SCIENCE INTERNSHIP PROGRAM

AAI-06

Deep Learning for aerial vision and language navigation

Vision and Language Navigation

Sources

Object Detection

Acknowledgements

Semantic Segmentation

Introduction

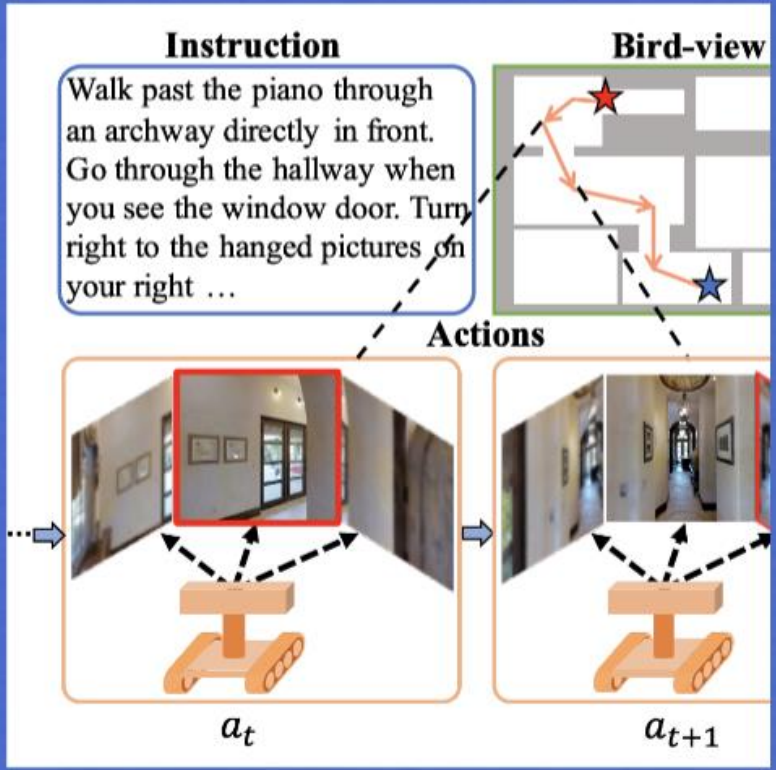
Members

Mentor: Yue Fan
Faculty advisor: Prof. Xin Wang

Interns:
- Bhavani Venkatesan - International School of Hyderabad
- Shaashvat Shetty - Pacific Collegiate School
- Pranav Joshi - Fremont High School

Our Goal

Controlling a drone using Visual Language Navigation (VLN)



Purpose

Requirements

Purpose

Creating a VLN-enabled drone will allow users to control it more easily .

It may also understand the environment (ex. number of cars, and people) better than a human.

Requirements

In order for the VLN code to work, the model needs to accomplish three tasks:

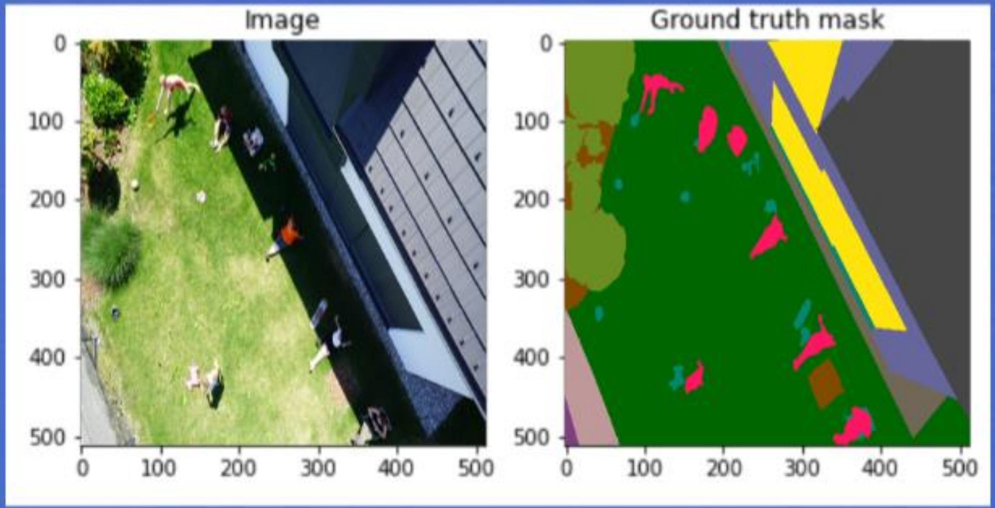
1) Semantic Segmentation

2) Object Detection

3) Understand and respond to language instructions

Semantic Segmentation

- Objects of the same class are assigned the same color.
- Trained by giving the model a ground truth mask.
- Produces its own prediction mask.



