COL780-Computer Vision

Assignment 4

Deadline: 11:59PM, 27th April 2024

1 Background

Breast cancer is among the most prevalent cancers, and screening mammography plays a crucial role in early detection. Mammograms, which are essentially X-rays of the breast, serve as the primary tool for radiologists to identify malignant tissues. Typically, mammograms are of dimensions 4k×4k, comprising X-rays captured from two perspectives: side-view (MLO-view) and top-view (CC-view).

While Deep Neural Networks (DNNs) have shown promise in **classifying** malignancy in cancers, their integration into medical practice faces challenges, notably in terms of model explainability. Despite achieving commendable results in classification tasks, these models cannot often explain/find the regions contributing to their predictions. To bridge this gap and foster trust among clinicians, it becomes imperative for models to not only classify but also localize cancerous regions within the mammograms. Object detection emerges as a viable solution to this problem, allowing models to draw bounding boxes around areas indicative of tumors.

In this assignment, your objective is to train a deep neural network for object detection in breast cancer screening. You will work with a mammogram detection dataset containing images and corresponding bounding boxes outlining malignant regions.

2 Dataset

The dataset is divided into three splits: training, validation, and testing. For this assignment, you will be provided with the training and validation sets. The dataset is present in link this link, kindly use your IIT-Delhi credentials to access the same.

2.1 COCO Data Format

The COCO dataset consists of the following directories:

- **annotations**: Contains JSON files storing bounding box annotations for both training and validation sets (instances_train.json and instances_val.json).
- **train2017**: Contains training images in JPEG format.
- **val2017**: Contains validation images in PNG format.

The "instances_train.json" file typically follows a hierarchical structure to store bounding boxes, with the following key components:

- Images: This section contains information about each image in the training set, including its unique identifier, file name, height, width, and any other relevant metadata.
- Annotations: This section provides details about the annotations for each object instance present in the training images. Each annotation entry includes the following fields:
  - "id": A unique identifier for the annotation.
  - "image_id": The identifier of the image to which the annotation belongs, linking it to the corresponding image entry in the "images" section.
  - "category_id": The category label of the annotated object.
  - "bbox": The bounding box coordinates of the annotated object, represented as [x, y, width, height]. Here, (x, y) denotes the top-left corner of the bounding box, and width and height represent its dimensions.

2.2 YOLO Data

The YOLO dataset consists of two folders for training and validation, each containing:

- **images**: Contains images for the respective data split.
- **labels**: Contains text files with bounding box annotations for malignant images. Each line in the annotation file represents an object's class label and normalized bounding box coordinates.

YOLO format of bounding box: Each line the annotation file has 5 values, for example: "0 0.25 0.3 0.5 0.4". Here, "0" is the object class label, and the numbers represent the normalized coordinates of the bounding box: (x,y,w,h). These coordinates are normalized to the dimensions of the image, with (x,y) representing the center of the bounding box, and (w,h) representing its width and height.

3 Training Guidelines

- You will train two object detection models: one based on convolution and the other utilizing a transformer architecture. You have the flexibility to choose a convolution-based model such as Faster R-CNN or a YOLO-based model, and for the transformer-based model, you can use DETR-based models like DAB-DETR, Deformable DETR, DINO, etc.
- Utilize any image pre-processing techniques on both the training and validation sets. Ensure your code can perform the same pre-processing during testing.
- You are permitted to use platforms like Kaggle or Google Colab for model training, or any other accessible platform.

4 Submission Guidelines

The assignment must be completed in groups of up to two members. Submit the following files on the portal:

- **model.pt**: Trained model checkpoints for both models.

- **requirements.txt**: File listing the Python libraries used. Submit two files if libraries differ for each model.
- **test.py**: Test script for both models, accepting inputs of an image folder and trained model. Upon execution, the script runs the model on all images in the folder, saving predictions in text files.
- **train.py**: Train script containing dataloader and training loop for both models.
- **visualize_heatmaps.py**: Heatmap visualization script for both models, accepting inputs of an image folder and trained model. Upon execution, the script runs the model on all images in the folder, saving GradCAM/Attention Maps visualizations for the highest confidence bounding box for that image.
- **Report.pdf**: Detailed report covering all experiments and techniques employed in the assignment. Include visualizations before, during, and after training the detection model. The report should contain:
    1. Data visualizations: Distribution information and visualizations of bounding boxes on images.
    2. Effects of image pre-processing techniques.
    3. Training and validation loss curves.
    4. Model predictions on validation set images, pre and post-Non-Maximum Suppression (NMS). Please refer to Figure 1.
    5. Grad-CAM/Attention Maps visualizations of model predictions for malignant images in the validation set.

4.1 Prediction File Format

The test script should save all possible bounding boxes and their confidence scores for each image in a text file. The text file should share the same name as the corresponding image. For instance, if the image is named "Calc-Training_P_00538_RIGHT_MLO.png", the text file should be named "Calc-Training_P_00538_RIGHT_MLO_preds.txt". Each line in the text file represents a bounding box in YOLO format (center_x, center_y, width, height, confidence_score). Refer to the test samples in the shared repository for clarification.

