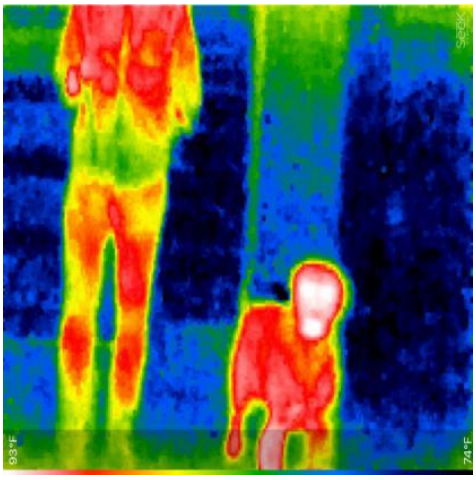
# Dataset Implementation

## Thermal Dog Dataset



### Dataset :

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

**Data Distribution :**

Train dataset : 142 Images
Validation dataset : 16 Images

**Classes:** Human (1), Dog (2)

**Annotation format:** MS-COCO

### Model: Mask-RCNN with r50 backbone.

### Model Training :

The repository mm-openlab/mmdetection has been used to train the dataset.

## Model environment :

Google Colab notebook with 1x GPU support.

In this project, the model has been trained on 2 techniques.

## **Classical Way :**

**Colab Link:**
**https://colab.research.google.com/drive/1u1_ippijor81betCjEpxrNU8db6krRtG?usp=sharing**

## Model Evaluation :

Final Bounding Box Results :

**Train Data :**

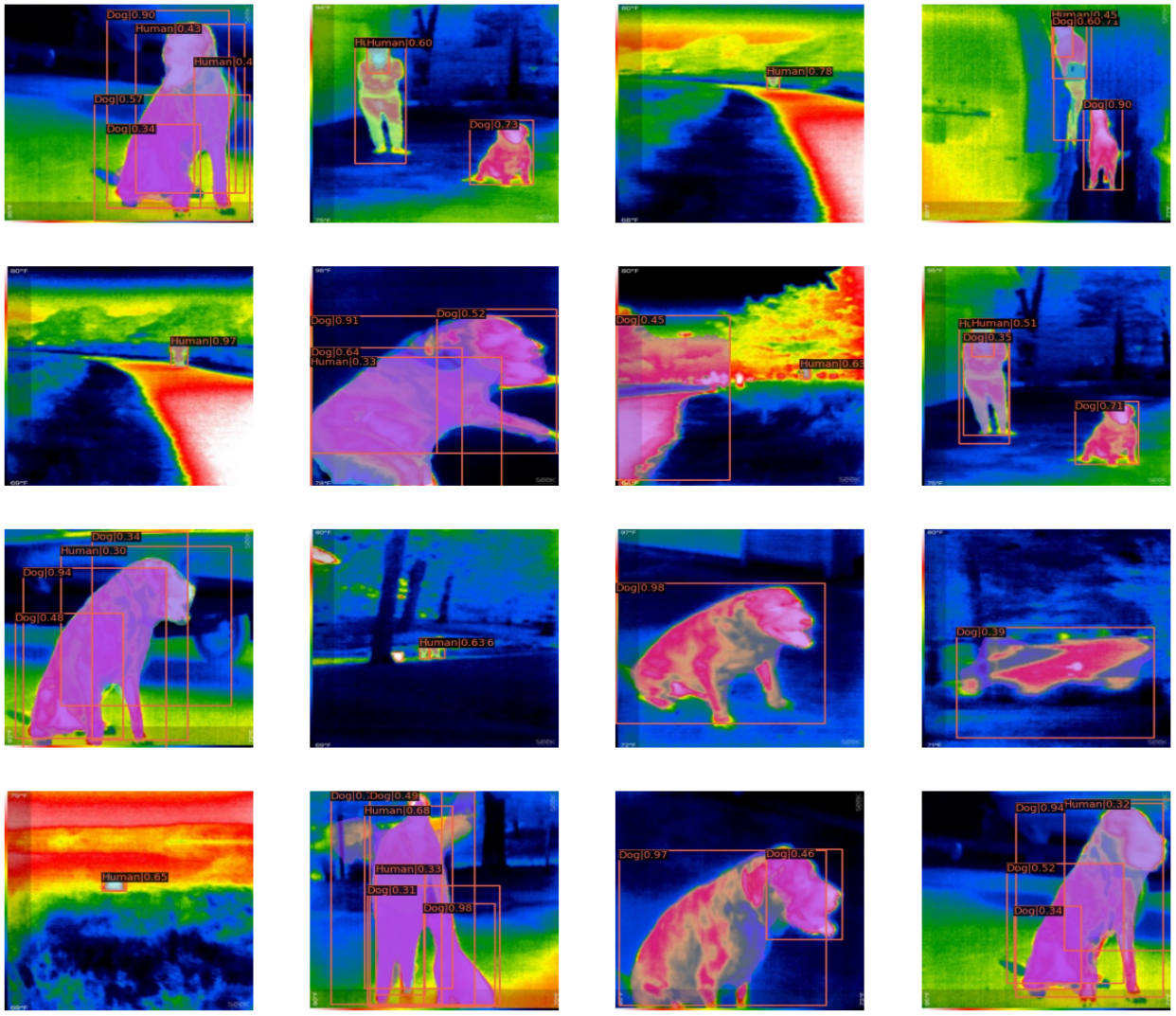
Accuracy : 98.2422
Loss : 0.3198
Bbox loss : 0.0677
Mask Loss : 0.1747
Class Loss : 0.0555
Bbox mAP : 0.6100
Segm mAP : 0.6620

