Infrared Person-Car Dataset

Dataset :

Link:
https://drive.google.com/drive/folders/1fEKQhJSjmt7k_rkdvTcQ9snjRf740lav?usp=sharing

Data Distribution :

Train dataset : 100 Images
Validation dataset : 500 Images

Classes :

1. Person
2. Car

Annotation Format: MS-COCO

Model :

Name: Cascade Mask-RCNN with Swin-Tiny backbone and 1x GPU support

Training :

Environment : Google Colab with Tesla-V4 GPU (Linux)

Scheduler :

Epochs : 36(3x of Mask-RCNN runtime, best possible baseline
Samples_per_gpu : 2
Workers_per_gpu : 2

Method :

In this experiment, the few-shot learning is adopted.
So, the training set has opted as the validation /test and vice versa.

Colab Link:
https://colab.research.google.com/drive/1lJ6CFQfihe8I1B1kOqNITZfoqwPjVKUR?usp=sharing

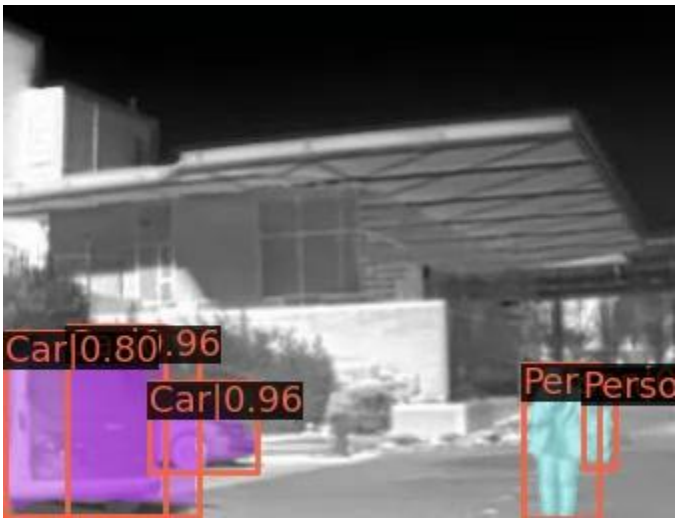
Evaluation :

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.332
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.583
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.334
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.280
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.649
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.375
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.375
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.375
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.329
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.688
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.286
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.555
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.273
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.224
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.631
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.324
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.324
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.324
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.273
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.664
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

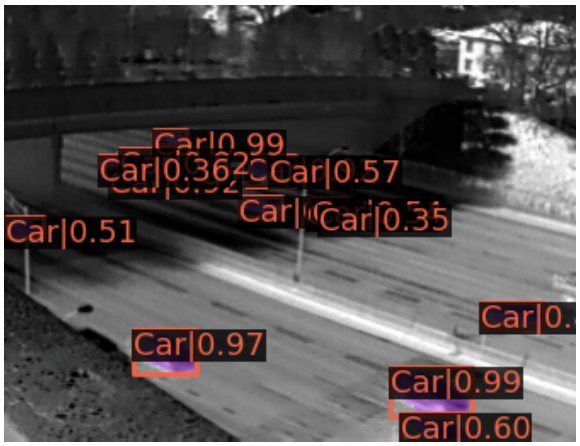
Test Results and explanation :

As we can see the objects have been classified and objectified quite nicely. The objects i.e. person and car are quite visible in the pictures and thus the model is predicted precisely.



In this, we can see again that the prediction has been quite nice, but some extra bounding boxes are present. This can be explained using the sliding window feature, the cycle shift effect adds the neighborhood thus predicting itself as a different object.

In this prediction we can see multiple bounding boxes over a single car frame; this phenomenon is normal in the Mask-RCNN model family. This can be cured by not using the decay rate of the optimizer. The experiment was possible and it showed predictions like the picture down below.



Leaving all these good predictions there were some bad predictions as well which completely decreased the accuracy.

In this picture, we can see that the building has been objectified as a car, which is normal as both of the structures are rectangles in a 2d plane, and these predictions are everywhere in the val predictions decreasing the accuracy scores.

Further Experiments：

After trying a transformer-based model, the QueryInst model was opted to check if query-based modeling gives better results or not.
As the objects are having quite linear boundaries, so query-based modeling should give a nice accuracy can be a base idea.

