COL780: Computer Vision
Assignment 4: Breast Tumor Detection using object detection techniques

Team Members:
Bogam Sai Prabhath (2023AIB2079)
Ajay Tomar (2023AIB2071)

Link for Models:
- Model_1
- Model_2

---

1. Convolution-based object detection using Faster R-CNN

The architecture of Faster R-CNN is designed to process the input image only once through a convolutional network. Then, it uses the feature map obtained to predict object bounds and objectness scores at different locations simultaneously. Faster R-CNN can also operate in near real-time. faster R-CNN contains the classification network and regression network.

1.1 Data format used:

We have used the coco data format. The coco annotation data set is used for both models further as it gives more information about the data than the Yolo data format.

1.2 Image preprocessing:

As the images are X-rays, we did not convert them into grayscale. we normalized the image's pixel values into the range -1 to 1. This helps in training the model efficiency and decreases the bias of the pixel values.

1.3 model used:

We have finally used the pre-trained model of `fasterrcnn_resnet50_fpn` we have loaded the weights of the model and fine-tuned the complete model we have tested the other models by freezing some of the layers and fine tuning the last n layers of the backbone (n=8) our results are shown below.

1.3.1 BackBone model:

The backbone is used for convolution and normalization to generate the feature map. We have used the ResNet 50 architecture as the backbone. We have also compared the performance of the model with vgg16 with full fine-tuning and partial fine-tuning. We observed that the ResNet 50 performs better than Vgg.

1.4 parameters used:

- **Optimizer**: Adam Optimizer
- **Learning rate**: 1e-4
- **Weight decay(L2)** : 1e-4
- **Scheduler**: stepLR
- **Stepsize**: 1
- **gamma**: 0.8
- **Number of classes**: 2 (background, tumor)
- **Epochs**: 30

1.5 Data loading :

The faster rcnn model takes the 2 arguments one is the list of tensors, which is images. The length of the list is the batch size, and the second argument is the targets containing the list of dictionaries. Each dict contains the box and class of the box. The format of the box is (x1,y1 x2,y2), representing the box's left top corner and the box's right bottom corner. Each box is labeled as class 0. During the training, it accepts two arguments; during inference, it accepts one argument about images.

1. A CustomDataset class is used for loading the dataset in batches using Pytorch implementation
2. A collate function is used for data customization according to the specific requirements of the dataset or the model being trained
3. Training of the model is performed as follows:
    a. Load a batch of images and targets.
    b. Perform forward pass through the model, compute losses, and sum them.
    c. Backpropagate gradients, update model parameters, and append batch loss to list.
4. Validation over the trained model is also performed in the batches. For each batch:
    a. Load a batch of images and targets.
    b. Perform forward pass through the model (no gradient calculation).
    c. Predict the boundary boxes for each image in a validation dataset
    d. Compute validation losses and append batch loss to list.

1.6: Results:

1.6.1 loss Curves :

