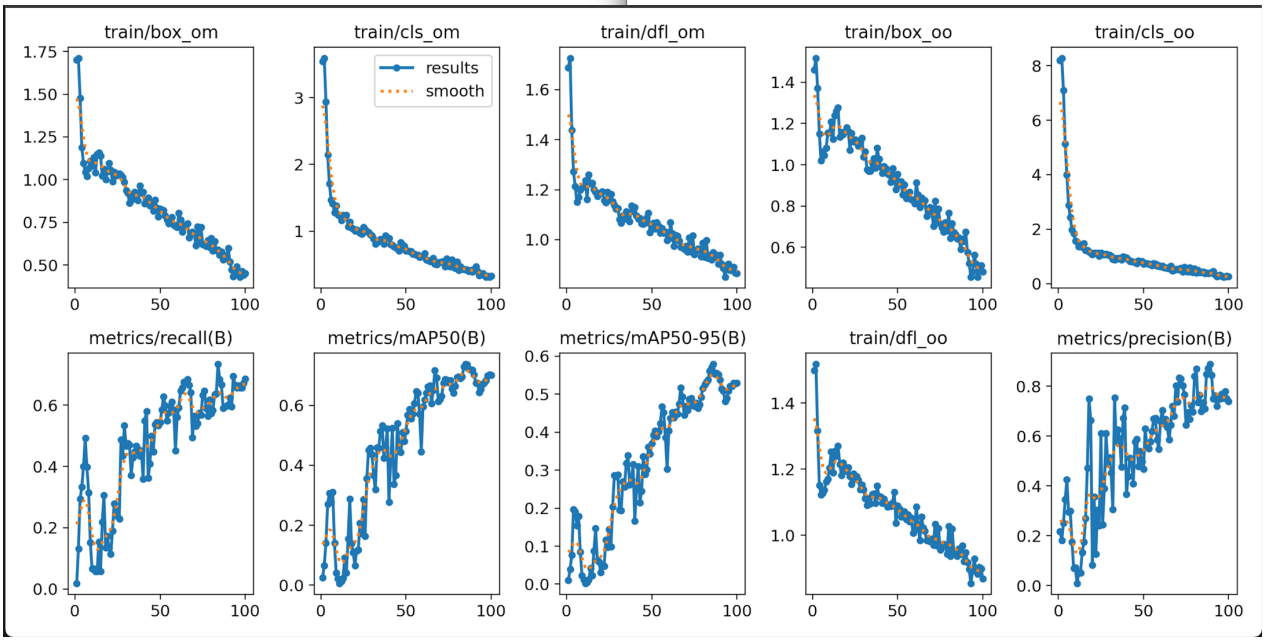




yolov10s_bs_32_extended:







Model		*Precision*	*Recall*	*F1 Score*	*mAP@0.5*	*mAP@0.5:0.95*
yolov5m_bs16_extended		0.584	0.00962	0.019	0.00244	0.00122
yolov5s_incorrect_bbox		-	-		-	-
yolov8n_bs16_annotation_fail	all	0.377	0.262	0,309	0.252	0.178
	undamaged building	0.42	0.381	0.399	0.342	0.259
	damaged building	0.334	0.143	0,200	0.162	0.096
yolov8n_fixed_annotation	all	0.727	0.552	0,628	0.605	0.333
	undamaged building	0.851	0.442	0,582	0.548	0.266
	damaged building	0.603	0.663	0,632	0.663	0.401
yolov8s_bs64_extended	all	0.795	0.42	0,550	0.608	0.361
	undamaged building	0.908	0.192	0,317	0.508	0.298
	damaged building	0.682	0.647	0,664	0.709	0.425
yolov8s_bs32_extended	all	0.69	0.522	0,594	0.598	0.365
	undamaged building	0.755	0.385	0,510	0.471	0.304
	damaged building	0.626	0.66	0,643	0.726	0.426
yolov10_bs16_non_extended	all	0.823	0.611	0,701	0.756	0.627
	undamaged building	0.848	0.615	0,713	0.692	0.613
	damaged building	0.797	0.606	0,688	0.82	0.641
yolov10s_bs_16_extended	all	0.77	0.606	0,678	0.733	0.529
	undamaged building	0.695	0.439	0,538	0.608	0.482
	damaged building	0.845	0.774	0,808	0.858	0.576
yolov10s_bs_32_extended	all	0.71	0.669	0,689	0.736	0.579
	undamaged building	0.658	0.615	0,636	0.669	0.554
	damaged building	0.761	0.723	0,742	0.802	0.603

Model		Precision	Recall	mAP@0.5	mAP@0.5:0.95
yolov5m_bs16_extended		0.584	0.00962	0.00244	0.00122
yolov5s_incorrect_bbox		-	-	-	-
yolov8n_bs16_annotation_fail	all	0.377	0.262	0.252	0.178
	Undamaged-building	0.42	0.381	0.342	0.259
	Damaged buildinag	0.334	0.143	0.162	0.096
yolov8n_fixed_annotation	all	0.727	0.552	0.605	0.333
	undamaged building	0.851	0.442	0.548	0.266
	damaged building	0.603	0.663	0.663	0.401
yolov8s_bs64_extended	all	0.795	0.42	0.608	0.361
	undamaged building	0.908	0.192	0.508	0.298
	damaged building	0.682	0.647	0.709	0.425
yolov8s_bs32_extended	all	0.69	0.522	0.598	0.365
	undamaged building	0.755	0.385	0.471	0.304
	damaged building	0.626	0.66	0.726	0.426
yolov10_bs16_non_extended	all	0.823	0.611	0.756	0.627
	undamaged building	0.848	0.615	0.692	0.613
	damaged building	0.797	0.606	0.82	0.641
yolov10s_bs_16_extended	all	0.77	0.606	0.733	0.529
	undamaged building	0.695	0.439	0.608	0.482
	damaged building	0.845	0.774	0.858	0.576
yolov10s_bs_32_extended	all	0.71	0.669	0.736	0.579
	undamaged building	0.658	0.615	0.669	0.554
	damaged building	0.761	0.723	0.802	0.603

INFERENCES FROM YOLOV10S_BS16_EXTENDED DATASET:

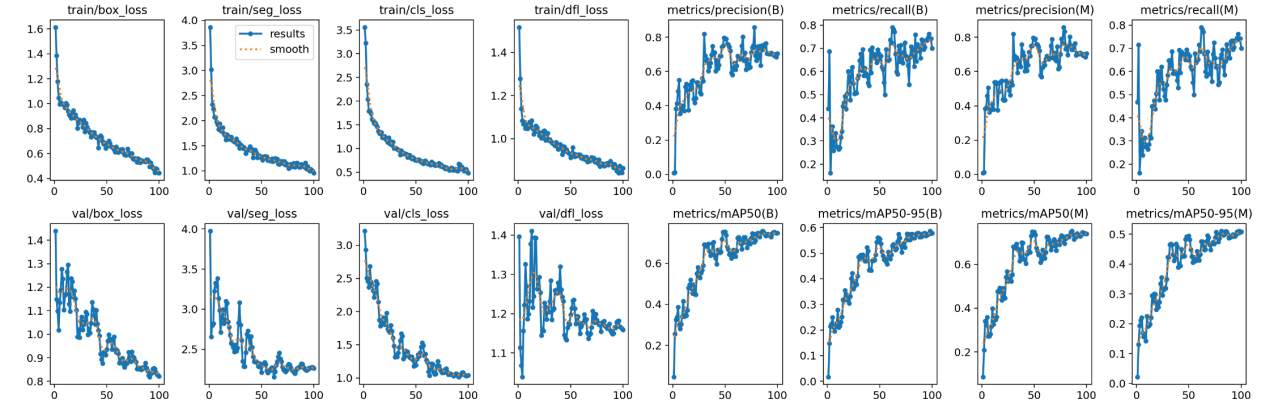
24.07- Researches on Worker dataset
Visdrone dataset has been processed. And so on.. this part will be written on 25.07
Documentation continues at other project

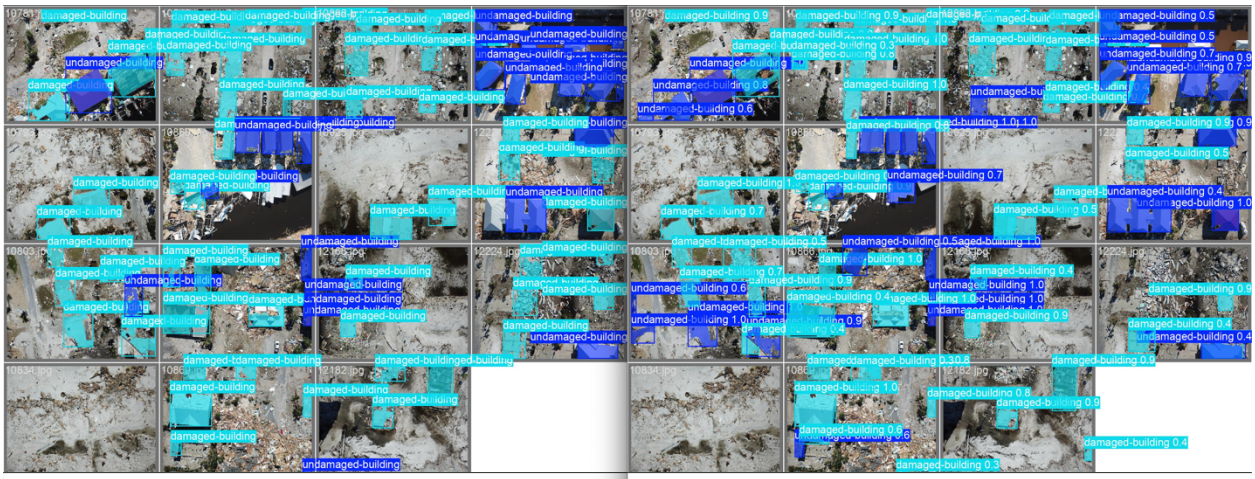
26.07- Segmentation Tests

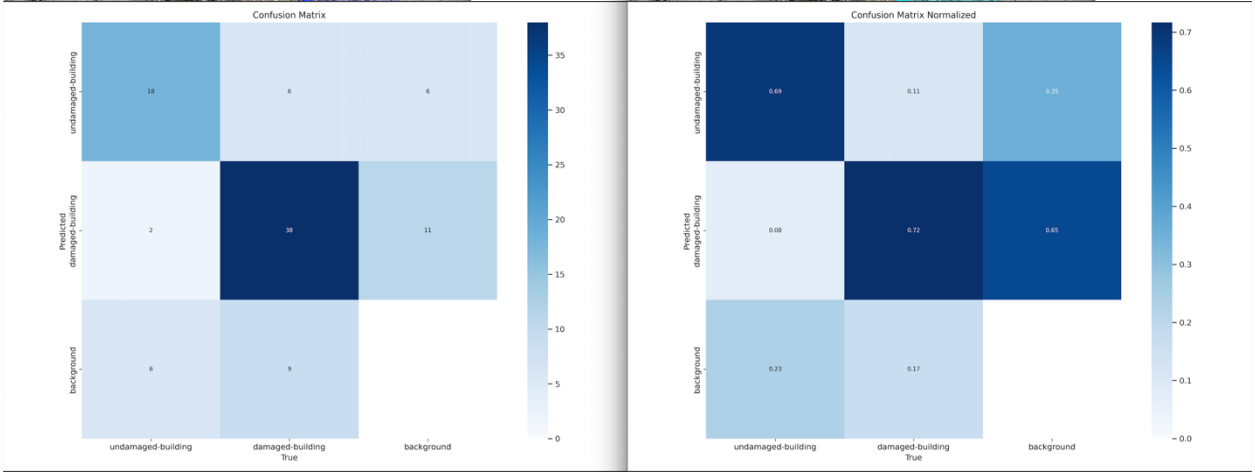
Creating new labels for segmentation using yolo_bbox2segment method
Saving new labels to drive
Utilized yolov8n-seg.pt pretrained model on Google Colab notebook
Model trained with custom dataset with Transfer learning on 100 epochs and 16 batch size

