CS 771 - Computer Vision
Assignment - 3

Team Members: Sujay Chandra Shekara Sharma, Jiangyi Liu, Venkata Abhijeeth Balabhadruni

Wisc Emails:
schandrashe5@wisc.edu, jiangyi.liu@wisc.edu, balabhadruni@wisc.edu

4.1 Understanding FCOS

1 What is the output of the backbone (i.e., the input to FPN)?

The output of the backbone consists of hierarchical feature maps that capture spatial and semantic information from the input image at multiple scales. These feature maps are critical for the subsequent Feature Pyramid Network (FPN) and object detection stages. The backbone extracts the following:

- *Low-level features:* Basic edges, textures, and patterns.
- *High-level features:* Semantic representations, complex patterns, and contextual information.

Each feature map F_i is represented as $\mathbb{R}^{H \times W \times C}$, where:

- H and W: Height and width of the feature map.
- C: Number of channels in the feature map.

These multi-scale feature maps are passed into the FPN, which enhances the representation of objects at varying scales.

2 How many levels are there in the FPN output? And how does the FPN generate these output feature maps?

The Feature Pyramid Network (FPN) generates a hierarchical set of feature maps with multiple levels, typically denoted as $\{P_3, P_4, P_5, P_6, P_7\}$. These levels correspond to progressively coarser resolutions, enabling the model to detect objects at various scales. In total, the FPN produces five levels of output feature maps. The levels are:

- P_3: High-resolution feature map for detecting small objects.
- P_4, P_5: Intermediate-resolution feature maps for medium-sized objects.
- P_6, P_7: Low-resolution feature maps for detecting large objects.

How FPN Generates Output Feature Maps : The FPN uses a top-down architecture with lateral connections to combine semantic and spatial information across multiple scales. The process involves the following steps:

1. The backbone network (e.g., ResNet-18) extracts multi-scale feature maps from different stages of the network, typically referred to as C_3, C_4, C_5. These correspond to feature maps from intermediate layers of the backbone.

2. Starting from the highest-resolution feature map C_5, the FPN performs upsampling by a factor of 2 (using nearest-neighbor interpolation). This upsampled feature map is combined with the corresponding feature map from the bottom-up pathway (e.g., C_4) via lateral connections.

3. Each lateral connection applies a 1×1 convolution to the bottom-up feature map to align the number of channels with the top-down feature map before adding them.

4. After merging the upsampled feature map and the laterally connected feature map, a 3×3 convolution is applied to reduce aliasing effects and refine the feature representation.

5. The P_6 feature map is obtained by applying a 3×3 convolution with a stride of 2 on P_5. Similarly, P_7 is generated by applying another 3×3 convolution with stride 2 on P_6.

These multi-scale feature maps (P_3, P_4, P_5, P_6, P_7) enable the model to effectively detect objects of varying sizes by combining fine-grained details from shallow layers with semantic-rich representations from deeper layers.

3 How does FCOS assign positive/negative samples during training? What are the loss functions used in the training?

In FCOS, the assignment of positive and negative samples is determined based on the location of a feature point relative to the ground-truth bounding boxes.

A feature point is assigned as a positive sample if it lies within a ground-truth bounding box and its location satisfies the center sampling condition, which restricts positive samples to regions near the center of the object to avoid ambiguous assignments in overlapping regions. The center region is defined by a factor t of the bounding box size.

The feature point is associated with the smallest bounding box that encompasses it, based on the area of the bounding box.

Feature points that do not fall within any ground-truth bounding box or fail the center sampling condition are treated as negative samples.

Loss Functions in FCOS

There are three loss components used in implementation: **classification loss**, **bounding box regression loss** and **Center-ness loss**. The classification loss is based on *focal loss*, borrowed from RetinaNet. The total loss function for FCOS is defined as:

$$\begin{aligned}\mathcal{L}(\{p_{x,y}\},\{t_{x,y}\}) = &\frac{1}{N_{\text{pos}}}\sum_{x,y}\mathcal{L}_{\text{cls}}(p_{x,y},c^*_{x,y})\\ &+\lambda\frac{1}{N_{\text{pos}}}\sum_{x,y}\mathbf{1}\{c^*_{x,y}>0\}\mathcal{L}_{\text{reg}}(t_{x,y},t^*_{x,y})\\ &+\frac{1}{N_{\text{pos}}}\sum_{x,y}\mathbb{1}_{\{c^*_{x,y}>0\}}\mathcal{L}_{\text{center}}.\end{aligned}$$

where:

- $\mathcal{L}_{\text{cls}}$: *Focal loss*.
- $\mathcal{L}_{\text{reg}}$: *Generalized IoU (GIoU) loss*.
- $\mathcal{L}_{\text{center}}$: *Center-ness loss*.
- N_{pos}: The total number of positive samples.
- λ: A balance weight for the regression loss $\mathcal{L}_{\text{reg}}$, set to 1 in this work.
- $\mathbf{1}\{c^*_{x,y}>0\}$: An indicator function that is 1 for positive samples and 0 for negative samples.