Train Loss: 0.18

Validation loss: 0.16

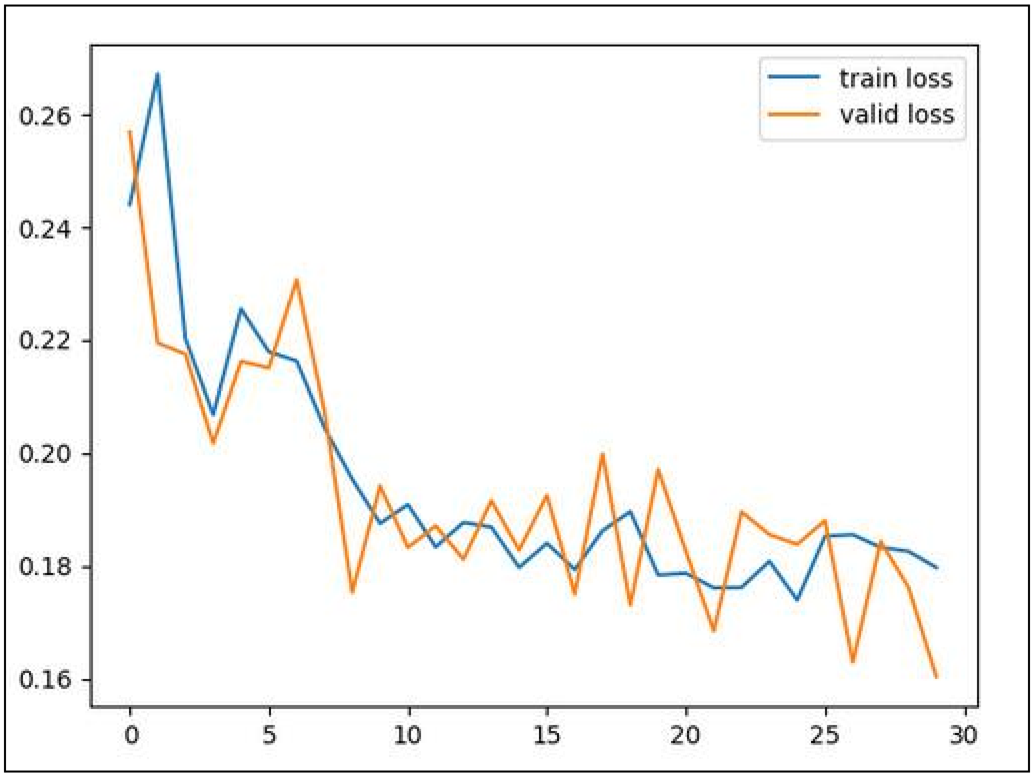

Figure 1.1 train and validation loss vs epochs

1.6.2 output with different Non-Maximum Suppression thresholds:

The model is evaluated on different nms thresholds and the results are shown below in Fig 1.2.