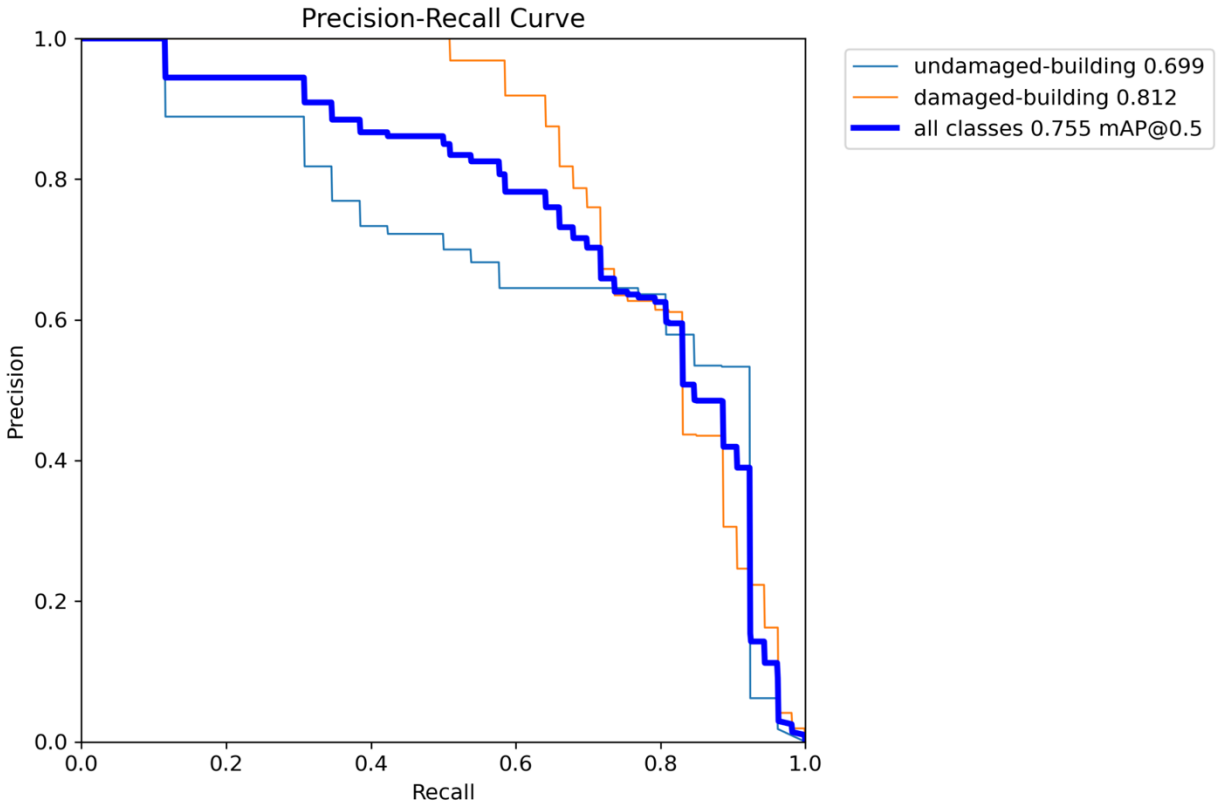
```
!yolo task=segment mode=train model=yolov8n-seg.pt
data=/content/dataset.yaml epochs=100 imgsz=640 cache=True
```

```
Validating runs/segment/train2/weights/best.pt...
Ultralytics YOLOv8.2.66 🚀 Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n-seg summary (fused): 195 layers, 3,258,454 parameters, 0 gradients, 12.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.95it/s]
                   all         15         79        0.7      0.739      0.755      0.583        0.7      0.739      0.743      0.512
    undamaged-building          8         26      0.643      0.761      0.698       0.59      0.643      0.761      0.698      0.531
      damaged-building         13         53      0.756      0.717      0.811      0.577      0.756      0.717      0.788      0.492
Speed: 0.2ms preprocess, 2.8ms inference, 0.0ms loss, 2.4ms postprocess per image
```

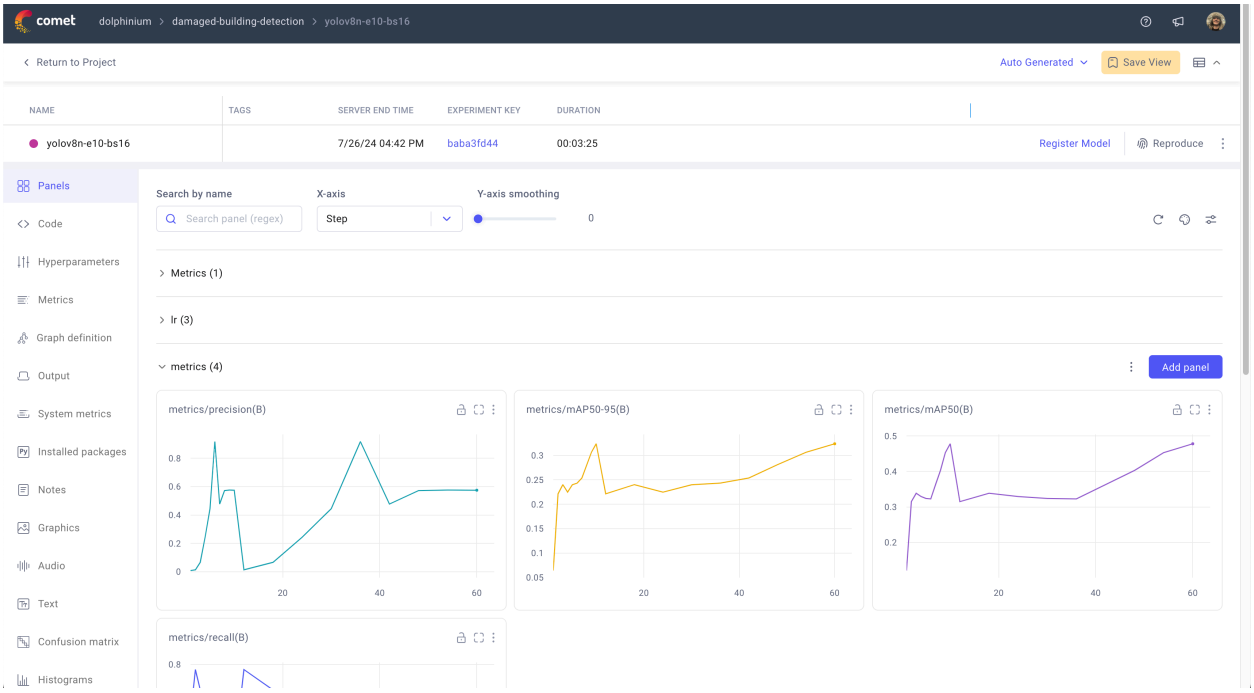








Comet-ML has been utilized for tracking YOLOv8 experiments.
Rest of the work will be tracked at comet-ml



31.07- GEOREFERENCING AND MAPPING

```
todos:
* automate digging params from image metadata
* do it for all test images
* build and interface that you can upload a image or images and extract georeferences
```

