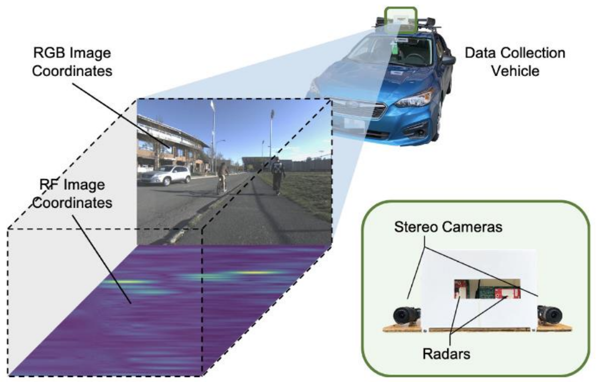
Radar Object Detection

Kushal Gadhiya, John Santos, Irfan Khan

Problem/Use-case

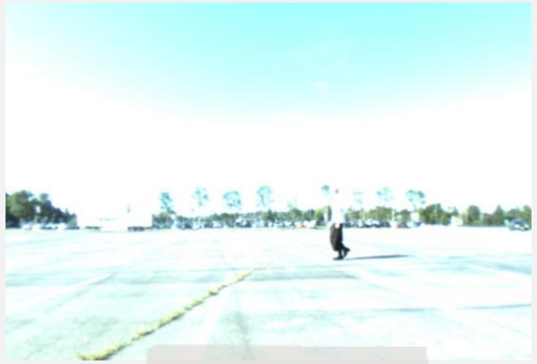
- Assisted driving technology heavily relies on cameras to gather schematic information about the dynamic environment.
- Extreme weather conditions like fog, storms, and low visibility situations can hinder the detection of objects on the road.
- Radar can be more reliant than the camera in low visibility/lighting conditions, as it uses radio frequency for sensing.



(a) Illustration for CRUW dataset collection

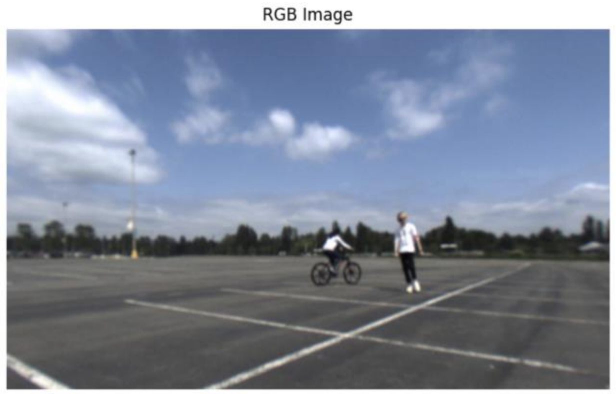
Project Motivations and Goals

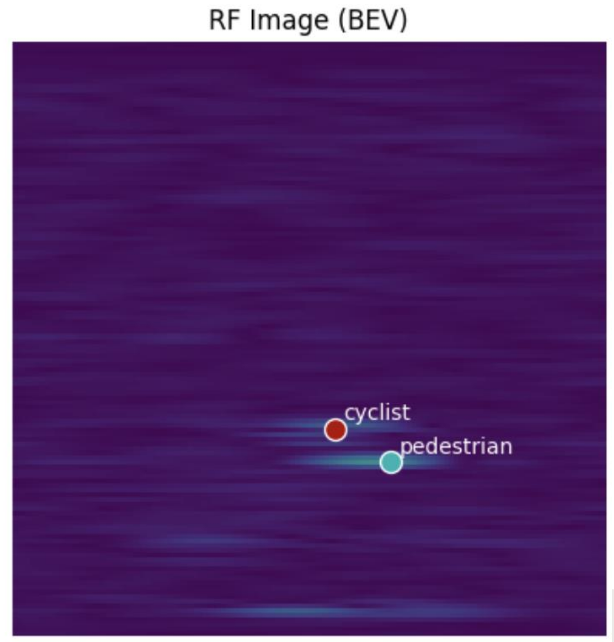
- Compare and analyze the performances of radar-based and camera-based detection models on three levels of difficulty:
    - Easy: clear/high object visibility images.
    - Medium: semi-low visibility images.
    - Hard: objects are highly indistinguishable from the background.
- Explore multi-branching techniques (FastAI) and compression techniques such as pruning and quantization (TinyAI) for potential performance gains and tradeoff analysis.



Radar-Detection Dataset

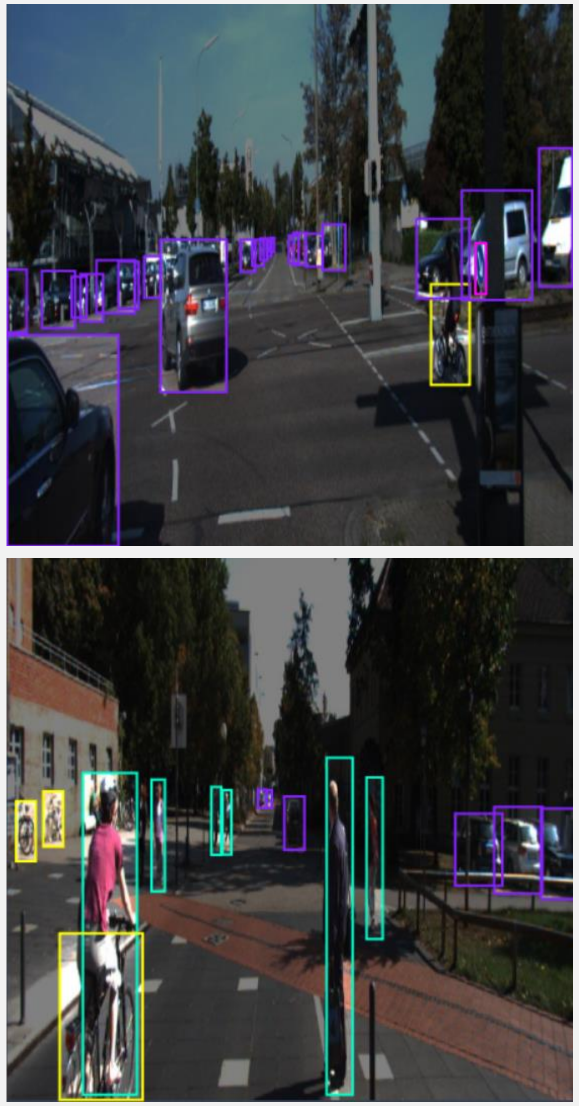
- ROD2021
- Sensors - A pair of stereo cameras and two 77 GHz FMCW radar with FoV of 0-25m and $\pm 60^o$
- Training Set
  - 3 classes of object: Pedestrian, Cyclist, and Car
  - 40 video sequences captured at 30FPS
  - Each sequence contains around 800-1700 frames comprising of an RGB images and 4 NumPy files.
  - A CRF annotation text file indicating the frame ID, range(m), azimuth(rad), and class name of object(s)
- For training the model, we are only using 12 video sequences.