Scheduler : x3(36 epochs)
Results:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.118
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.224
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.115
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.105
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.294
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.468
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.471
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.471
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.421
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.825
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.097
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.187
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.102
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.079
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.491
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.408
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.414
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.414
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.361
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.778
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

It is visible that Query-based models couldn't perform any well in the dataset. Most of the data has been left untouched and that’s a bad sign. Some humans are not masked so the prediction is not predictable.

Failing on QueryInst, the next model opted for was GCnet as it performed very well for the Hard-Hat Dataset.

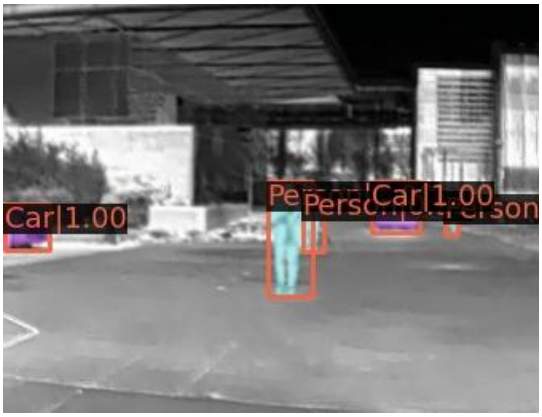
Why use GCNet? In the previous experiment, we have seen that GCNet has performed well on the smaller objects very precisely, also it differentiated some similar-looking objects so that features can be used here because the errors are similar.

Scheduler : x1 (12 epochs)
Results :

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.457
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.680
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.527
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.401
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.809
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.491
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.491
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.491
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.443
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.834
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.413
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.662
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.470
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.351
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.784
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.440
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.440
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.440
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.388
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.799
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

Results can be seen upgraded 2x times, so the GCNet performed well in this scenario. Previously some persons were left in the images, but GCNet mostly identified them and masked them quite nicely.



Also the building has been treated as the background and that's a very good improvement from the previous model. Objectifying the cars and overlapping bounding box problem has also been rectified here.

Still, there are some people unmasked, and that is also a problem to overcome.
In the image below we can see only cars have been inferred but the human in the upper right corner.

After this DCN has been opted for rechecking on this dataset. Moreover, GCNet and DCN are giving promising performances competing for neck to neck. DCN's spatial character is surpassing the small objects and curvy objects, that's why this also can be beneficial in this dataset.

Scheduler : x1(12 epochs)

Results :

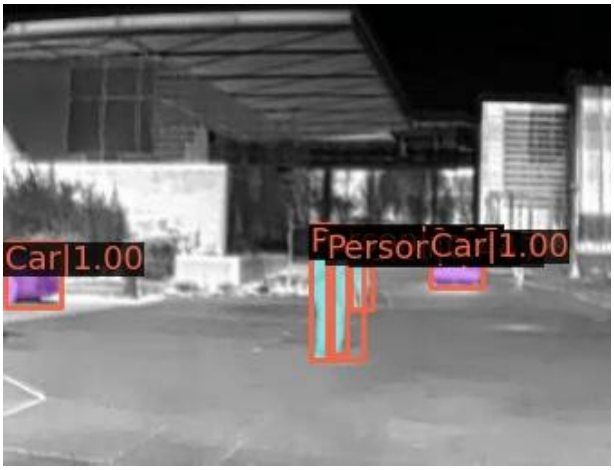
```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.462
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.681
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.533
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.406
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.804
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.448
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.833
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.418
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.663
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.469
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.358
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.776
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.444
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.444
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.444
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.394
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.788
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

Seeing the results it can be said that the accuracy is slightly improved for DCN, but both of them are providing quite similar results.
Though most of the errors done by Swin Tiny Cascade Mask-RCNN have been rectified by GCNet, still it failed in objectifying humans in small figures. But DCN performed well and rectified this error too.

The cars in the sideways are also being updated in this model implementation which is very positive feedback.

Most of the humans are correctly classified in the seq-19 data of the dataset. Also, the buildings are not inferred a bit, showing a very nice instance segmentation.



Conclusion :

The model prediction over the test set was very good compared to the validation set. Although similar objects were present in the train image, still the model somehow learned the buildings' cars and degraded the training accuracy. Leaving that, the model performed quite well. Updating with DCN and GCNet surpassed most of the errors standing as the best-opted model for this task more cheaply.

THE END