**Validation Data :**

Segm mAP : 0.662

**Model Testing:** The model has been tested on the validation data.

Tested Images



**Observation :**

In the training process, we can see that there are multiple bounding boxes on the images wrongly predicted with high threshold accuracy; these errors can be minimized by calibrating the optimizer performance. Primarily the optimizer used here is SGD, but tweaking that part to Adam may show better results.

# Few-shot way :

Previously the model was training on 142 images and testing the validation set with 16 images, but in a few-shot way, we just reversed the process. Now, the train images are the validation ones and vice versa.

**Colab Link:**
https://colab.research.google.com/drive/1unTpJHOhNujliAcz7Fs82JbduY6aAiJe?usp=sharing

## Results :

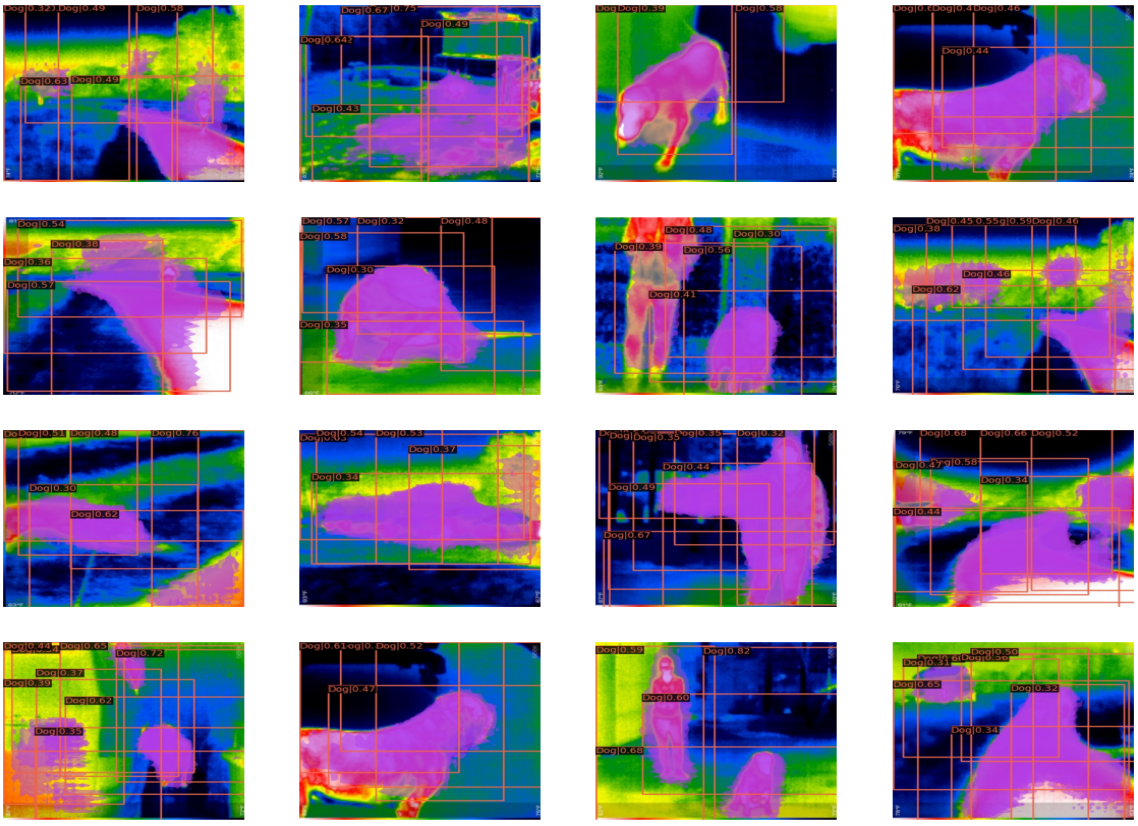
**Train Data :**

**Bbox mAP** : 0.007
**Segm mAP :** 0.0790

**Validation Data :**

**Bbox mAP :** 0.007
**Segm mAP :** 0.022

Tested Images



## Improvisations :

After failing on Mask RCNN, the QueryInst model has opted for training.

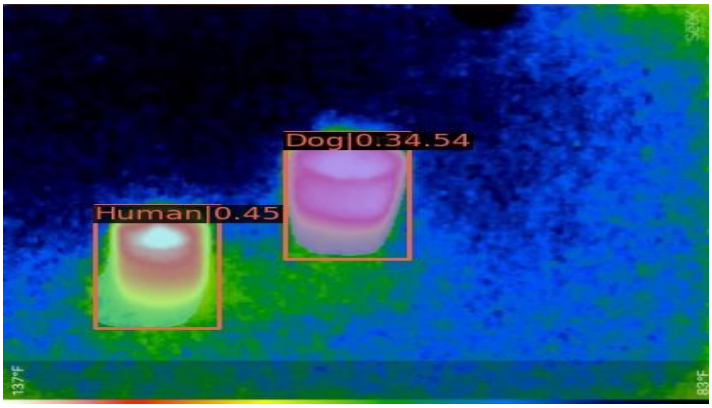
**Why chose QueryInst?** QueryInst has a different approach than Mask-RCNN. Using queries over the image features will positively update the result.

**Scheduler:** x3 (36 epochs)

**Results :**

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.281
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.352
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.312
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.201
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.261
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.361
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.847
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.854
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.854
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.798
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.794
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.929

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.113
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.257
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.108
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.125
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.132