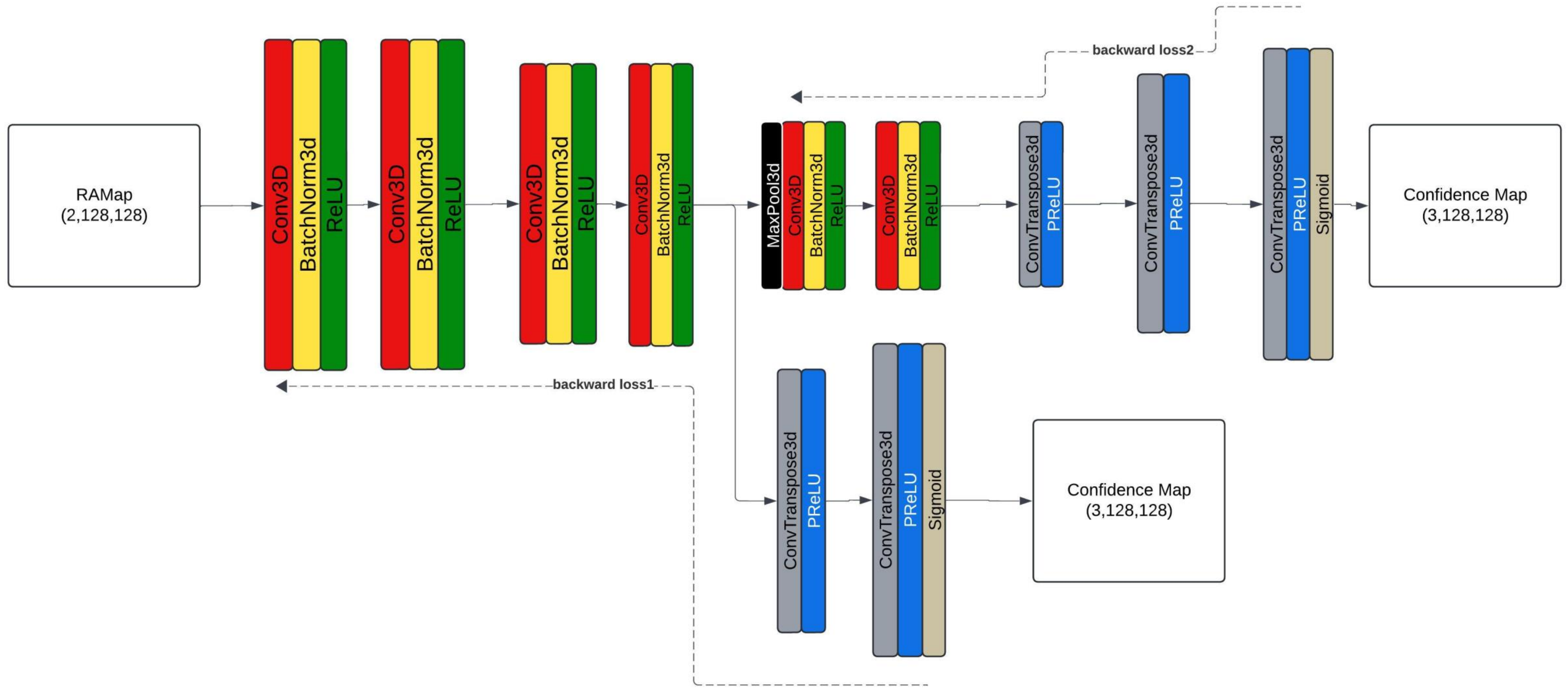


Camera-Detection Dataset

- EDI TY (Roboflow)
- Training Set
  - 5 classes of object: bike, car, cycle, pedestrian, signal
  - Darknet format dataset (image, annotation) pairs.
  - Yolo format annotations (x-center, y-center, width, height) per bounding box.
  - 2801 training images, 802 validation images, 397 test images.
  - Postprocessing: images resized to 640x640 (square resolution).

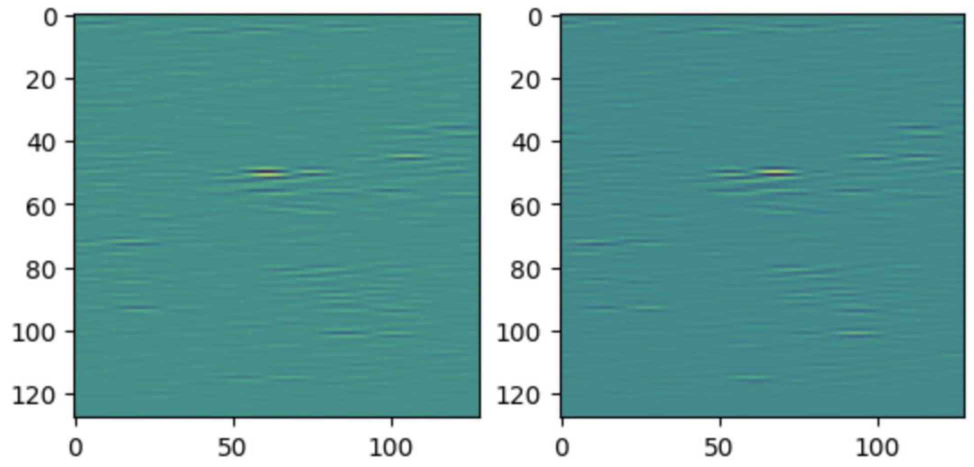


Multi Branching Architecture

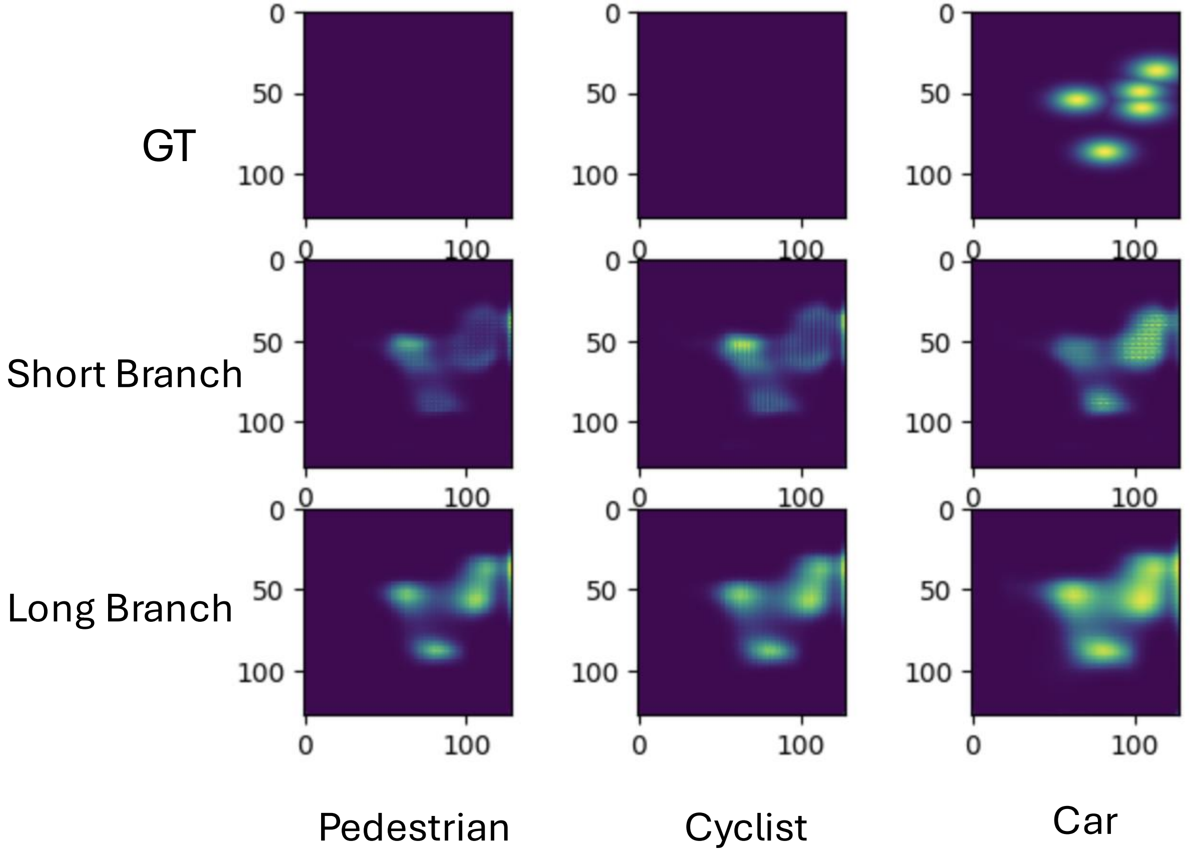


Input and Output Visualized

- RA Heatmaps (Input)



- Confidence Map (Output)



Post-Processing

- Image based detection uses traditional NMS with IoU for removing redundant bounding boxes.
- Object Location Similarity (OLS) metrics is used to determine the distance between two peaks.