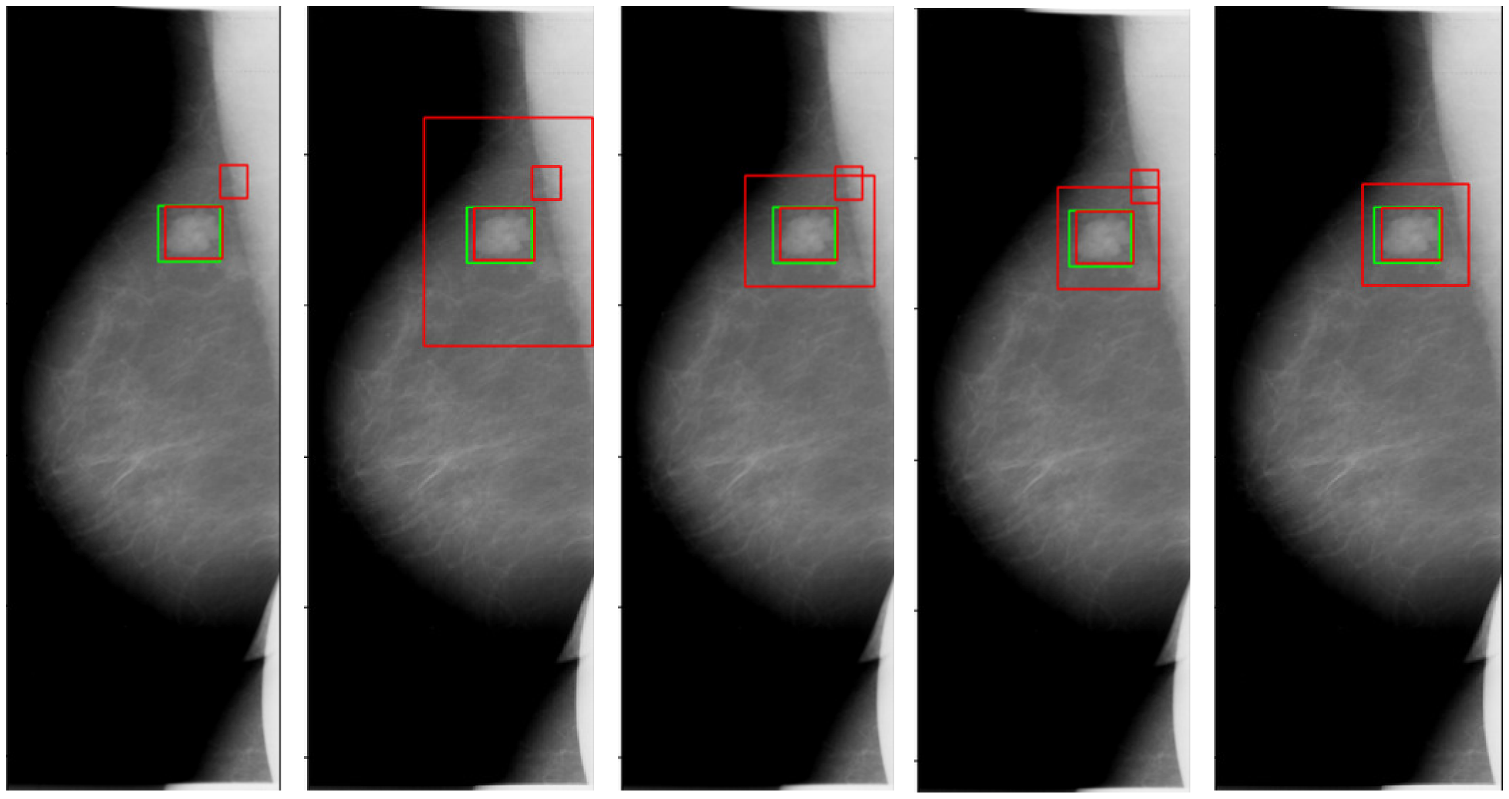

Figure 1.2: Evaluation on different NMS thresholds increasing from left to right 0,0.1,0.25,0.5,1.0

1.6.3 Gradcam visualization:

The gradcam of the image is calculated by taking the gradients of the backbone wrt image and using the heat map to visualize the gradients where the model is focusing on predicting the object or tumor. The heat map is overlapped on the original image to get an even more intuition where the model is focusing; this helps to evaluate the explainability of the model.

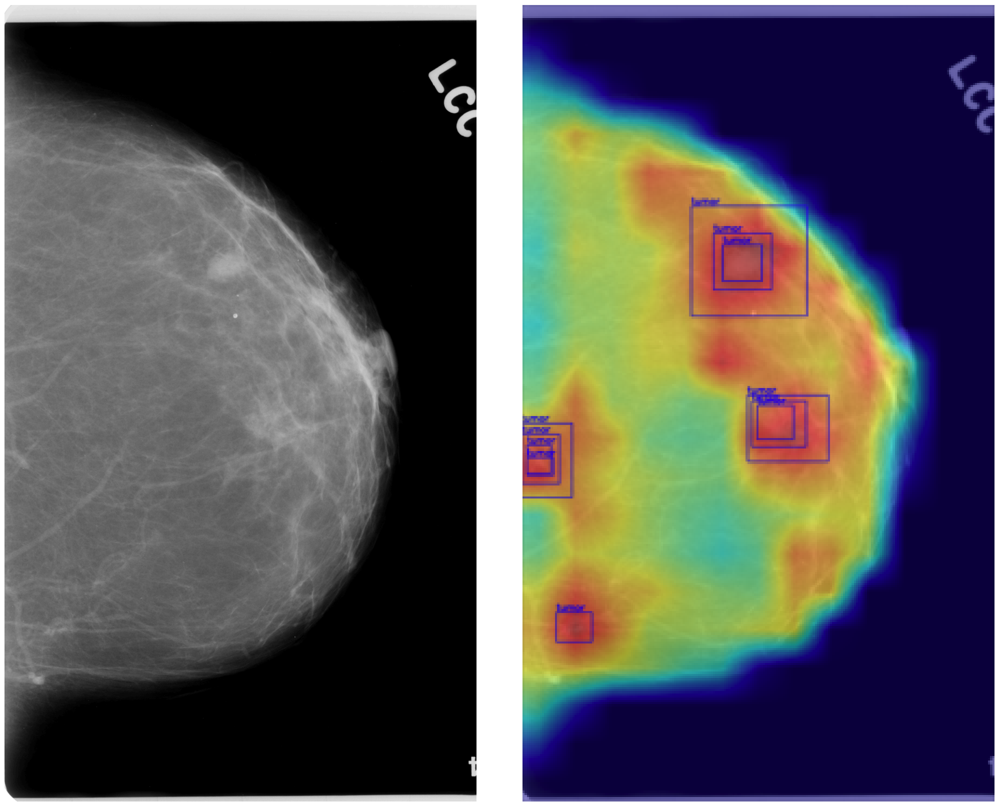

Figure 1.3 The left image is the ground truth right is the gradcam image with tumors predicted.