Datasets

Single-class segmentation

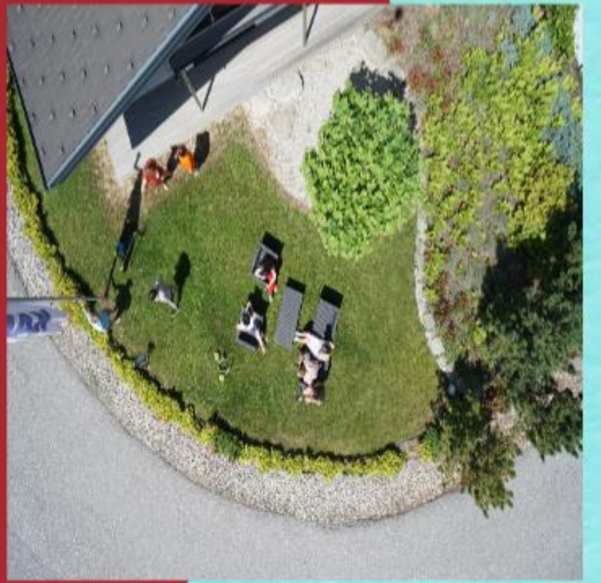
Results

Model Accuracy

Data used for semantic segmentation

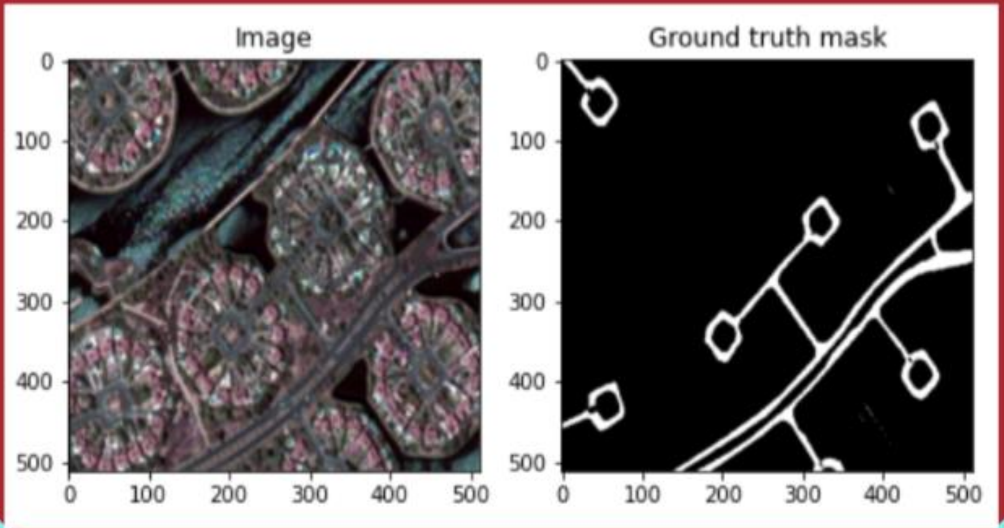
- We used three different datasets.
- Each dataset has images at different distances from the ground.

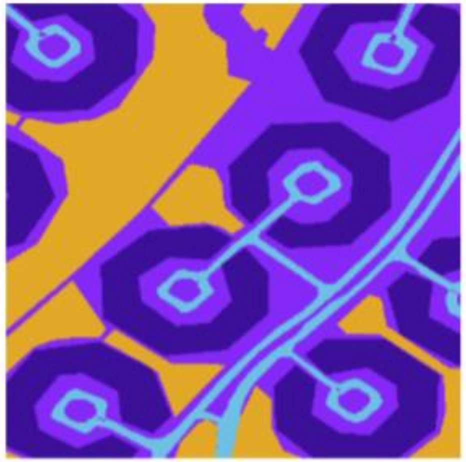




Single class segmentation

- Seperate segmentation model for each class.
- Objects of the specified class are white and the background is black.
- Easier to train the model since color coding of masks is different in each dataset.
- VLN model calls only the required segmentation model.