$$OLS = \exp\left(\frac{-d^2}{2(s * \kappa_{cls})^2}\right)$$

- OLS threshold value set to 0.3

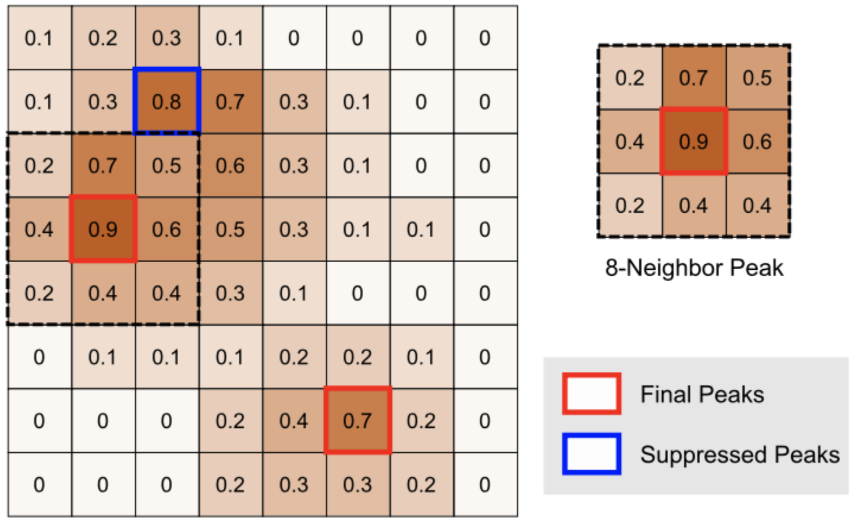
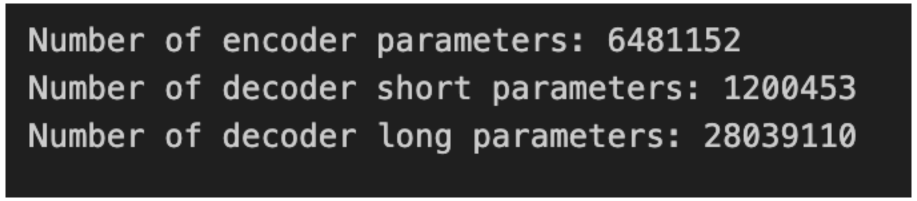


Fig. 4. Example for L-NMS on a ConfMap. The numbers represent the confidence scores predicted by the RODNet. The 8-neighbor peaks are first detected and some peaks are then suppressed if they are nearby some other peaks with higher confidence.

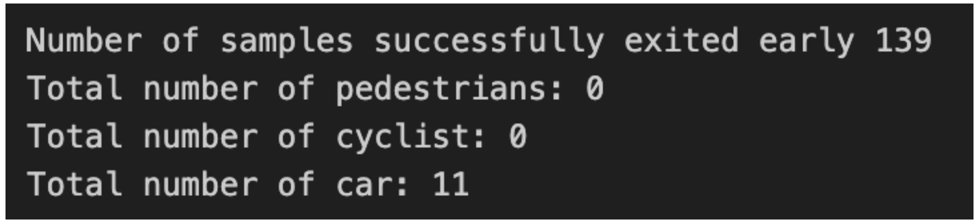
Results

- Total number of trained parameters:

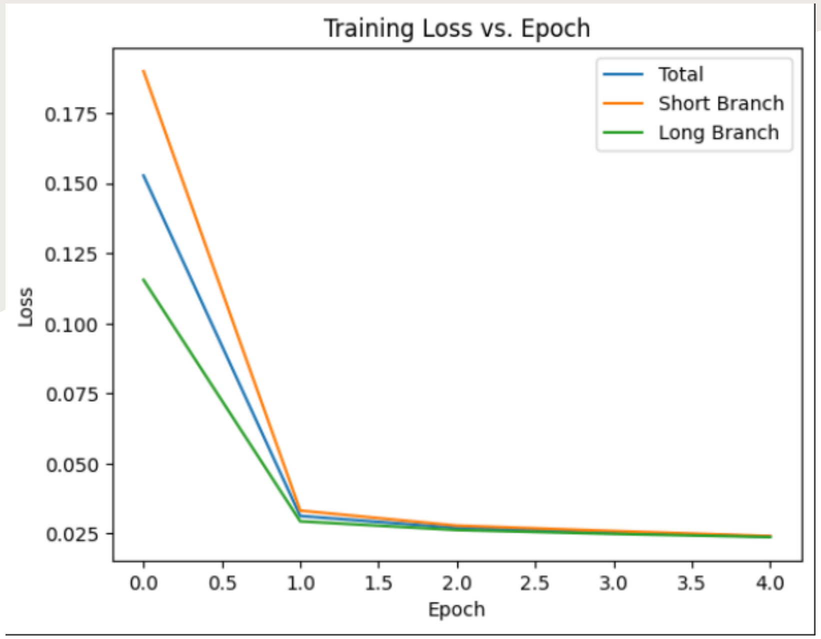


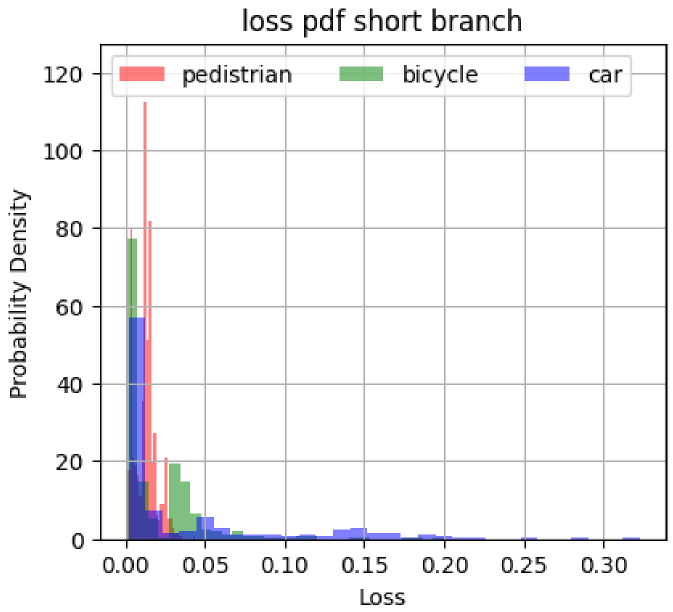
```
Number of encoder parameters: 6481152
Number of decoder short parameters: 1200453
Number of decoder long parameters: 28039110
```

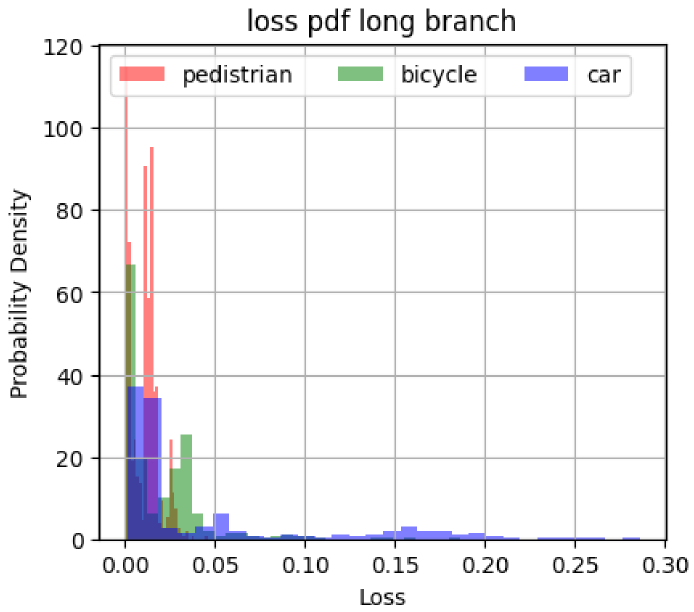
- Early exit statistics



```
Number of samples successfully exited early 139
Total number of pedestrians: 0
Total number of cyclist: 0
Total number of car: 11
```







Radar-based Multibranch and Training Challenges

RODNet Sample Data

Issue with Python script leading to key-value errors during testing sample enumeration.

Inconsistencies in data handling or script compatibility.

Complexity in OLS metrics Calculations

Significant effort required to understand and adapt the code for calculating the Object Location Similarity (OLS) metric

Training Time

Prolonged training durations on GPU, compounded by the lack of available pretrained models to accelerate development.