1.6.4 FROC scores:

The trained model is used for inference. We ran the model on a given test data set, and all results shown below are from the train data set. The provided evaluation metric code is used to evaluate the model. The plots obtained between Sensitivity and threshold value with FPI are shown below:

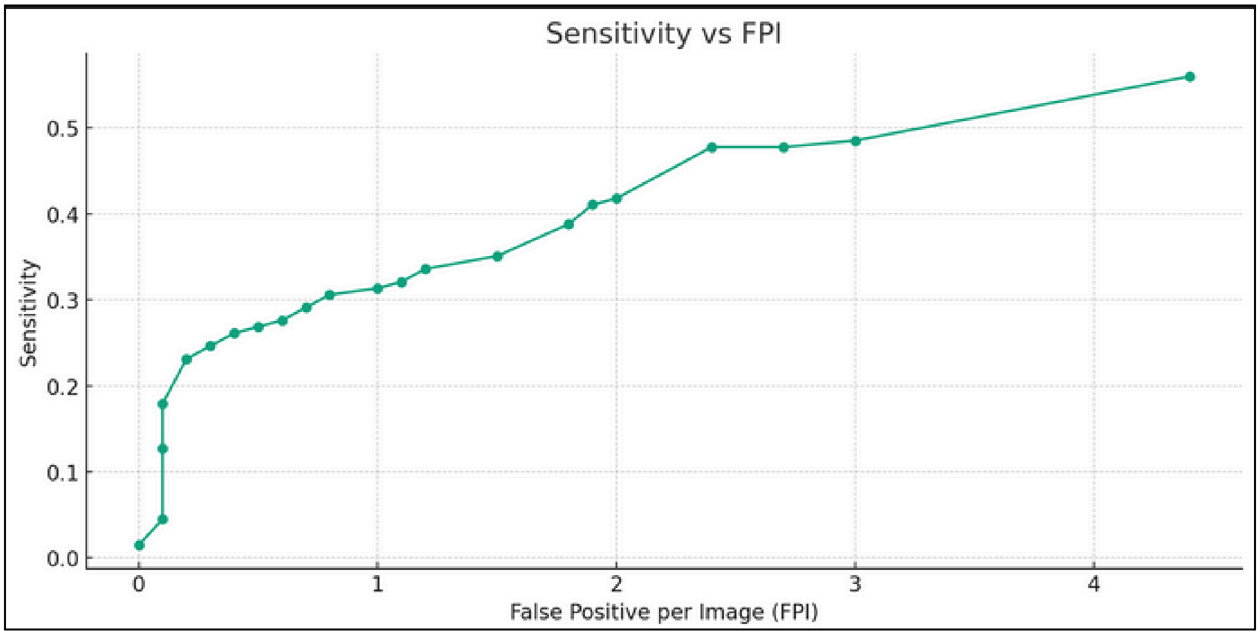

Figure 1.4 Sensitivity vs FPI

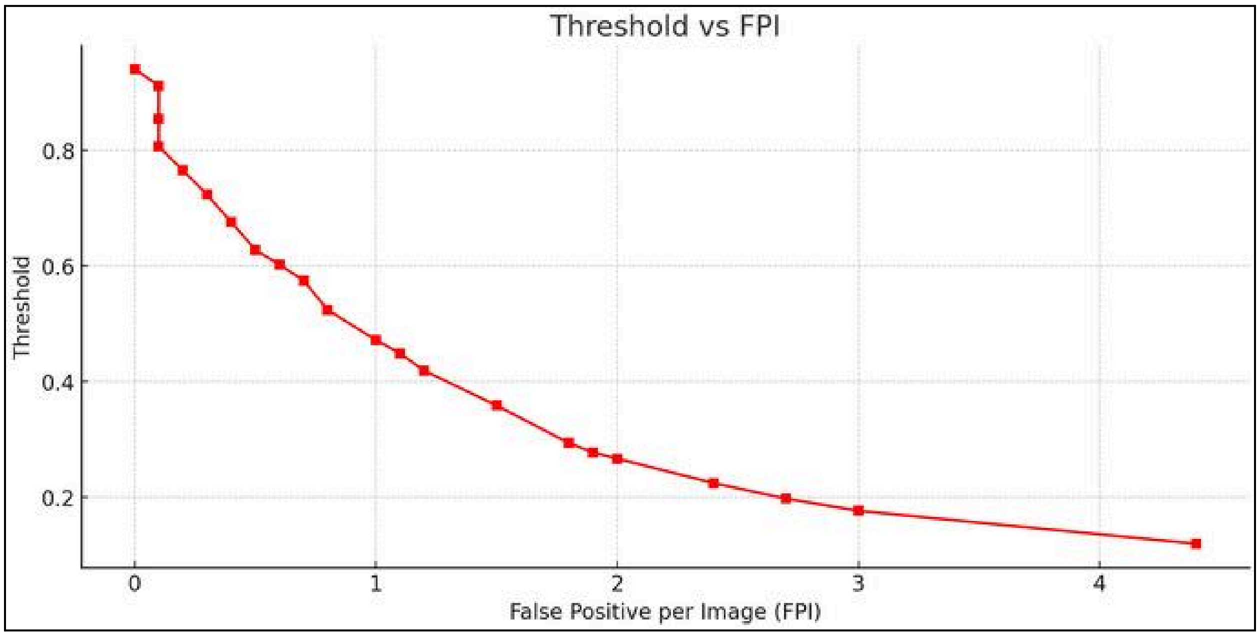

Figure 1.5 Threshold vs FPI

2. transformer-based model for object detection using DeformableDetrForObjectDetection:

Deformable DETR (Deformable Transformers for End-to-End Object Detection) represents a significant advancement in object detection technology, utilizing the strengths of transformer models to enhance detection performance. deformable attention modules allow the model to focus on a subset of key sampling points around the objects instead of the entire image. This selective attention mechanism significantly reduces the computational complexity and improves the model's efficiency.

2.1 Data format used:

We have used the coco data format. The coco annotation data set is used for both models further as it gives more information about the data than the Yolo data format.

2.2 Image preprocessing:

We have used the `DetrImageProcessor` to process the image.the image is reshaped automatically by maintaining the image ratio.

2.3 model used:

We have finally used the pre-trained model of **`SenseTime/deformable-detr and`** fine-tuned the complete model.

2.3.1 BackBone model:

The backbone is used for convolution and normalization to generate the feature map. The Backbone is typically a convolutional neural network (CNN) architecture.

2.4 parameters used:

Optimizer: Adam Optimizer

Learning rate: 1e-4

Weight decay(L2) : 1e-4

Scheduler: stepLR

Stepsize: 1

gamma: 0.8

Number of classes: 2 (background, tumor)

Epochs: 30

2.5 Data loading :

We have used the `torchvision.datasets.CocoDetection` for loading the data. It simplified the loading part and converted the annotated coco data set into a model required format . The collate function is used for padding the all images in batch with max image in batch and mask is generated to differentiate the image and padding the image pixels are masked with 1 and padded pixels are masked with 0. The model requires the pixel_values, pixel_mask and the labels for training the model.

2.6: Results:

2.6.1 loss Curves :