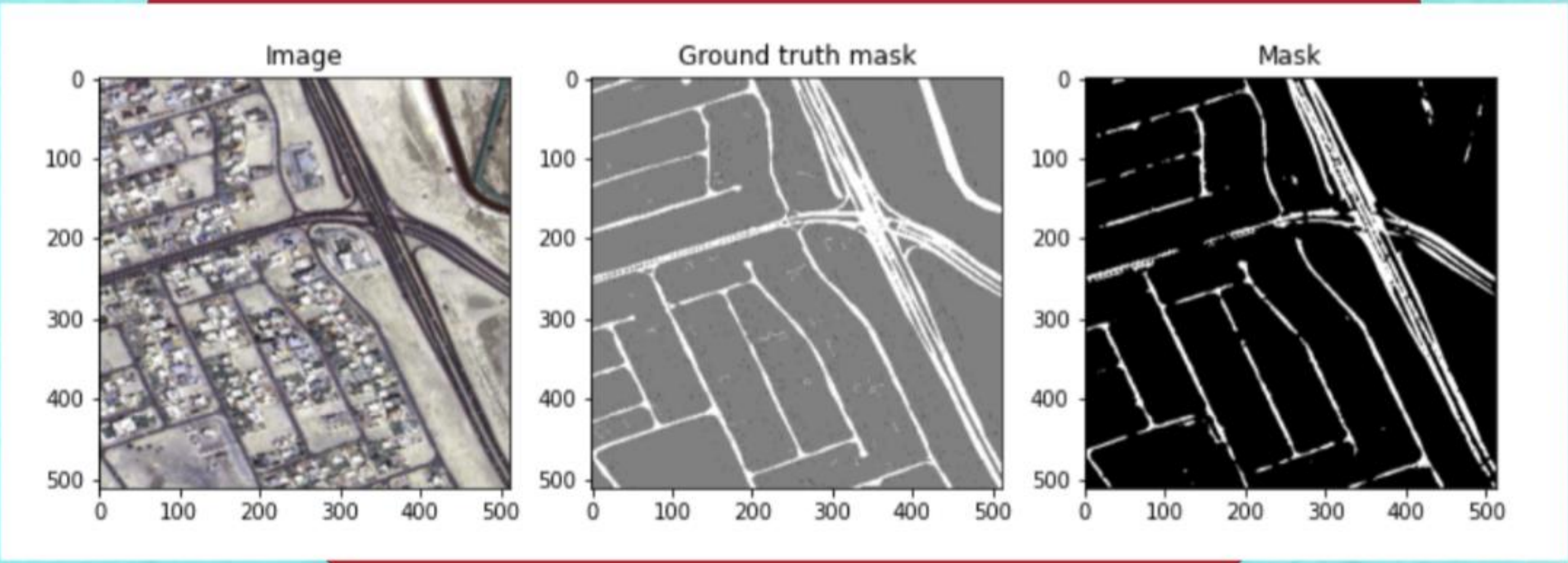




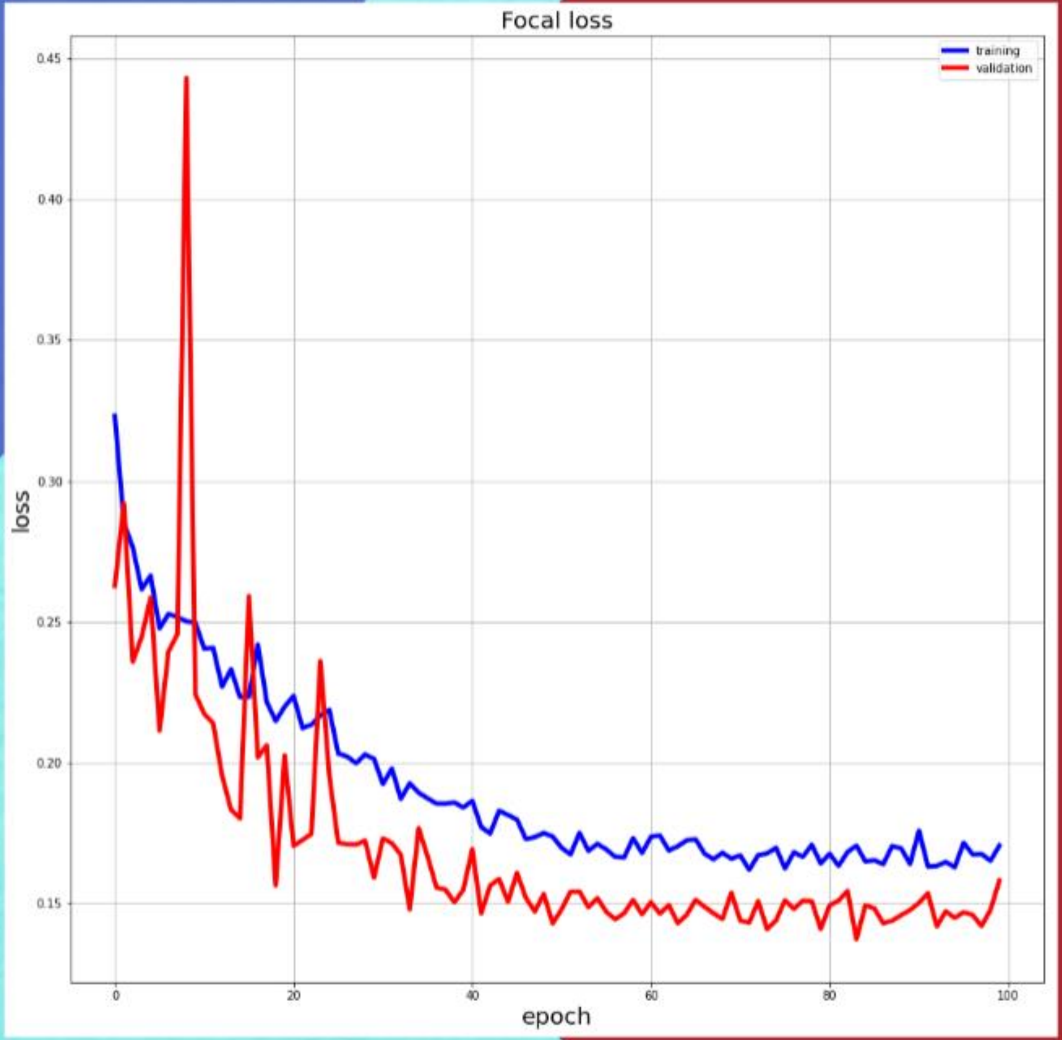
Original mask

Training and Results

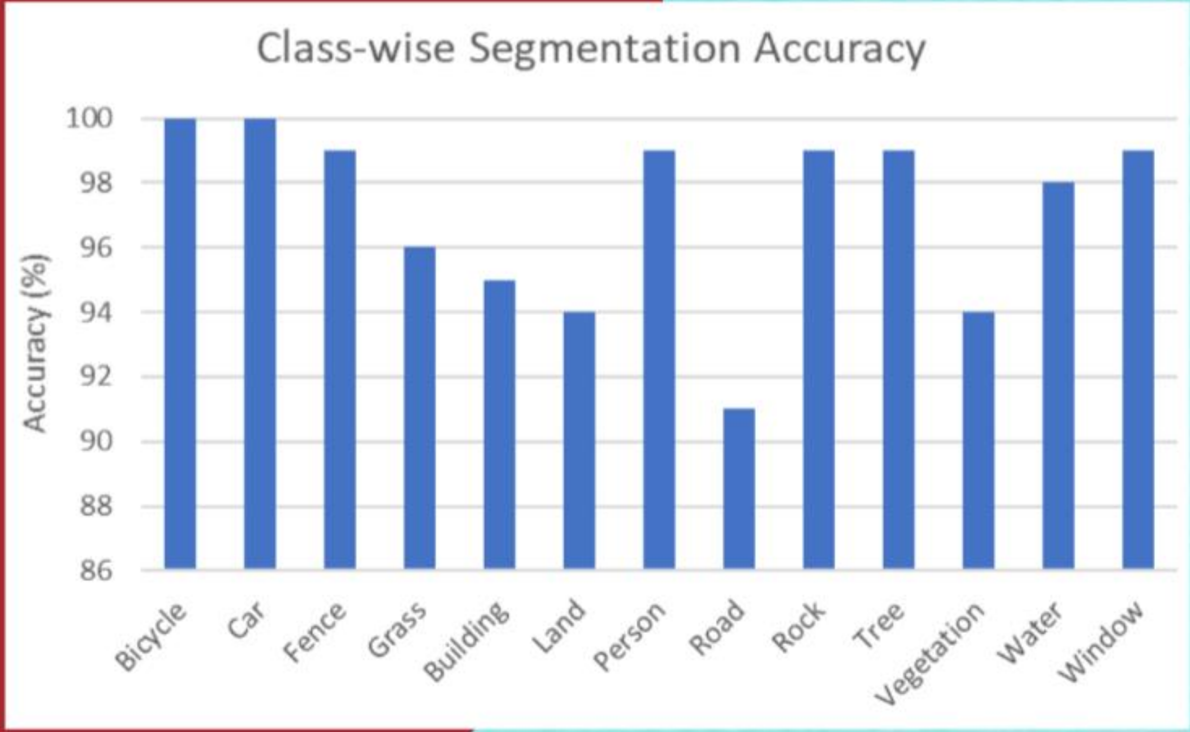
- Combined single-class ground truth masks from the three datasets to train the model.
- Model produces its own mask when given an image.