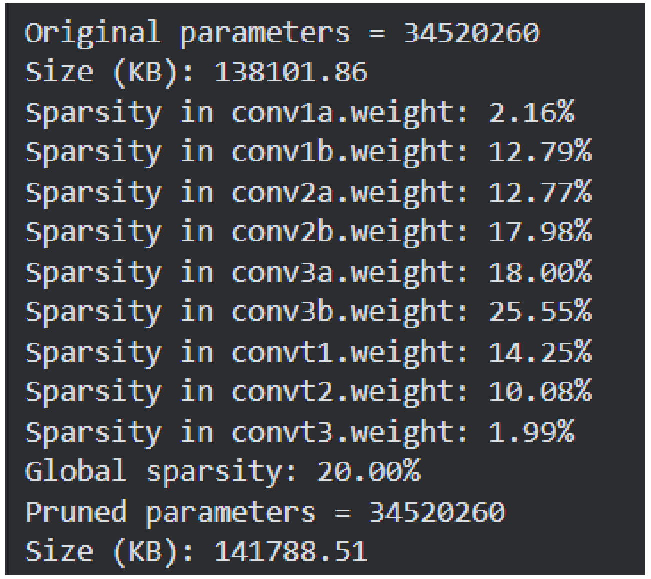
Radar-based Pruning and Quantization Challenges

RODNet Pruning Challenges

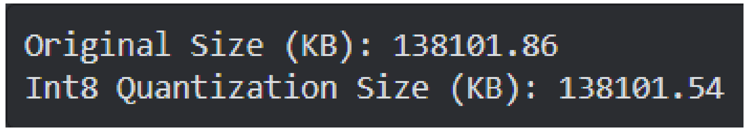
- Utilized PyTorch *Global Unstructured Pruning*.
- Achieved expected global model sparsity (20%), but the model size increased.
- Reason: PyTorch prune functions work as a weight mask, no memory savings are associated because for each pruned weights, the original weights are still stored along with a pruning mask, and the pruned tensors.

RODNet Quantization Challenges

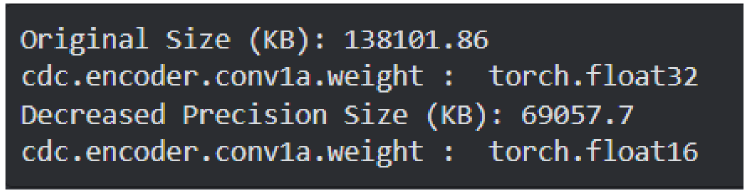
- Int8 dynamic quantization slightly decreased model size.
- Float16 quantization noticeable decrease in model size but requires changes in the training configurations. However, after training for 5 epochs, the model did not generate any detections.



```
Original parameters = 34520260
Size (KB): 138101.86
Sparsity in conv1a.weight: 2.16%
Sparsity in conv1b.weight: 12.79%
Sparsity in conv2a.weight: 12.77%
Sparsity in conv2b.weight: 17.98%
Sparsity in conv3a.weight: 18.00%
Sparsity in conv3b.weight: 25.55%
Sparsity in convt1.weight: 14.25%
Sparsity in convt2.weight: 10.08%
Sparsity in convt3.weight: 1.99%
Global sparsity: 20.00%
Pruned parameters = 34520260
Size (KB): 141788.51
```



```
Original Size (KB): 138101.86
Int8 Quantization Size (KB): 138101.54
```

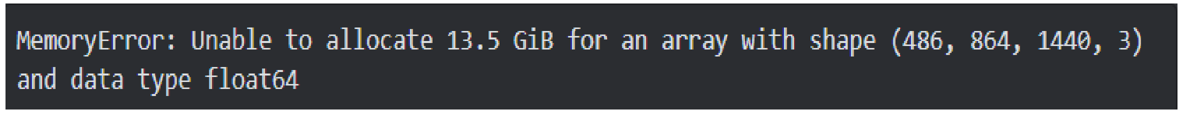


```
Original Size (KB): 138101.86
cdc.encoder.conv1a.weight :  torch.float32
Decreased Precision Size (KB): 69057.7
cdc.encoder.conv1a.weight :  torch.float16
```

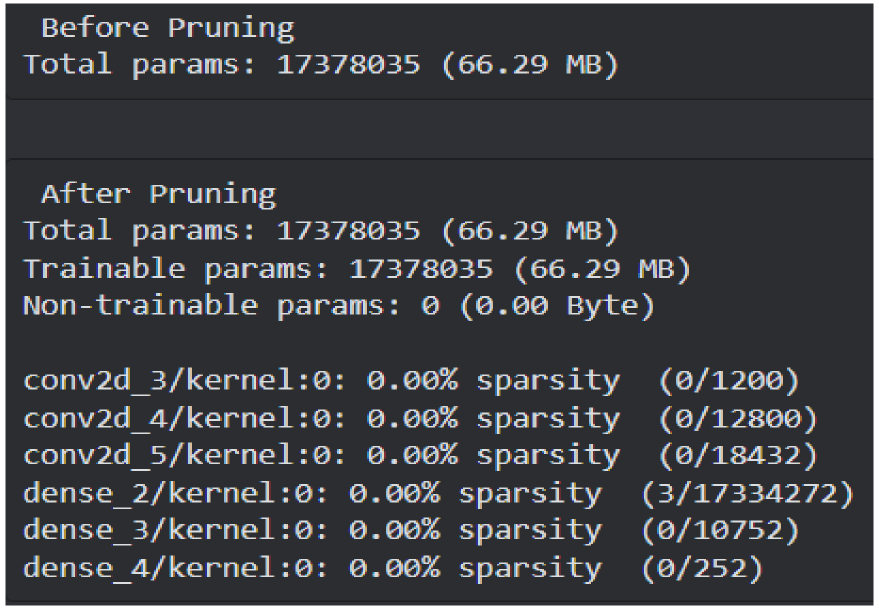
Classification Pruning and Quantization Challenges

Tensorflow Classification (proof of concept) Challenges

- Memory input resolution limitations (864,1440,3) -> (200,200,3) image resizing.
- Pruning sparsity remains 0% after one-shot and iterative pruning.
- Theory might be due to the dataset used which is a series of images from a video sequence which are very similar. The model had possibly overfitted and the dataset did not have too many variations which could have prevented any weights to train with values close to zero.
- Quantization had expected results, but it was not worth the effort to pursue classification tasks since pruning was not successful.



```
MemoryError: Unable to allocate 13.5 GiB for an array with shape (486, 864, 1440, 3)
and data type float64
```



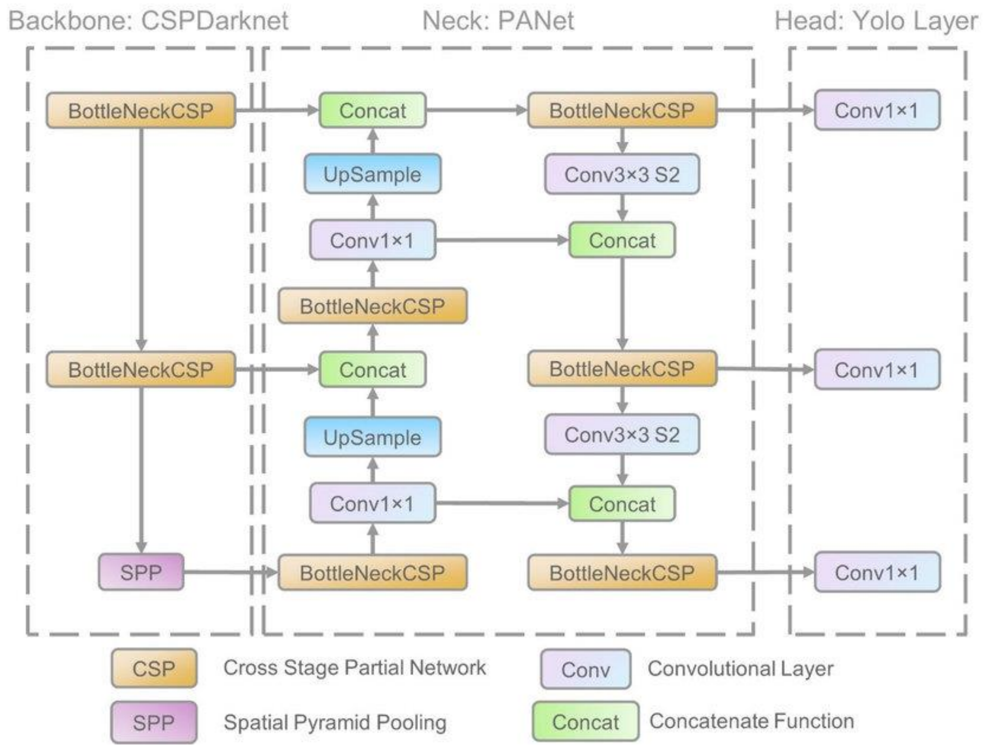
```
 Before Pruning
Total params: 17378035 (66.29 MB)


 After Pruning
Total params: 17378035 (66.29 MB)
Trainable params: 17378035 (66.29 MB)
Non-trainable params: 0 (0.00 Byte)

conv2d_3/kernel:0: 0.00% sparsity  (0/1200)
conv2d_4/kernel:0: 0.00% sparsity  (0/12800)
conv2d_5/kernel:0: 0.00% sparsity  (0/18432)
dense_2/kernel:0: 0.00% sparsity  (3/17334272)
dense_3/kernel:0: 0.00% sparsity  (0/10752)
dense_4/kernel:0: 0.00% sparsity  (0/252)
```