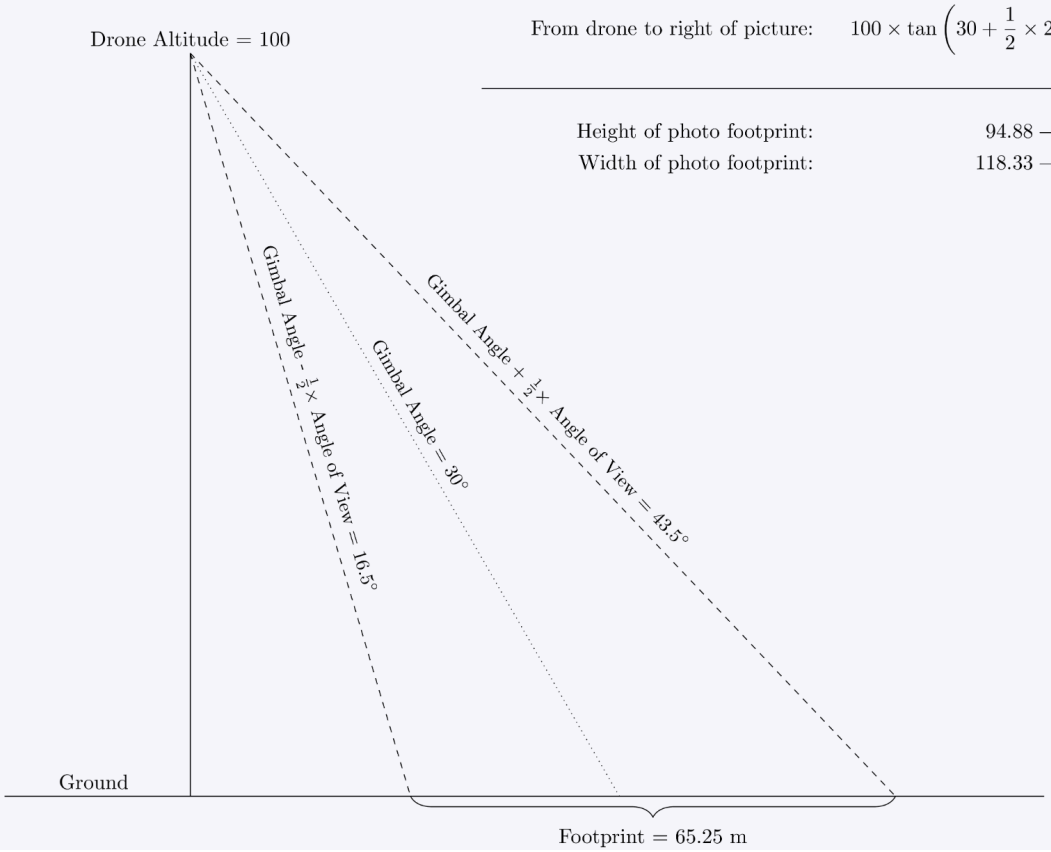
JOBS DONE:
* Digging the important metadata from jpg img.
* Beside that information building a georeferencing structure and and obtaining lat, lon values for detections.
* Visualize this points on map.
* Gradio implementation done for upload and inference, see: "gradio-test.py".
* Gradio implementation with map is on the way.
See: "gradio_with_map.py".

The Photographic Footprint of a Camera on a Drone

variables		
xsensor	36	width of sensor in mm
ysensor	24	height of sensor in mm
focallen	50	focal length of lens in mm
altitude	100	height in m
xgimbal	30	x-axis gimbal angle
ygimbal	30	y-axis gimbal angle

Field of view wide:	$2\tan^{-1}\left(\frac{36}{2\times 50}\right) = 39.6°$
Field of view tall:	$2\tan^{-1}\left(\frac{24}{2\times 50}\right) = 26.99°$
From drone to bottom of picture:	$100\times\tan\left(30-\frac{1}{2}\times 39.6\right) = 17.99m$
From drone to top of picture:	$100\times\tan\left(30+\frac{1}{2}\times 39.6\right) = 118.33m$
From drone to left of picture:	$100\times\tan\left(30-\frac{1}{2}\times 26.99\right) = 29.63m$
From drone to right of picture:	$100\times\tan\left(30+\frac{1}{2}\times 26.99\right) = 94.88m$

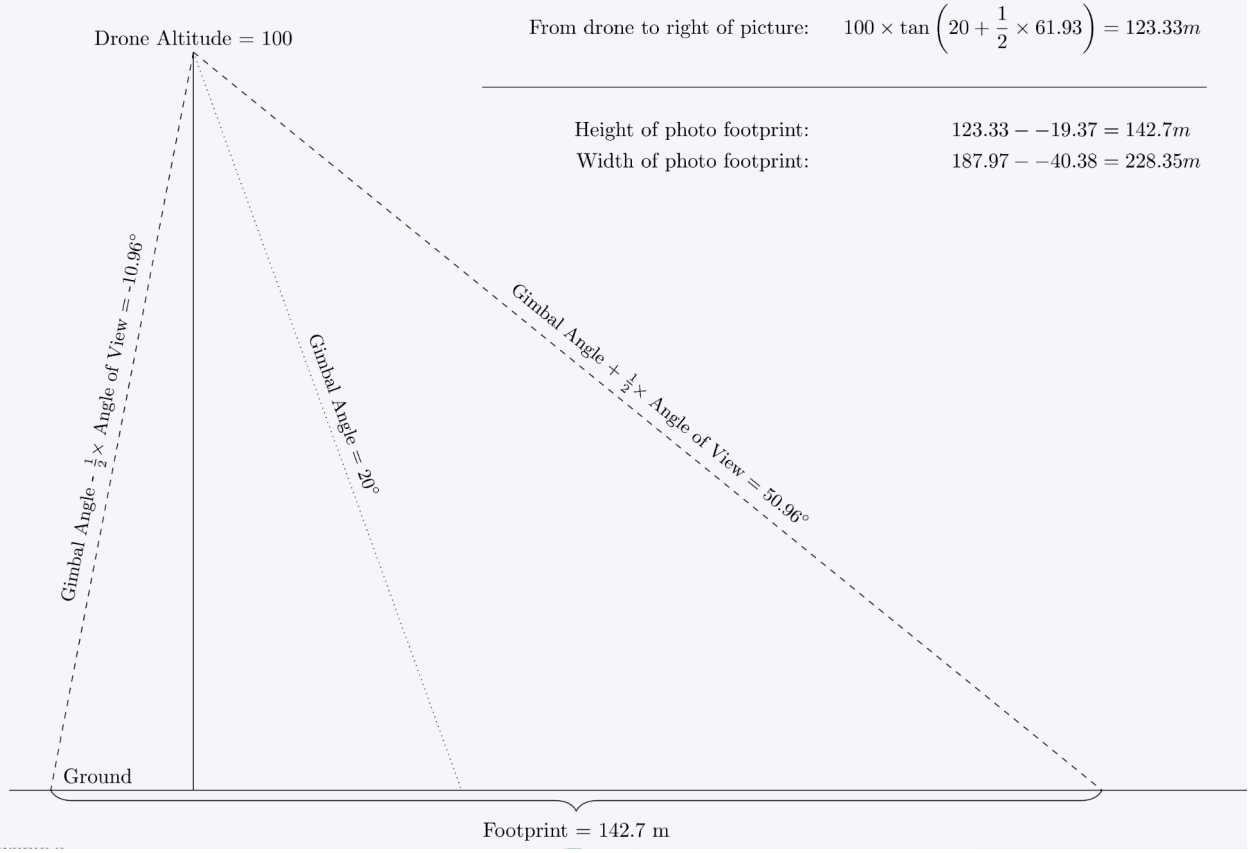
Height of photo footprint:	$94.88 - 29.63 = 65.25m$
Width of photo footprint:	$118.33 - 17.99 = 100.33m$



variables		
xsensor	36	width of sensor in mm
ysensor	24	height of sensor in mm
focallen	20	focal length of lens in mm
altitude	100	height in m
xgimbal	20	x-axis gimbal angle
ygimbal	20	y-axis gimbal angle