Model Accuracy



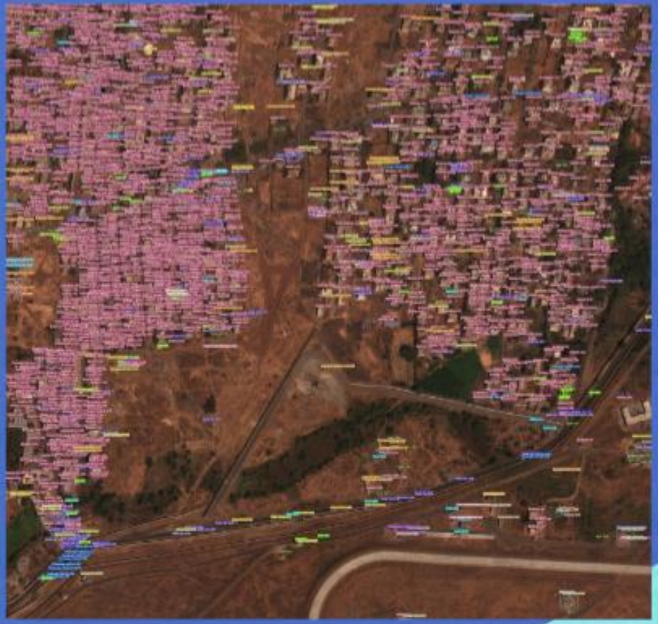
We calculated the loss and accuracy of the model after each iteration of trianing.



Object Detection

In order to detect the number of objects of each class (over 60 classes such as trailers, trucks, buildings etc).

Our team implemented YOLOv3 which is an object-detection algorithm



Dataset

How it works

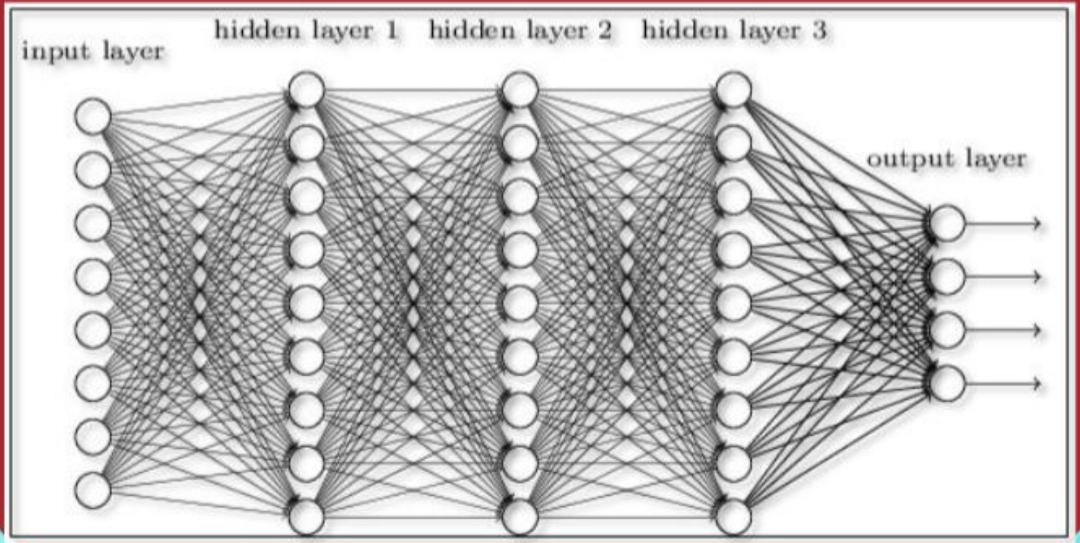
Code

Dataset

The object detection model uses the xview dataset which includes 1127 high resolution aerial images that are split into 60 classes.