Camera-based Detections Using YoloV5

- Utilized YoloV5 to train camera-based detection model.
  - Trained PyTorch model for 20 epochs -> exported model to F16 TFLite and F16 Tensorflow saved model.
  - YoloV5 architecture consists of 3 parts:
    1. Backbone: CSP Darknet => Responsible for feature extractions from the input data.
    2. Neck: PANet => Responsible for fusing features.
    3. Head: Yolo layer => Responsible for producing detection results such as bounding boxes, scores, and labels.
- Implemented dynamic quantization methods in tensorflow to convert F16 Tensorflow saved model with Int8 quantization.
- YoloV5 layers are not supported by the PruneRegistery (built-in keras layers).



Camera-based Model Quantization Results

Float16

Size: 13.51 Mb

Accuracy	Precision	Recall	mAP0.50	mAR0.50	mACC0.50
57.18%	90.58%	60.76%	64.01%	44.88%	41.69%

Avg Inference Time (ms)	Max Inference Time (ms)	Min Inference Time (ms)
388.67	1297.0	328.0

Int8

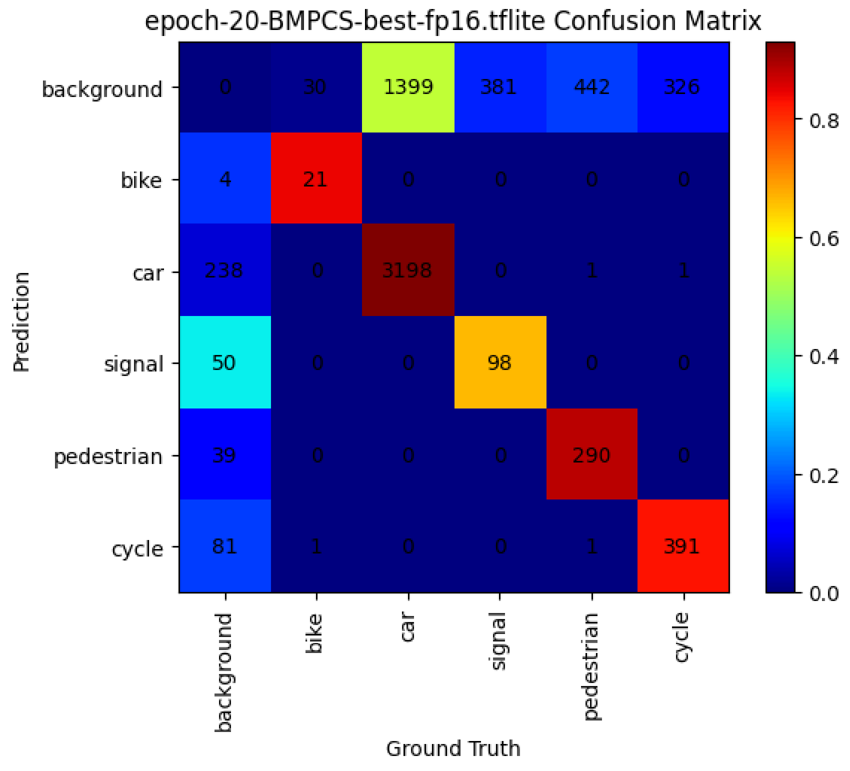
Size: 6.97 Mb

Accuracy	Precision	Recall	mAP0.50	mAR0.50	mACC0.50
57.0%	90.36%	60.65%	63.82%	44.65%	41.47%

Avg Inference Time (ms)	Max Inference Time (ms)	Min Inference Time (ms)
2106.06	3422.0	1828.0

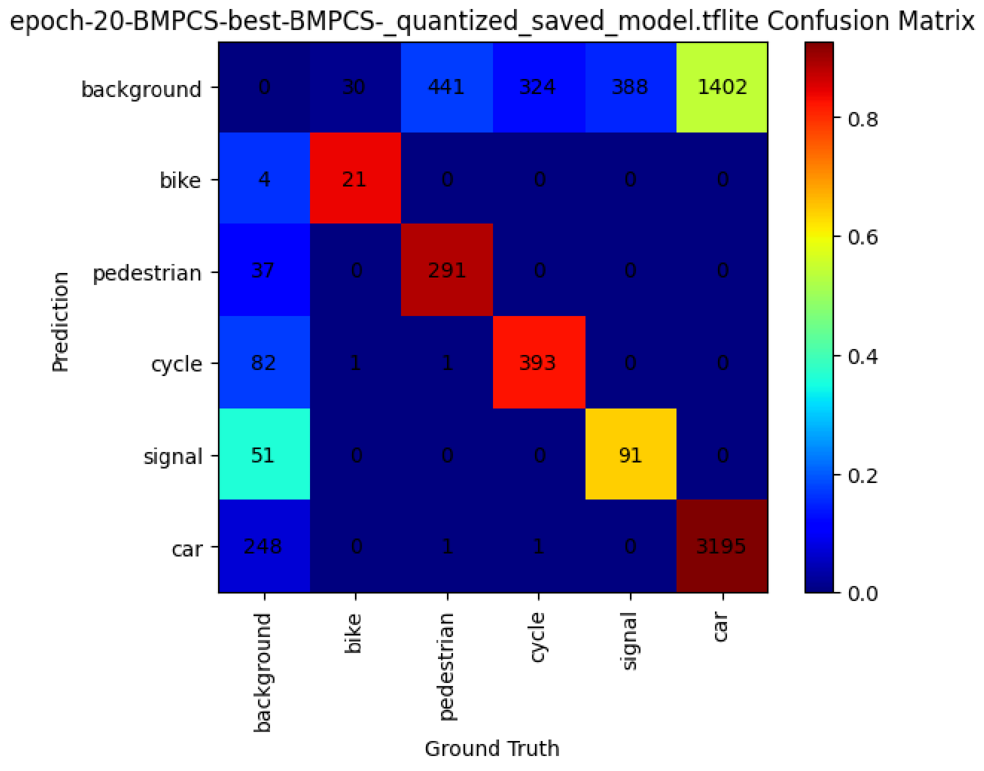
Camera-based Model Confusion Matrix

Float 16



Prediction \ Ground Truth	background	bike	car	signal	pedestrian	cycle
background	0	30	1399	381	442	326
bike	4	21	0	0	0	0
car	238	0	3198	0	1	1
signal	50	0	0	98	0	0
pedestrian	39	0	0	0	290	0
cycle	81	1	0	0	1	391

