VHR-10 Dataset

Dataset :

Link : https://github.com/chaozhong2010/VHR-10_dataset_coco

Data Distribution :

Train dataset : 150 Images
Validation dataset : 500 Images

Classes :

1. Airplane
2. Ship
3. Storage_tank
4. Baseball_diamond
5. Tennis_court
6. Tennis_court
7. Basketball_court
8. Ground_track_field
9. Harbor
10. Bridge
11. vehicle

Annotation Formatanopted,: MS-COCO

Model : InstaBoost with ResNet-50 backbone and 1x GPU support

Training :

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

Scheduler :

Epochs : 48 (4x of Mask-RCNN runtime, best possible baseline)
Num Workers : 1
Samples_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/1M0Px8S58QwUZYXZiRI_gBisUseBi0iHr?usp=sharing

Evaluation :

Train-Set :

Accuracy : 98.2207
Loss : 0.2458
Bbox loss : 0.0957
Mask Loss : 0.866
Class Loss : 0.0412

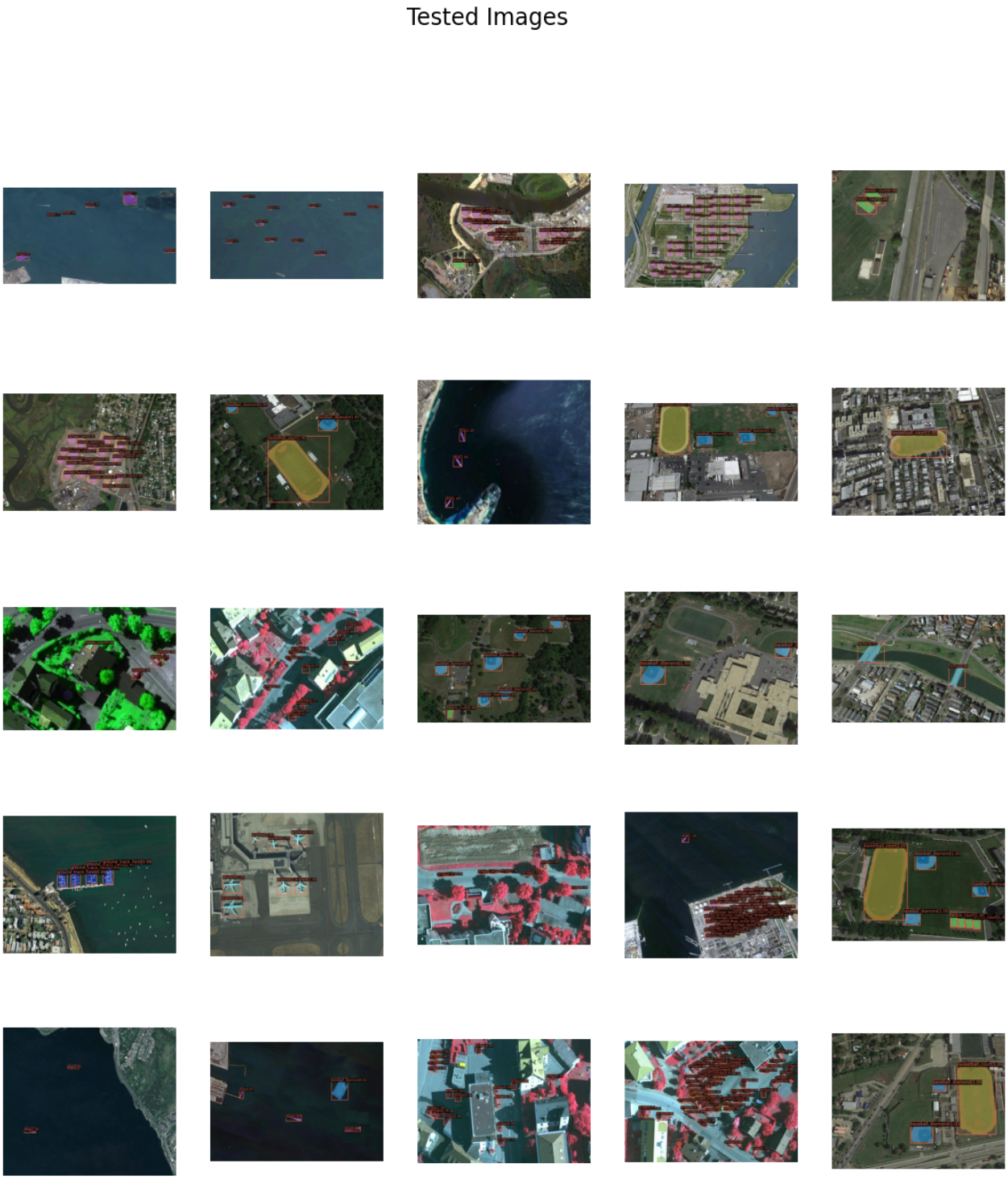
Validation-Set :

Bbox mAP : 0.661
Segm mAP : 0.661

P.S. As we can see a very powerful prediction has been done by the model. In the model intuition, it has been told that the model will smoothen the rough edge points of the objects. So, as the dataset mainly occurs with the objects which have predictable linear edges, this helped

the model to identify the object's presence in a locality very easily and thus getting nice accuracies on both of the datasets.

Test Results :



Conclusion :

The model Instaboost comes with some nice features as well it also indicates that too many curves of the object may not be identified by the model due to its smoothening property. The increased number of epochs helps the model to achieve better results, still, runtime on 100 images took more than 1.30 hours on a single GPU support machine, and thus it can be taken as a slow training model.