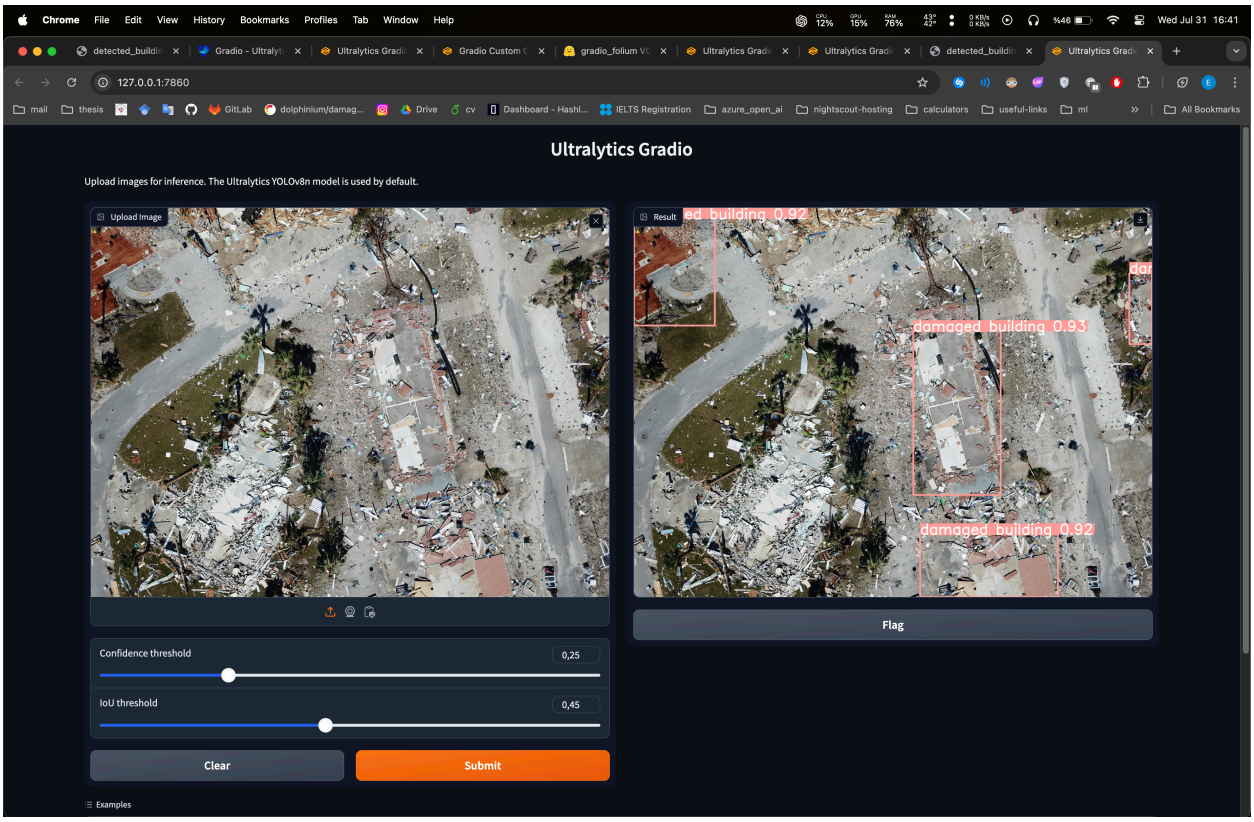
Field of view wide:	$2\tan^{-1}\left(\frac{36}{2\times 20}\right) = 83.97°$
Field of view tall:	$2\tan^{-1}\left(\frac{24}{2\times 20}\right) = 61.93°$
From drone to bottom of picture:	$100\times\tan\left(20-\frac{1}{2}\times 83.97\right) = -40.38m$
From drone to top of picture:	$100\times\tan\left(20+\frac{1}{2}\times 83.97\right) = 187.97m$
From drone to left of picture:	$100\times\tan\left(20-\frac{1}{2}\times 61.93\right) = -19.37m$
From drone to right of picture:	$100\times\tan\left(20+\frac{1}{2}\times 61.93\right) = 123.33m$

Height of photo footprint:	$123.33 - -19.37 = 142.7m$
Width of photo footprint:	$187.97 - -40.38 = 228.35m$



See "detected_buildings_sattelite.html" and "dig_metadata.py" on GitHub.