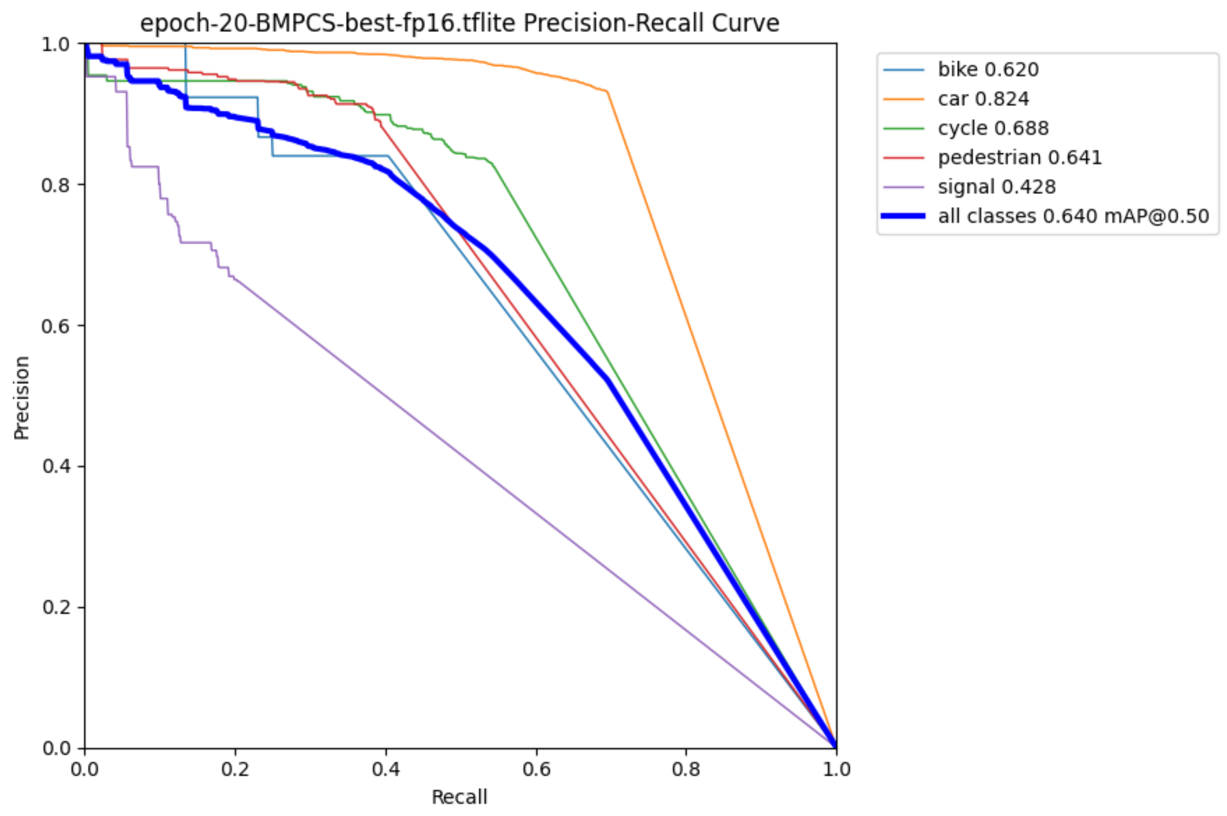
INT8



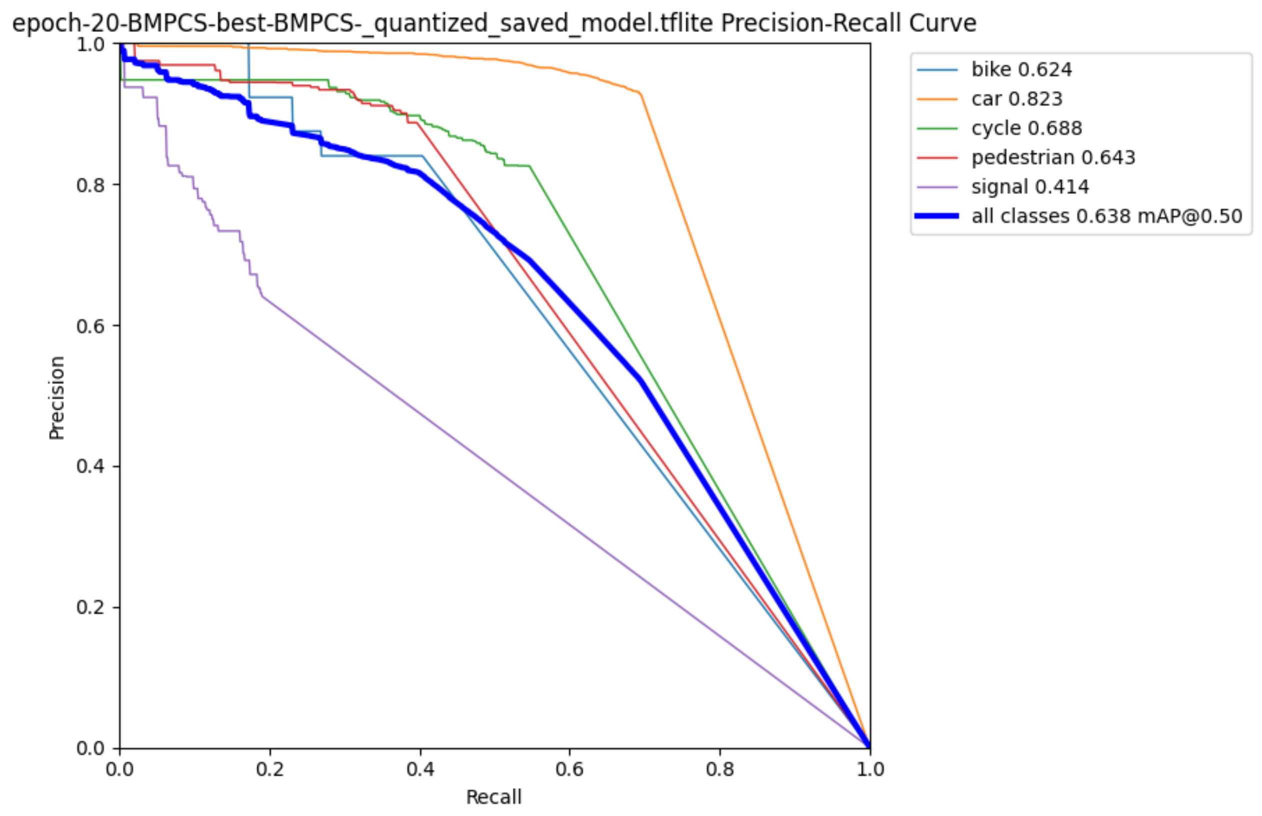
Prediction \ Ground Truth	background	bike	pedestrian	cycle	signal	car
background	0	30	441	324	388	1402
bike	4	21	0	0	0	0
pedestrian	37	0	291	0	0	0
cycle	82	1	1	393	0	0
signal	51	0	0	0	91	0
car	248	0	1	1	0	3195