Improvisations:

Model: Cascade Mask-RCNN with Swin Backbone

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 been opted as the validation /test and vice versa.

Colab Link :

https://colab.research.google.com/drive/1JVebnObpNKiCZeWTr252VAsr3vEVtDNt?usp=sharing

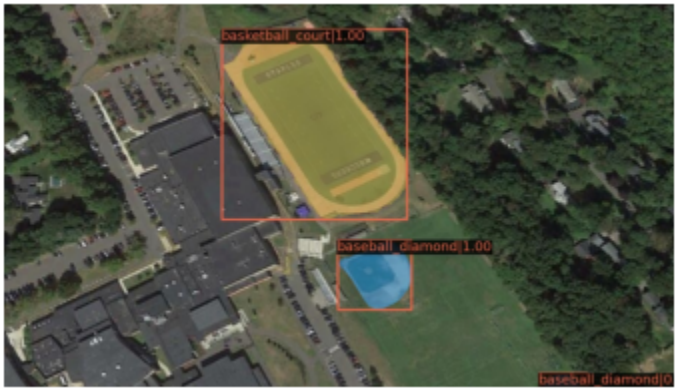
Evaluation :

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.716
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.925
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.813
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.678
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.711
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.654
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.786
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.786
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.786
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.706
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 ] = 0.768

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.657
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.917
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.716
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.555
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.645
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.651
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.719
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.719
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.719
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.579
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.706
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.744
```

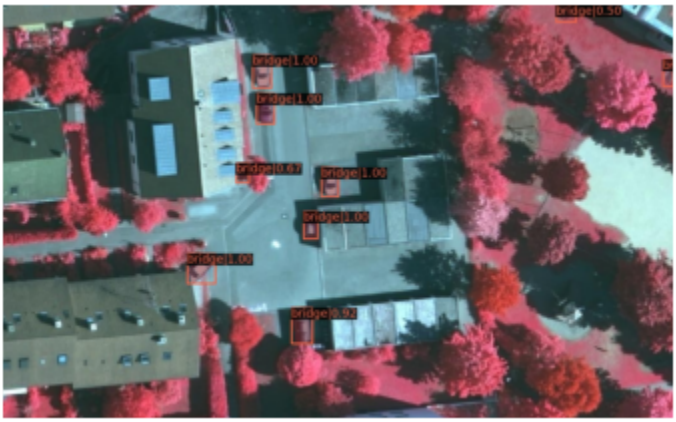
Test Results and explanation :

In this image we can see that the airplanes have been precisely segmented and objectified. The class accuracies are almost 1 in
every case.



Here we can see that the grounds are identified nicely but the smaller cars,based GPUin this case, have been left. This can be explained that the very short neighborhood of each object led them to be unidentified.

The bridges are identified nicely but we can see that they are elongated towards the road of the mainland. So, this problem is visible because of the parallelism of the bridge structure or likelihood (bridges and roads are very similar and classified as different classes when the neighborhood is present i.e. mainland or sea/river). This phenomenon can be explained as a likelihood property given by the attention structure.



Nothing problematic here, the cars are quite well objectified.

Though the airplanes are nicely predicted the tankers are not, because in a neighborhood every single one has been predicted, but some are left untouched.

Getting very nice results from the Swin transformer, the next implementation was with DCN as it has shown much proficient accuracy on the Thermal Dog Dataset.

Scheduler: x1 (12 epochs)

Results :

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.537
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.781
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.659
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.350
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.586
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 ] = 0.661
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.661
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.661
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.483
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.707
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

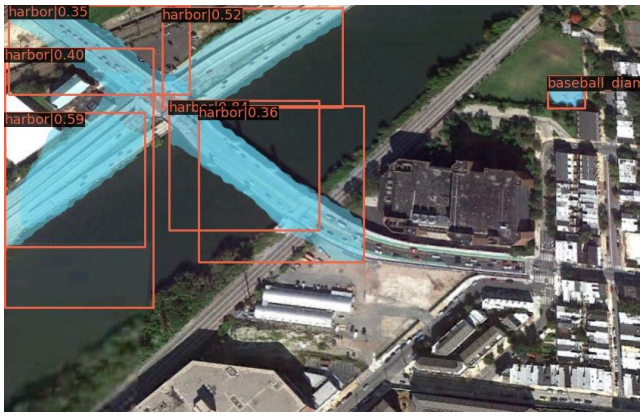
```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.498
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.784
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.591
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.246
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.564
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 ] = 0.604
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.604
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.604
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.425
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.651
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = -1.000
```

As we can see, DCN has raised a decent accuracy, but it couldn’t surpass the brilliant performance of Swin tiny on Cascade Mask-RCNN.

Test Results :

The planes and the ships are predicted very well as we can see in this picture, there was hardly a ship or plane missing in all of the predictions. On the note of the previous test, the accuracy on the storage tanks was upgraded nicely with DCN.

The prediction went devastating around the harbor class as it predicted the roads as that even we can see overlaps of single class decreasing the results.



Cars are also predicted very well and can be taken as a better prediction than the swin-tiny one.

The prediction also went a bit worse in this prediction for the track field class. There were some overlaps in those. Still, most of the predictions of this class are better, so only showing the worst prediction.

Conclusion: The model is quite reactive towards the classes and segmentations are quite well done for each object. Still, some errors are there which can be diminished by upgrading the attention layers. Implementing with different models still gave some betterment but couldn't surpass the best accuracy so far. So, the transformers-basedGPU model performance can achieve higher results are proven in this scenario.

THE END