- Train Loss: 1.2
- Validation loss: 1.6

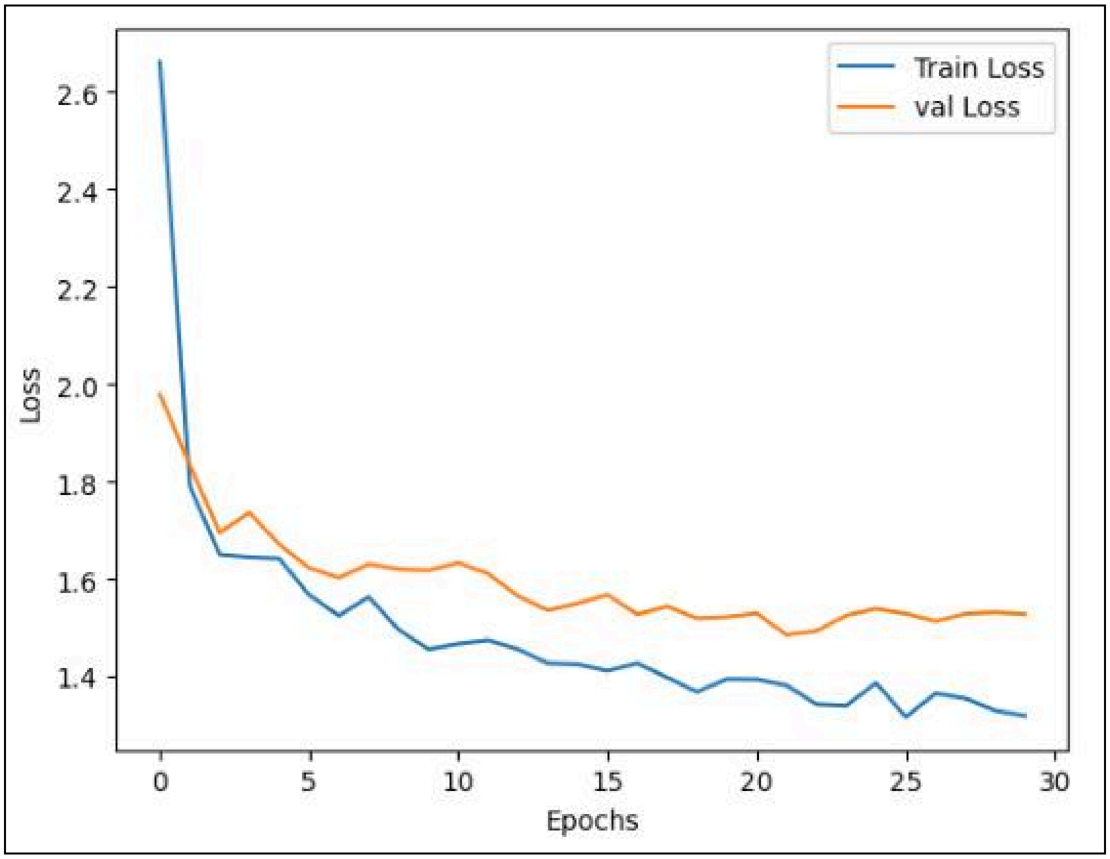

Figure 2.1 train and validation loss vs epochs

2.6.2 output with different Non-Maximum Suppression thresholds:

The model is evaluated on different nms thresholds and the results are shown below in Fig 2.2.