Camera-based Model Precision Recall Curves

Float16

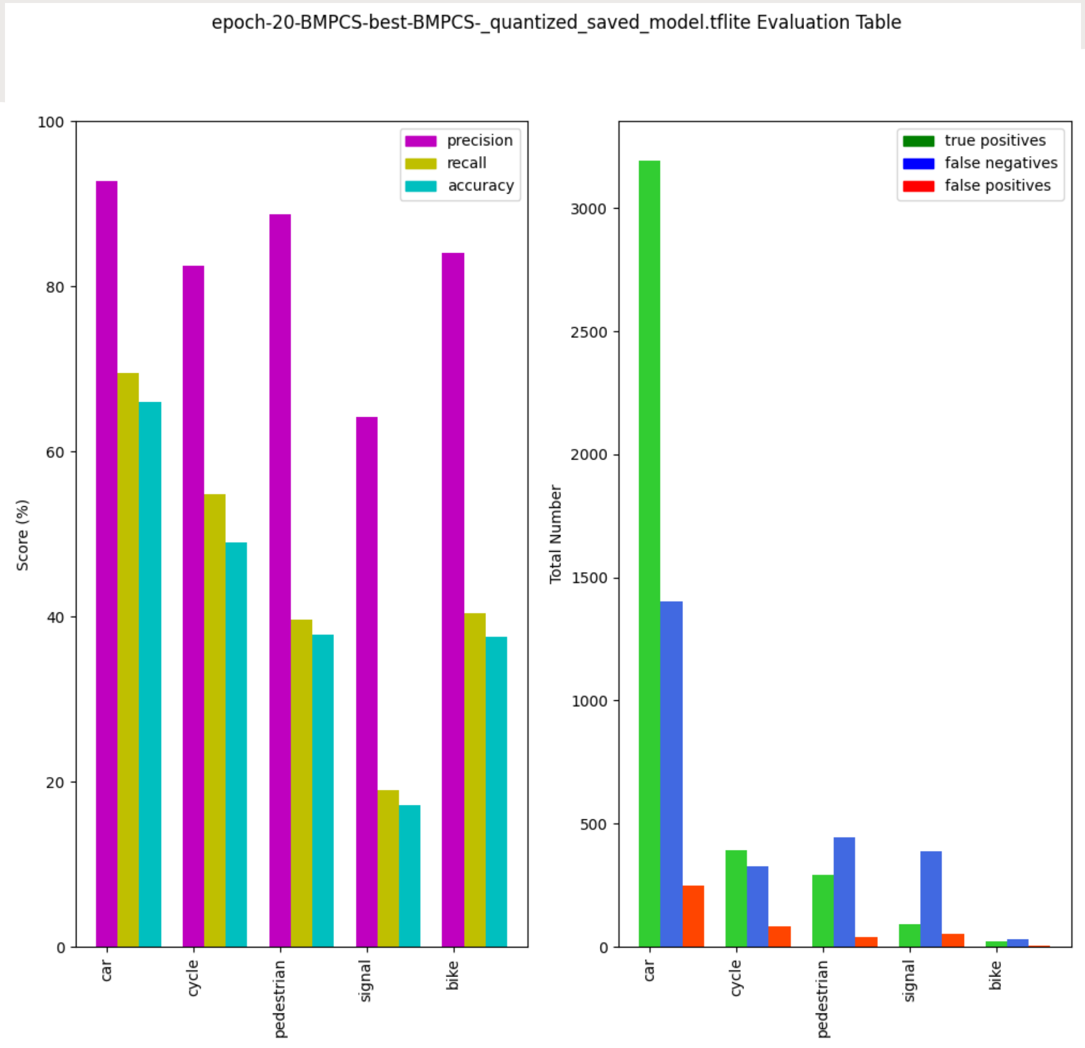


INT8

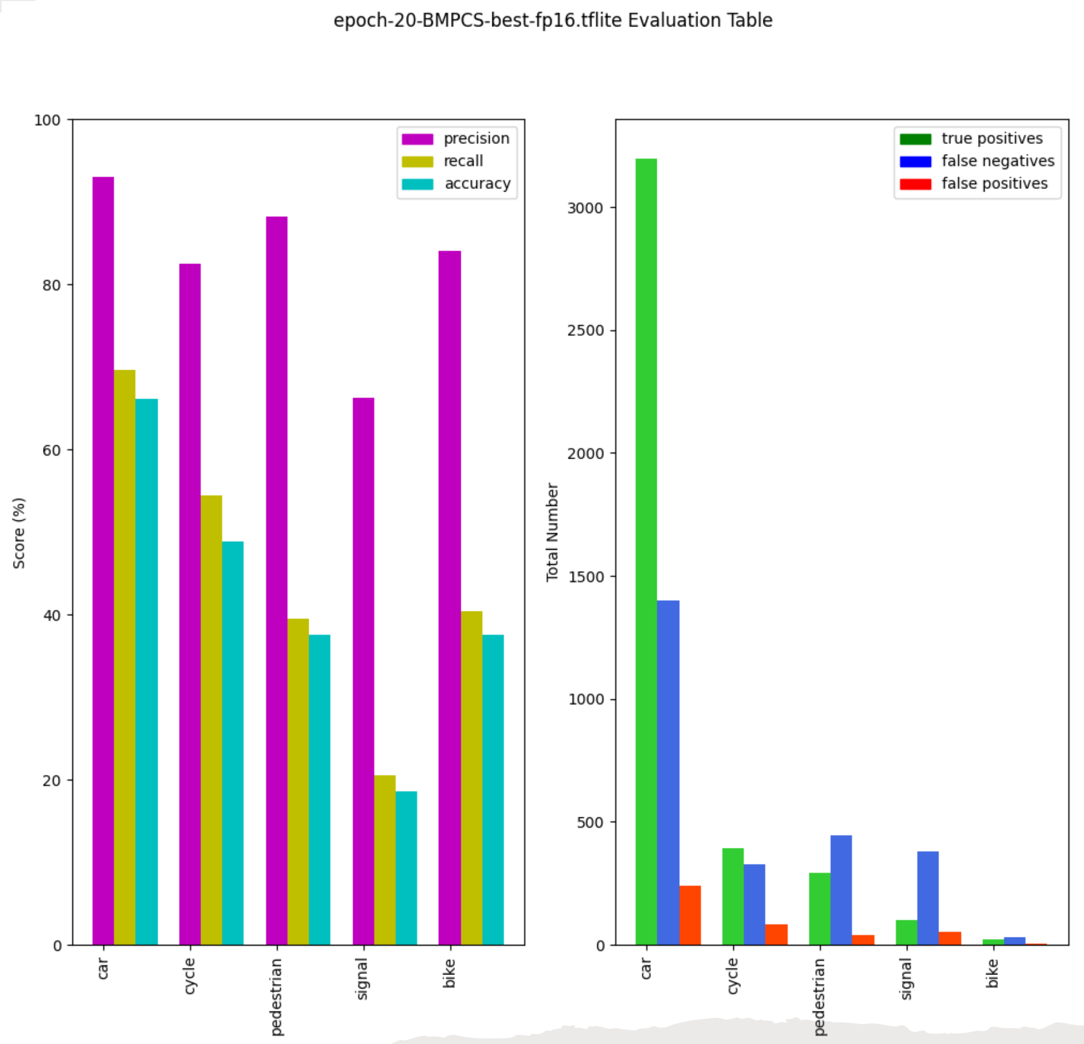


Camera-based Model Class Histogram

Float16



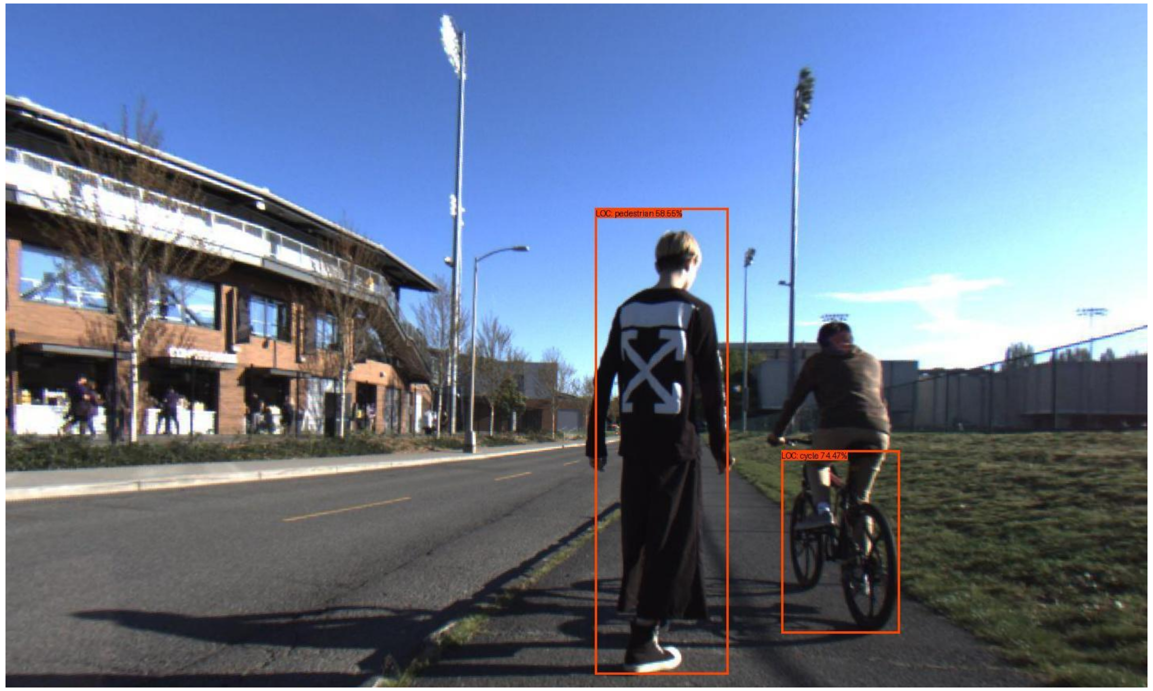
INT8



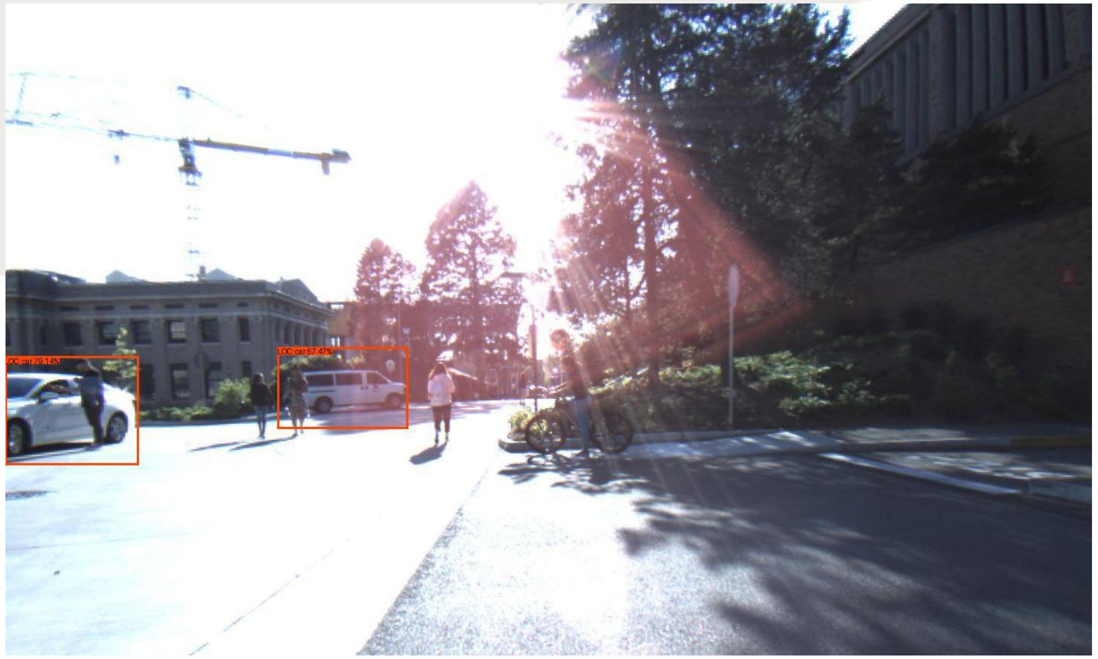
Radar-based vs. Camera-based Detections

Camera-based Detection Results

Easy Difficulty



Medium Difficulty

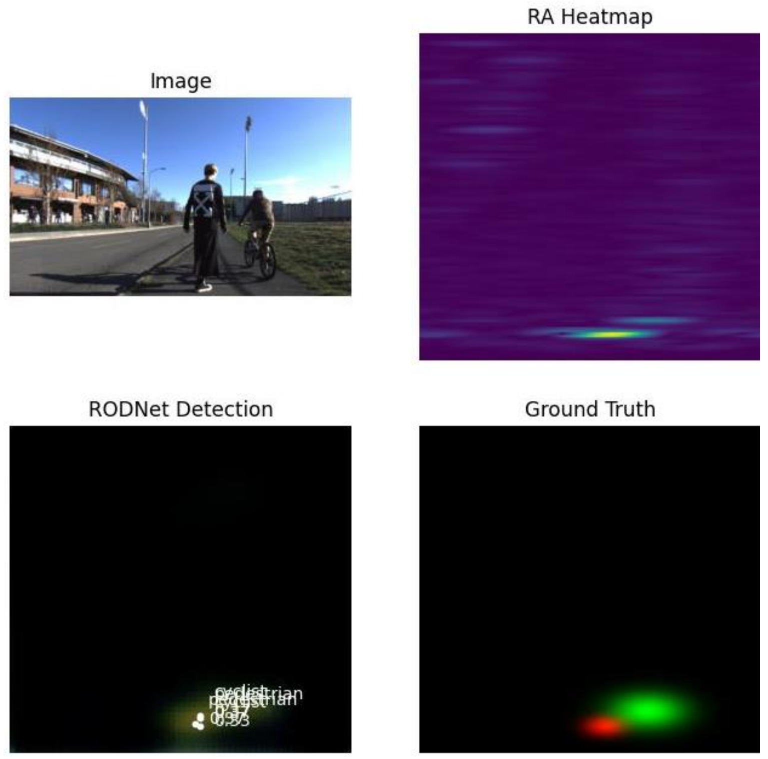


Hard Difficulty

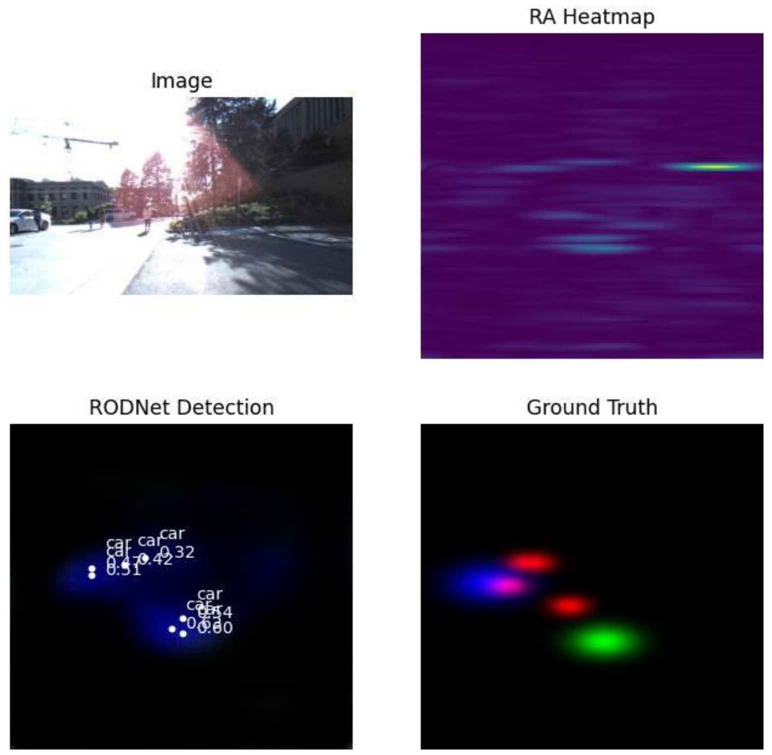
Radar-based vs. Camera-based Detections

Radar-based Detection Results

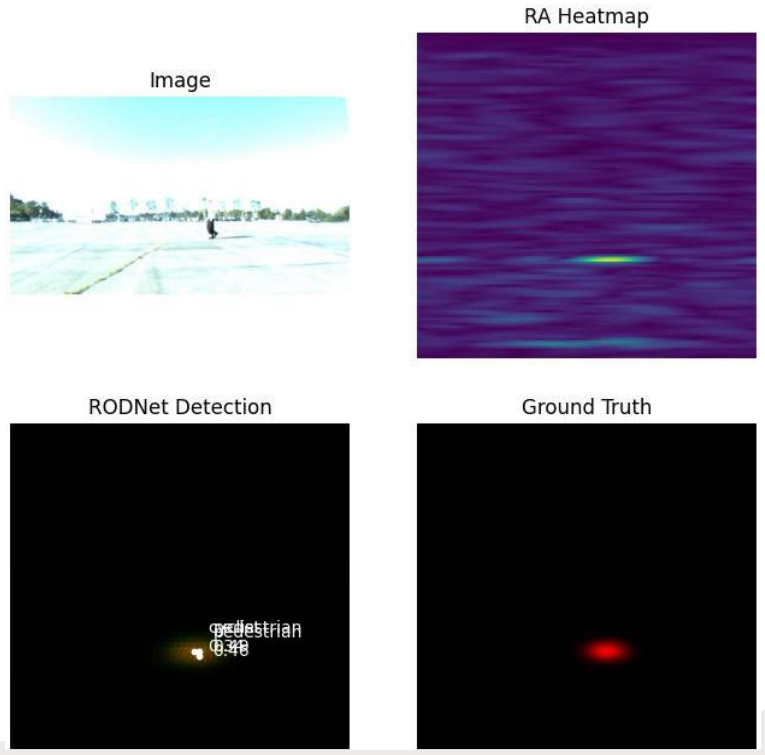
Easy Difficulty



Medium Difficulty



Hard Difficulty



Future Work

Optimization Techniques Implementation:

- Integrate pruning and quantization in PyTorch for RODNet models to streamline the network and improve computational efficiency.

Performance Metric Enhancement:

- Implement refined measurement techniques for Average Precision (AP), Average Recall (AR), and accuracy to evaluate both the base and multibranching RODNet models.

Camera-Based Model Refinement:

- Apply oneshot and iterative pruning methods to camera-based detection models to enhance their performance in real-world conditions.

Process Dataset:

- Remove all frames that does not have annotations in provided text files.

Any Questions?