Explanation of Loss Components

- **Focal Loss:** The focal loss helps focus training on hard-to-classify examples by down-weighting easy negative samples. It is given by:

  $$\text{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t),$$

  where p_t is the predicted probability of the correct class. The parameter γ controls the degree of focus on hard examples, with larger values increasing the emphasis on difficult cases.

- **Generalized IoU (GIoU) Loss:**

  The Generalized IoU (GIoU) loss is computed between the predicted bounding box B_p and the ground-truth bounding box B_g. It accounts for the overlap as well as the relative placement of the boxes.

  - The IoU is calculated as:

    $$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}.$$

  - The Generalized IoU (GIoU) incorporates the smallest enclosing box B_c that contains both B_p and B_g, defined as:

    $$\text{GIoU} = \text{IoU} - \frac{\text{Area of Enclosing Box} - \text{Area of Union}}{\text{Area of Enclosing Box}}.$$

– The GIoU loss is then computed as:

$$L_{\text{GIoU}} = 1 - \text{GIoU}.$$

- **Center-ness Loss:**

  Uses Binary Cross Entropy loss.

4 How does FCOS decode objects at inference? What are the necessary post-processing steps (e.g., non-maximum suppression)?

At inference, FCOS decodes objects by passing the input image through the network, where the model generates classification scores $p_{x,y}$ and bounding box regression predictions $t_{x,y} = (l, t, r, b)$ for each location on the feature map. A location is considered a positive sample if the classification score $p_{x,y}$ exceeds a threshold of 0.05, indicating a high confidence that an object is present at that location.

Once a positive sample is identified, the predicted bounding box is computed using the regression values and the inverse of the formula:

$$l = \frac{x - x_0^{(i)}}{s}, \quad t = \frac{y - y_0^{(i)}}{s}, \quad r = \frac{x_1^{(i)} - x}{s}, \quad b = \frac{y_1^{(i)} - y}{s}$$

where $x_0^{(i)}, y_0^{(i)}$ and $x_1^{(i)}, y_1^{(i)}$ represent the coordinates of the bounding box corners, and s is the stride associated with the feature map.

For post-processing, **Non-Maximum Suppression (NMS)** is applied to eliminate duplicate or overlapping bounding boxes. The NMS algorithm sorts the bounding boxes by their classification scores and suppresses boxes that overlap significantly with higher-scoring boxes, retaining only the most confident, non-overlapping predictions. In FCOS, the NMS threshold is set to 0.6, which is a higher value compared to RetinaNet's 0.5 threshold. This threshold adjustment helps to reduce false positives by removing less accurate bounding boxes.

4.2 Model Inference

1. Classification and Regression Heads

The forward passes in the `FCOSClassificationHead` and `FCOSRegressionHead` process the feature maps from the feature pyramid to predict classification scores, bounding box regressions, and center-ness scores.

1. **Input:** A list of feature maps (`x`) from the feature pyramid.
2. **Classification Head:**
   - Each feature map is passed through multiple 3×3 convolutional layers, followed by GroupNorm and ReLU activation.
   - The processed feature maps are passed through a classification layer to predict class logits.

- The outputs are reshaped from (N, C, H, W) to $(N, H \times W, C)$, where N is the batch size, C is the number of classes, and H, W are spatial dimensions.

3. **Regression Head:**
   - Each feature map is processed through similar convolutional layers as in the classification head.
   - **Bounding Box Regression:**
     - A regression layer predicts offsets (l, t, r, b) from the feature map locations to the bounding box edges.
     - Outputs are reshaped from $(N, 4, H, W)$ to $(N, H \times W, 4)$.
   - **Center-ness Prediction:**
     - A separate layer predicts the center-ness score for each location.
     - Outputs are reshaped from $(N, 1, H, W)$ to $(N, H \times W, 1)$.
4. **Output:**
   - The classification head outputs a list of class logits for each feature map.
   - The regression head outputs two lists for each feature map: bounding box offsets and center-ness scores.

These forward passes enable FCOS to efficiently predict classes, bounding box regressions, and confidence scores at each feature map location.

2. Decoding the Objects

The inference function performs object detection on input images using a Feature Pyramid Network (FPN)-based model. The steps are outlined as follows:

1. **Setup:**
   - The function operates on a batch of images.
   - For each feature pyramid level, sigmoid activation is applied to classification and centerness logits to convert them into probabilities.
2. **Processing Each Image:**
   - The function iterates over all images in the batch.
   - For each feature pyramid level, it performs the following operations:
     (a) *Object Scoring:* Computes objectness scores by combining classification and centerness probabilities. Only candidates with scores above a predefined threshold are retained.
     (b) *Candidate Selection:* Extracts the corresponding class labels, regression outputs (bounding box deltas), and anchor points for retained candidates. If the number of candidates exceeds a set limit (`topk_candidates`), only the top-scoring candidates are kept.