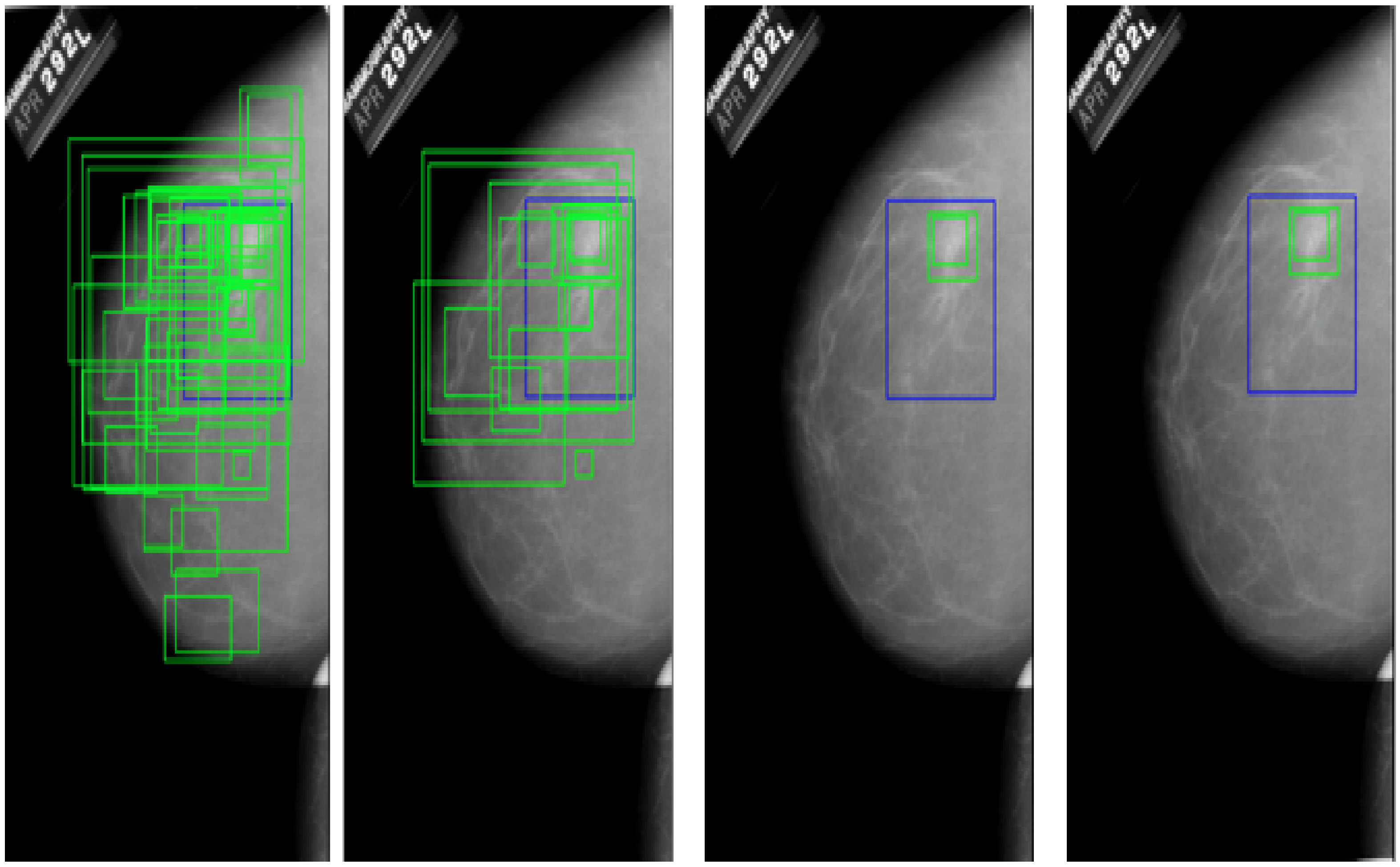

Figure 2.2: Evaluation on different NMS thresholds increasing from left to right 05,0.56,0.58,0.6

2.6.4 FROC scores:

The trained model is used for inference. We ran the model on a given test data set and all results shown below are from the train data set. The provided Evaluation metric code is used for evaluating the model. The plot obtained between sensitivity and threshold wrt to FPI are given below:

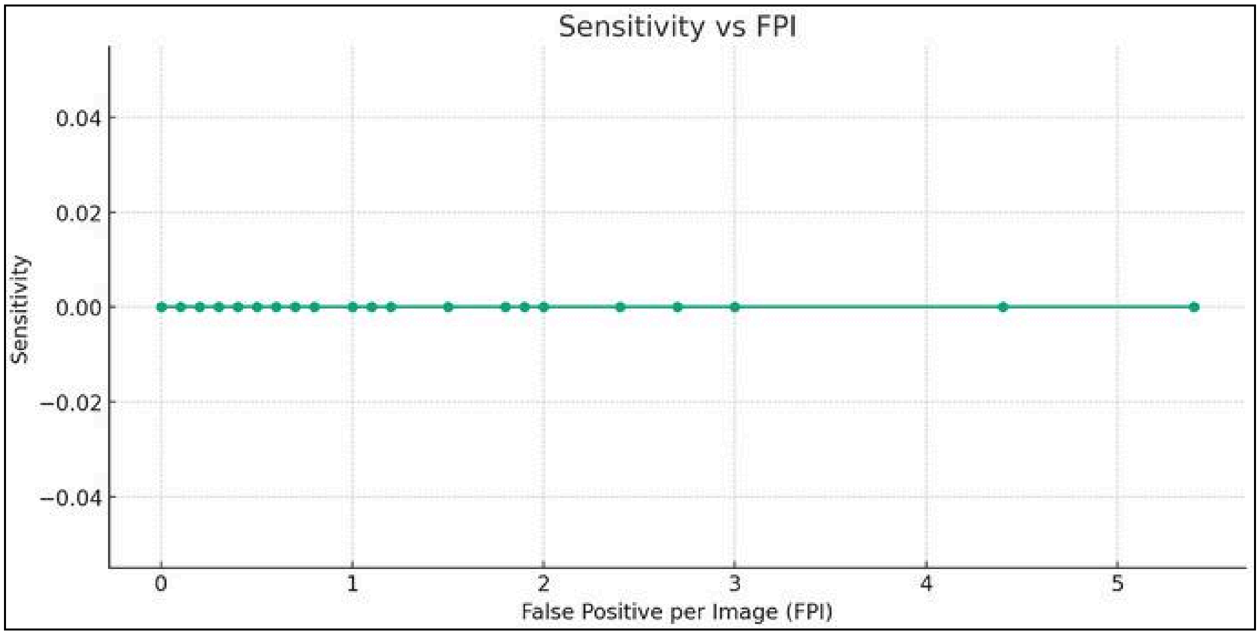

Figure 2.3 Sensitivity vs FPI

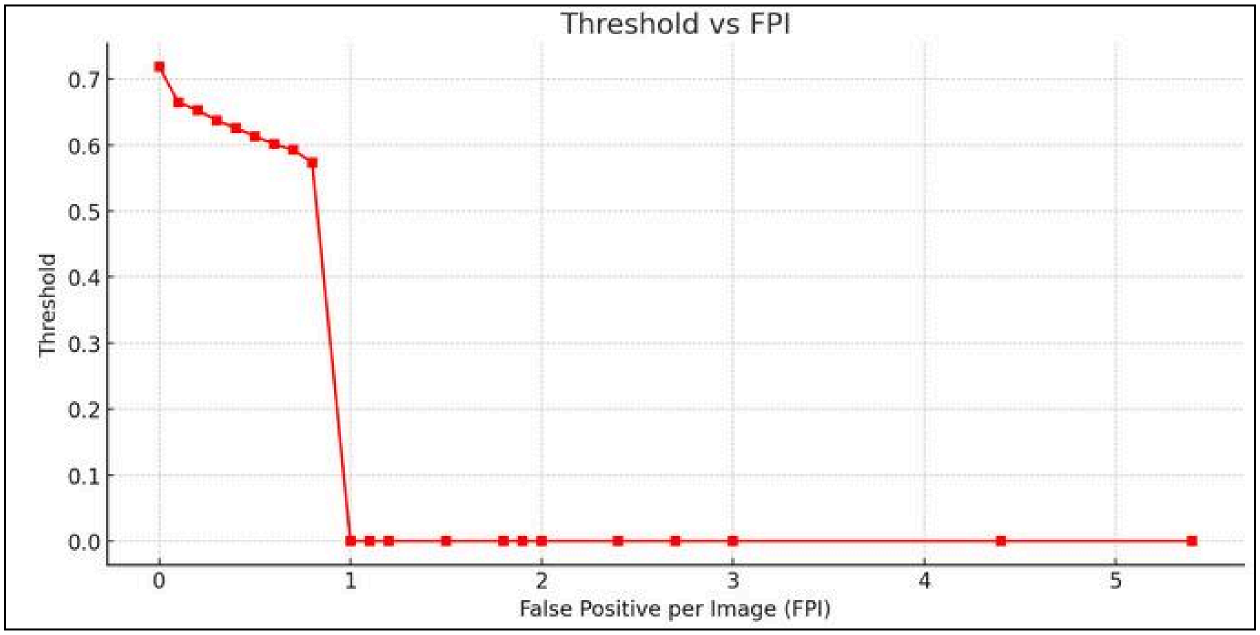

Figure 2.4 Threshold vs FPI

3. Comparison :
For our trained models, we observe that model 1 i,e Faster RCNN performs better compared to model 2 Transformer model by using the FROC evaluation metrics.

4. Guidelines to run the model:

4.1 : train.py :

```
Python3 train.py <model_number> <coco_folder_path>
```

4.2 : test.py :

```
Python3 train.py <model_number> <images_path> <model_path>
```

4.2 : visualize heatmaps.py :

```
Python3 train.py <model_number> <images_path> <model_path>
```

Model_number can be 1 or 2 for all three files.