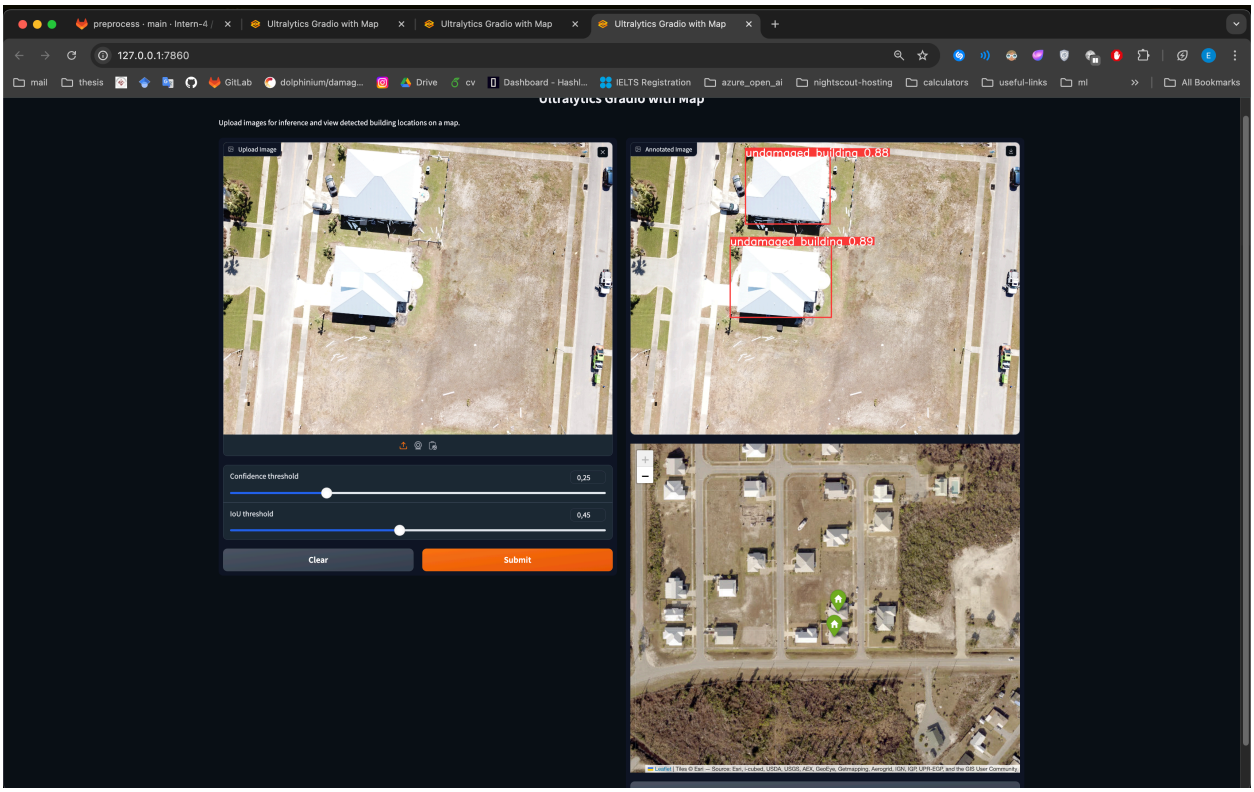
Upload and inference mechanism:



01.08 Full Dataset Run Trials on Edward's Machine

Preparing:
1- clean images and labels for training:
preparing a python script for extracting labels on dataset
2- cometml background.
experiments on fly
3- dealing with ntfs format
on macOS file format is different with Windows arch. Paragon NTFS has been installed for fixing this problem and file priviliges have been set on Recovery Mode(Its like BIOS settings in Windows)
4- Adding the show on map according to detected buildings' location functionality to Web Application using Folium Framework. There is a problem about image metadata on this stage. This part will be continue at next day.
5- Data has been moved to Edward's machine

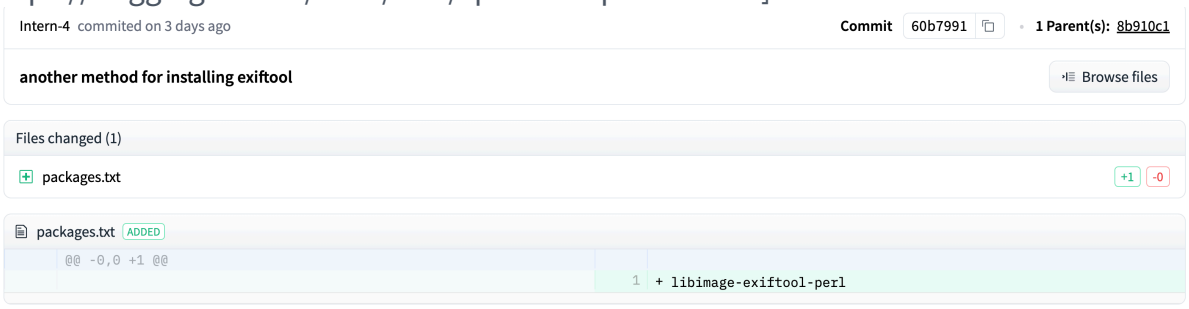
Fixing the gradio-with-map.py



02.08- Hosting the model and Gradio web page on HuggingFace Spaces platform and Full Dataset Training

Hosting Part:

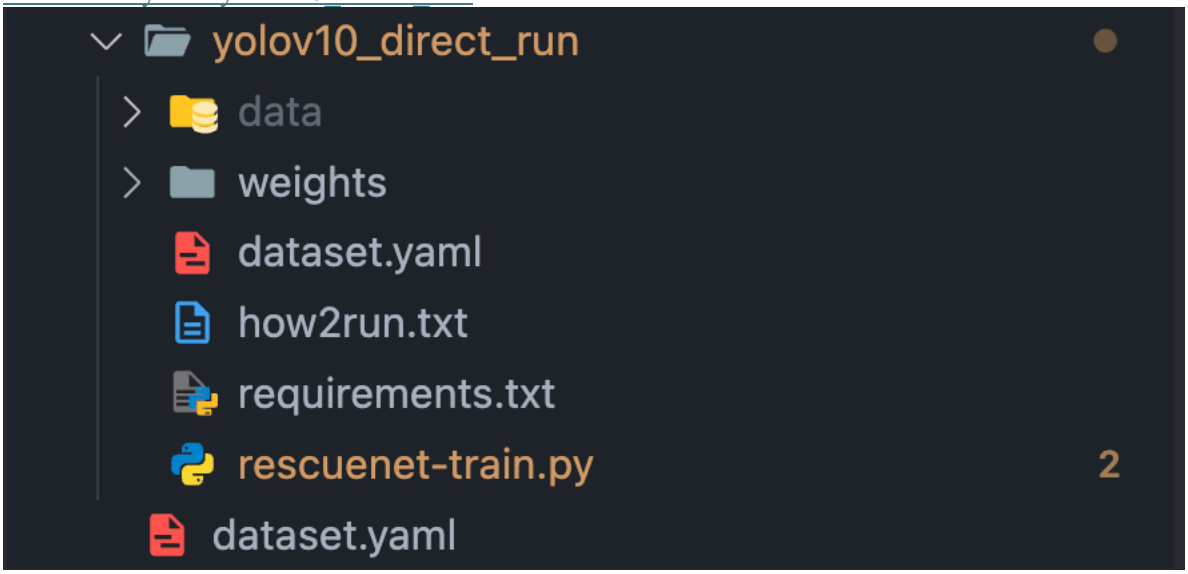
HuggingFace Spaces: https://huggingface.co/spaces/launch
HuggingFace Spaces basically provides hosting option to developers for their Machine Learning Projects for free. It has also high integration capabilities with Python and Gradio so it is a perfect place to be :)
You can find my final python script for hosting at "dbd-host" repository on GitLab(Damaged building detection): https://gitlab.bewelltech.com.tr/Intern4/dbd-host
Website is live at URL: https://huggingface.co/spaces/dolphinium/rescuenet-damaged-building-detection
Problems at hosting:
* path problem fix: relative path to root directory for weights
* requirements.txt handling:
HFSpaces automatically install requirements based on you requirements.txt file. There was a issue about dependencies and resolved.
* ExifTool Installation:
Besides dependencies on requirements.txt ExifTool which I use for "digging metadata from images" should be installed as an application on hosting platform which is a Linux Distro. I specified it on packages.txt file.
Debian dependencies are also supported. Add a **packages.txt** file at the root of your repository, and list all your dependencies in it. Each dependency should be on a separate line, and each line will be read and installed by `apt-get install`.
[https://huggingface.co/docs/hub/spaces-dependencies]



exif-tool has been successfully installed after that.
* Digging the metadata:
XMP data on the image can not be gathering on hosted environment. I tried several ways but it didnt resolve the issue. The issue may be caused from image uploading part. Some of the metadata from image can be corrupted on upload process. This issue will be handling on next days.