How it works

The YOLO machine learning algorithm detects objects using features learned by a deep convolutional neural network.



Example Code

The object detection code can detect how many objects exist in the image.

It was also modified to include word similarity and find the location of a specific object

```
{'passenger vehicle': 5, 'small car': 46, 'bus': 9, 'pickup truck': 8, 'utility truck': 28, 'truck': 55, 'cargo truck': 11, 'truc
k w/box': 10, 'truck tractor': 5, 'trailer': 53, 'truck w/flatbed': 11, 'truck w/liquid': 1, 'crane truck': 1, 'maritime vessel':
 1, 'engineering vehicle': 4, 'mobile crane': 2, 'dump truck': 7, 'haul truck': 9, 'front loader/bulldozer': 17, 'excavator': 4,
'ground grader': 3, 'hut/tent': 5, 'shed': 31, 'building': 92, 'damaged building': 7, 'facility': 2, 'construction site': 1, 'veh
icle lot': 1, 'helipad': 3, 'storage tank': 27, 'shipping container lot': 13, 'shipping container': 9, 'pylon': 24}
command: farthest vehicle
bus = 9  word similarity=0.8235294117647058  cosine similarity=0.0
pickup truck = 8  word similarity=0.780952380952381  cosine similarity=0.0
truck = 55  word similarity=0.8  cosine similarity=0.0
truck w/box = 10  word similarity=0.8  cosine similarity=0.0
truck tractor = 5  word similarity=0.8210526315789474  cosine similarity=0.0
truck w/flatbed = 11  word similarity=0.8  cosine similarity=0.0
truck w/liquid = 1  word similarity=0.8  cosine similarity=0.0
farthest truck w/flatbed at (2562.9414,2038.4631
```

Vision and Language Navigation

- End goal of the project was for us to enter in text instructions
- Code would interpret those instructions and get coordinates
- The drone would then move to those coordinates.

Step 1: Getting Instructions

Step 2: Getting Coordinates

Step 3: Moving the Drone

User Commands

- Instructions entered into the code in one of two formats
  - with regards to position of another object on the map
  - farthest/nearest

```
# format 1: move [obj_rel_dirs] the [numbers] [objects] [drone_rel_dirs]
#           e.g. move to the left of the third car in front of you

# format 2: move to the nearest [objects]
#           e.g.  move to the nearest car
```

Converting Instructions to Coordinates

- Code first identifies which format command is in
- Calculates the coordinates that the drone should move to in order to follow the instructions

Format 1: Segmentation Model
Format 2: Object Detection Model

```
pick an object: pylon
pylon = 24  word similarity=1.0  cosine similarity=1.0
nearest pylon at (839.372,633.1892)
pylon = 24  word similarity=1.0  cosine similarity=1.0
farthest pylon at (2845.511,1940.8402
```

Moving the Drone

- After interpreting the instructions and getting coordinates
- Drone moves in increments until it reaches the destination

Sources

Segmentation datasets:

https://www.kaggle.com/datasets/balraj98/massachusetts-roads-dataset
https://www.kaggle.com/datasets/balraj98/massachusetts-buildings-dataset
https://www.kaggle.com/datasets/bulentsiyah/semantic-drone-dataset
https://www.kaggle.com/datasets/humansintheloop/semantic-segmentation-of-aerial-imagery

Segmentation base code:

https://github.com/amirhosseinh77/UNet-AerialSegmentation

Object detection datasets:

http://xviewdataset.org/

Object detection base code:

https://github.com/ultralytics/xview-yolov3
https://github.com/ShaashvatShetty/Vln_objectDetection

Thanks to our mentor, Yue Fan, for his guidance throughout this project, and for sharing the base code for the drone simulator.

Thank you SIP team for giving us a great summer learning experience!

Any questions?