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.160
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.501
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.508
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.508
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.608
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.527
```

We can see the model has upgraded the results to another level. Still, when looking at the results, only some of the images were annotated and the rest of them were left untouched as well as the annotations were completely bizarre.



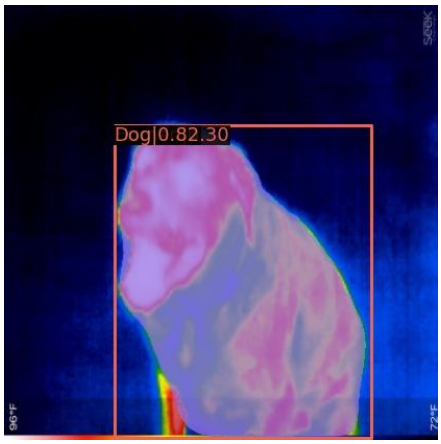

As the model with queries failed the next objective was to check how the model can update itself by looking at various structures in the pictures themselves. Then GCNet has opted.

**Scheduler:** x1 (12 epochs)

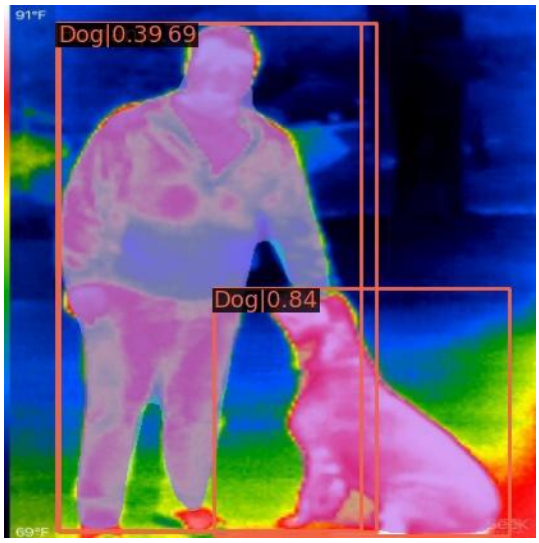
**Results :**

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.656
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.841
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.762
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.415
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.604
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.761
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.751
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.751
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.751
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.545
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.685
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.854

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.609
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.818
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.751
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.319
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.566
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.735
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.698
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.698
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.698
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.465
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.652
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.796
```