Model Training Part on Edward's Machine:

repo dir url: https://gitlab.bewelltech.com.tr/Intern4/afad-object-detection/-/tree/main/yolos/yolov10_direct_run



rescuenet-train.py:

```python
import comet_ml
from ultralytics import YOLOv10


# login to cometml to save the run on cloud
comet_ml.login(project_name="damaged-building-detection")

# https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10s.pt
# download from url and place it under weights directory:

model = YOLOv10("weights/yolov10s.pt")

results = model.train(
    data="dataset.yaml",
    project="damaged-building-detection",
    batch=16, # try different batch sizes for gpu utility?
    save_json=True,
    epochs=30, # increase epochs to 100 for better training
)
```

how2run.txt
1: create a conda env
conda create -n yolov10 python=3.9
conda activate yolov10
pip install -r requirements.txt
pip install -e .
pip install cometml
pip install -q supervision roboflow "comet_ml>=3.44.0"

export cometml api key
2: export COMET_API_KEY=6hQ0**********

3: download weights and place under weights directory on root:
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10n.pt
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10s.pt
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10m.pt
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10b.pt
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10x.pt
https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10l.pt

4: set data path variable on dataset.yaml

5: python rescuenet-train.py

05.08- Analyzing Run Results and Dockerizing the app

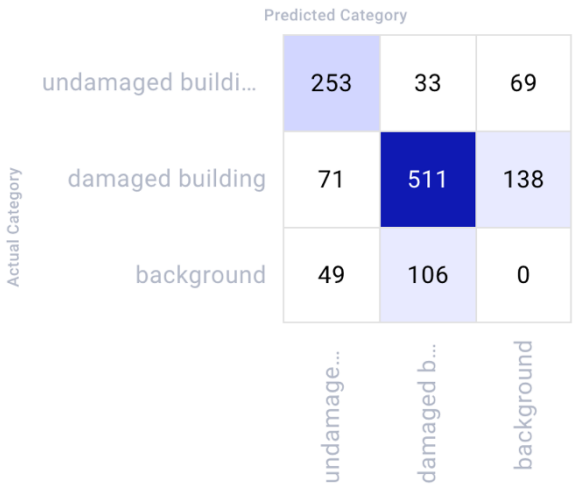
Runs:
yolov10s-e30-b16:

```
Validating damaged-building-detection/train7/weights/best.pt...
Ultralytics YOLOv8.1.34 🚀 Python-3.9.19 torch-2.0.1+cu117 CUDA:0 (NVIDIA GeForce RTX 4070, 12001MiB)
YOLOv10s summary (fused): 293 layers, 8036508 parameters, 0 gradients, 24.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 15/15 [00:02<00:00,  5.63it/s]
                   all        449       1023      0.739       0.74       0.79      0.642
    undamaged building        449        373      0.761      0.705      0.793      0.682
      damaged building        449        650      0.718      0.775      0.787      0.603
Speed: 0.4ms preprocess, 3.0ms inference, 0.0ms loss, 0.0ms postprocess per image
```

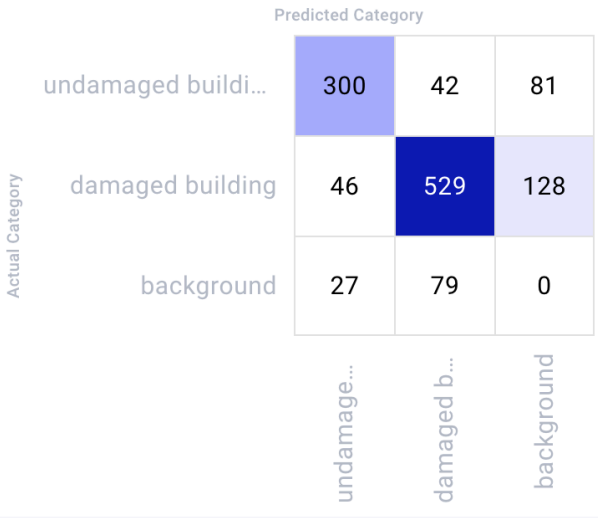
yolov10m-e100-b16:

```
Validating damaged-building-detection/train9/weights/best.pt...
Ultralytics YOLOv8.1.34 🚀 Python-3.9.19 torch-2.0.1+cu117 CUDA:0 (NVIDIA GeForce RTX 4070, 12001MiB)
YOLOv10m summary (fused): 369 layers, 16452700 parameters, 0 gradients, 63.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 15/15 [00:04<00:00,  3.53it/s]
                   all        449       1023      0.809      0.777      0.829      0.684
    undamaged building        449        373      0.803      0.787      0.829      0.717
      damaged building        449        650      0.814      0.768      0.829      0.651
Speed: 0.5ms preprocess, 5.9ms inference, 0.0ms loss, 0.1ms postprocess per image
```