5 Grading Guidelines

Your performance will be evaluated based on competitive performance among your peers and the depth of analysis presented in your report. Marks will be assigned according to the rank and performance of your model, assessed on two metrics. One metric will be kept hidden, while the other is described below:

FROC (Free-response Receiver Operating Characteristic) Curve: Similar to the ROC curve, FROC plots sensitivity against average false positives per image (FPI) on the x-axis. A prediction is considered a true positive if the center of the bounding box lies within the center of the ground truth box. False positives are calculated based on all predicted bounding boxes in an image. The final rank will be determined by a weighted average of both metrics.

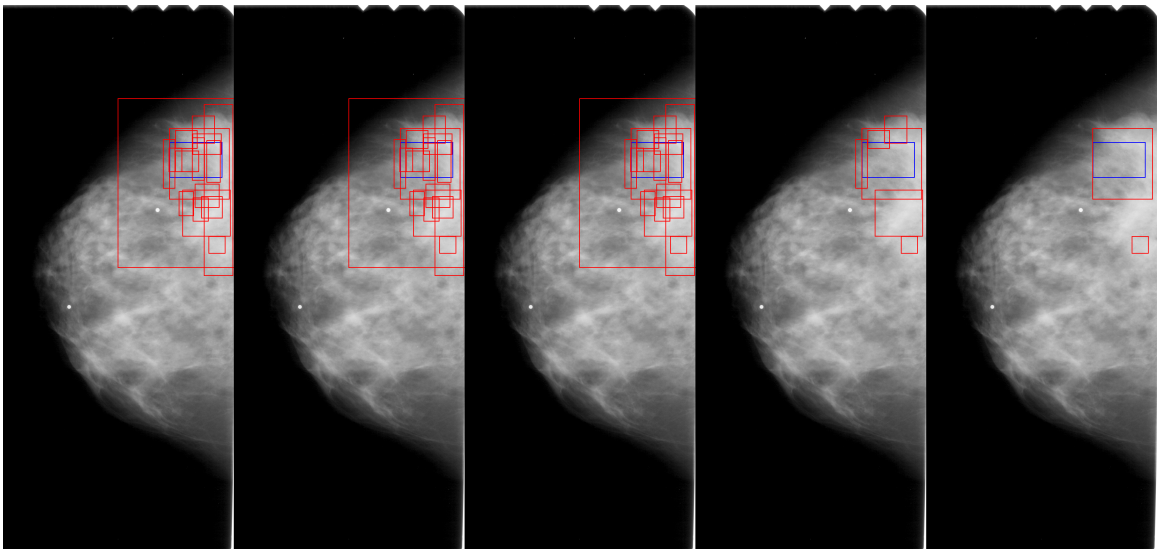

Figure 1: The illustration depicts the fluctuation of Non-Maximum Suppression (NMS) across the predictions generated by a detection module. The blue bounding box signifies the ground truth for a malignant tumor, while the red bounding boxes denote predictions made by the detection model. Within this visual representation, we observe the nuanced variations in NMS. We explore five Intersections over Union (IOU) thresholds: 1, 0.5, 0.25, 0.1, and 0. A threshold of 1 indicates the inclusion of a bounding box regardless of overlaps, while a threshold of 0 signifies the removal of bounding boxes even in the presence of slight overlaps. Please refer to this image when visualizing your bounding boxes for the assignment.

A sample test set is provided in the shared repository, comprising three folders: images, labels, and predictions. The FROC.py script in the data folder takes inputs from labels and predictions folders to evaluate your model on the validation set. It calculates sensitivity values at different FPI thresholds and prints corresponding threshold values.

In your report, compare the performance of the two trained models. Set a threshold corresponding to the same FPI and compare which model performs better. For instance, if both models achieve a sensitivity of 0.6 and 0.7 at FPI=0.3, identify the additional cases detected by the second model.

6 Do's and Don'ts

- Utilize the provided FROC metric code for evaluation; refrain from implementing it.
- You may refer to open-source libraries for generating GradCAMs/Attention maps and applying NMS on model predictions.
- Adhere strictly to the institute's plagiarism policy.
- Ensure code reproducibility.
- Follow specified formats to avoid a penalty of 10% for each mistake.

7 Tips and Tricks

- Resize image data to lower resolutions for faster training, but be mindful of potential loss of information.
- Employ transfer learning by initializing models with pre-trained weights for better convergence and performance.
- Consider freezing some layers of the detection model to accelerate training and enhance performance.
- Start the assignment early if training on the cloud, as there may be weekly limits on training time.
- Implement Non-Maximum Suppression (NMS) on predictions for improved performance.
- Utilize visualizations extensively. Verify data reading accuracy by visualizing bounding boxes on images. Verify prediction format correctness by visualizing saved predictions in text files.