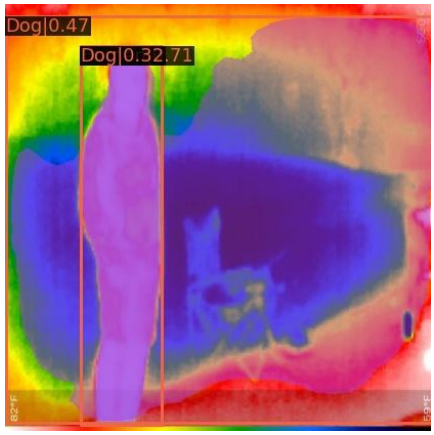
The results have gone too far better than the previous modeling. The results were having annotations and the bounding boxes were better than most of the previous cases.




We can see that the segmentation and the bounding boxes are more clear than the mask-rcnn implementations as well as the segmentation mask itself.
Still, in some pictures, some overlapping bounding boxes show a slight anomaly of the model.




And some of the masks were inflated badly making it count as bad predictions.




After learning about DCN  and its spatial deformation module's function, the next model opted was DCN.

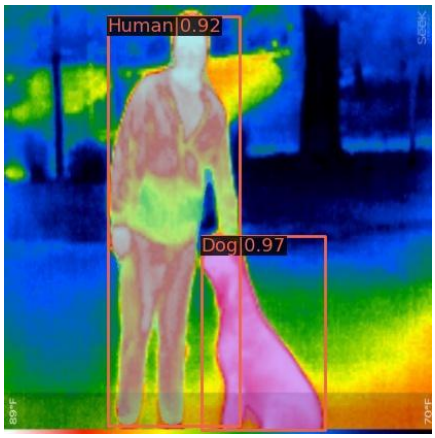
**Scheduler:** x1 (12 epochs)

**Results :**

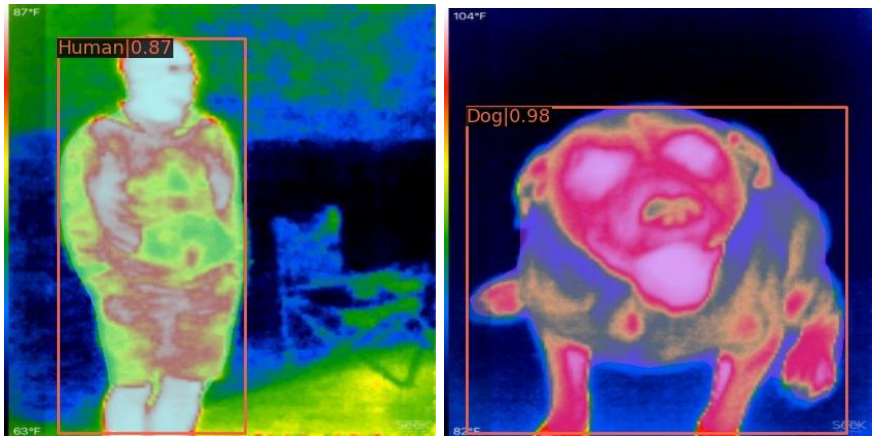
```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.718
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.891
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.828
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.536
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.667
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.832
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.799
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.799
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.799
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.677
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.737
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.884

Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.672
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.891
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.813
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.401
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.627
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.788
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.742
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.742
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.742
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.602
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.695
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.823
```

The results are updated up to 6 % positively, but the results are now more prominent and quite accurate for all cases, somehow there are some fewer bounding boxes opted, airplanes, but the previous problems have been resolved completely.

The overlapping bounding box problem has been rectified by the model.




Also, the segmented masks are not scattered around the whole picture body, but only in the object.




**Conclusion:** The dataset has a very small number of training data that's why the simple model fails to create remarkable accuracies thus failing, but the deformable models were pretty accurate due to their primary function to detect the spatial transformations.

# THE END