S:30:16



M:100:16

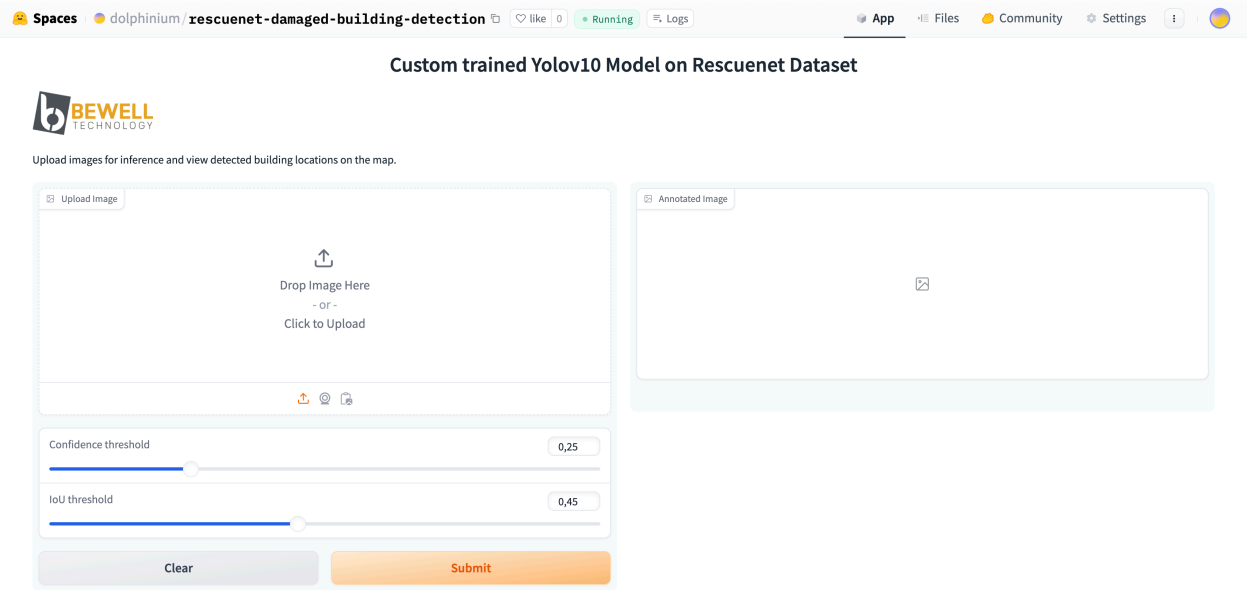


Dockerizing the app

```
# Use an official Python runtime as a parent image
FROM python:3.9.19-slim

# Install system dependencies (if any, from packages.txt)
COPY packages.txt .
RUN apt-get update && \
    apt-get install -y git && \
    xargs -a packages.txt apt-get install -y && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Expose port the app runs on
EXPOSE 7860

# Run the application
CMD ["python", "gradio_with_map.py"]
```

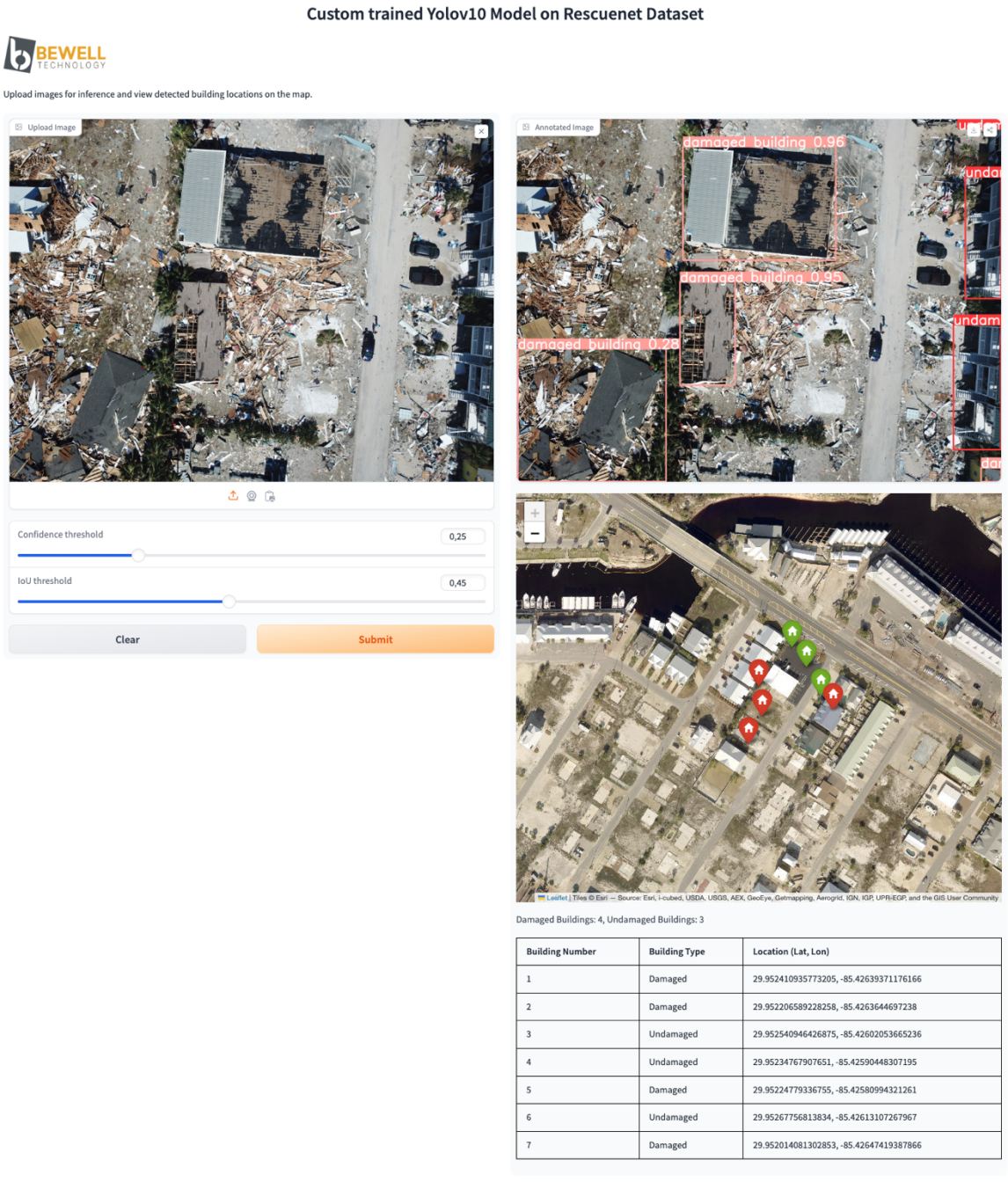
Fixing the xmp data digging problem:

On HuggingFace Spaces updating the model weights to "yolov10m-e100-b16-full-best.pt" and adding the Bewell Logo



06.08- Configuring the spaces and Fixing the Precise Location Problem

Adding summary part and table to website:



Building Number	Building Type	Location (Lat, Lon)
1	Damaged	29.952410935773205, -85.42639371176166
2	Damaged	29.952206589228258, -85.4263644697238
3	Undamaged	29.952540946426875, -85.42602053665236
4	Undamaged	29.95234767907651, -85.42590448307195
5	Damaged	29.95224779336755, -85.42580994321261
6	Undamaged	29.95267756813834, -85.42613107267967
7	Damaged	29.952014081302853, -85.42647419387866