(c) *Bounding Box Prediction:* Converts regression outputs into bounding box coordinates relative to anchor points and the stride of the current FPN level. Predicted boxes are clipped to image boundaries to prevent out-of-bound detections.

(d) *Filtering:* Removes boxes that have zero or negative area.

3. **Aggregation Across Levels:**
   - Candidate bounding boxes, scores, and labels from all FPN levels are concatenated.
4. **Post-Processing:**
   - *Non-Maximum Suppression (NMS):* Eliminates duplicate and overlapping bounding boxes based on a threshold.
   - *Top-K Detections:* Selects the highest-scoring detections up to a limit (`detections_per_img`).
5. **Output:**
   - For each image, the function returns a dictionary containing:
     - `boxes`: Final bounding box coordinates.
     - `scores`: Objectness scores for the boxes.
     - `labels`: Predicted class labels.

This inference process ensures efficient multi-scale object detection while filtering and refining the predictions to achieve high-quality results.

Figure 1 shows our mAP scores, matching the expected mAP@0.5 of 60.9%. The total inference time is 150.65 seconds.

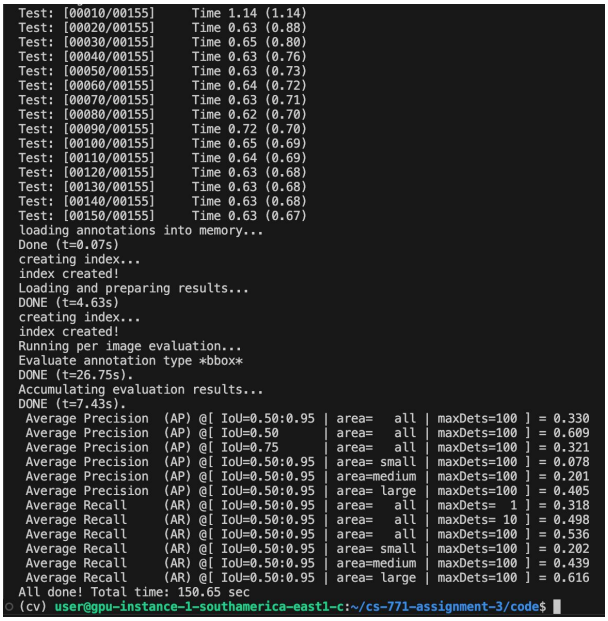
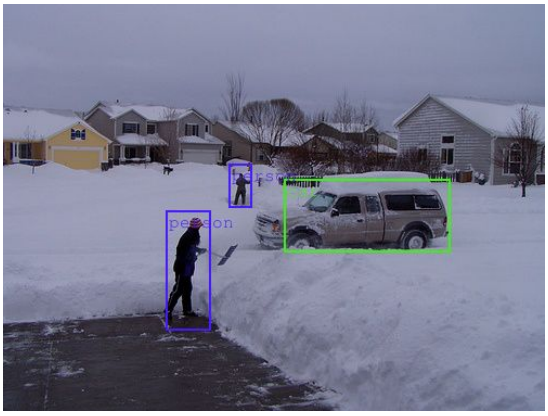


Figure 1: mAP Scores and Inference Time for Pre-trained model

Figure 2 shows some sample results with object detection.

(a) Single Object in Image



(b) Multiple Objects in Image

Figure 2: Detection results with pretrained FCOS model.

4.3 Model Training

The `compute_loss` function calculates the total loss for training the FCOS model by combining three components:

1. **Classification Loss**: Measures the error in class predictions using sigmoid focal loss. This helps balance the contribution of positive and negative samples.
2. **Regression Loss**: Uses Generalized Intersection over Union (GIoU) loss to quantify the difference between predicted and ground truth bounding boxes for valid anchor points.
3. **Centerness Loss**: Computes binary cross-entropy loss to prioritize anchor points closer to the center of objects, as these are more reliable for regression.

Key Steps in the Function:

- **Preprocessing Targets**: Ground truth bounding boxes, labels, and areas are processed to align with the anchor points generated across all feature pyramid levels.
- **Point Matching**: Boolean masks (`is_in_box`, `is_in_center`, `is_in_reg_range`) are used to filter valid anchor points based on their location and regression range.
- **Regression Targets**: Computes normalized distances from anchor points to bounding box edges for valid points.
- **Loss Computation**:
  - Classification loss is calculated using sigmoid focal loss.
  - Regression loss is computed using GIoU for valid points.
  - Centerness loss is calculated using binary cross-entropy to emphasize points near object centers.
- **Loss Aggregation**: The final loss is the sum of classification, regression, and centerness losses, averaged over the foreground (valid) anchor points.

The function outputs a dictionary containing:

- `cls_loss`: Classification loss.
- `reg_loss`: Regression loss.
- `ctr_loss`: Centerness loss.
- `final_loss`: Combined total loss for backpropagation.

ResNet - 18

For the default model with the following config:

- **Backbone:** Resnet-18
- **Epochs:** 10
- **Learning rate:** 0.005

Figure 3, 4 and 5 shows the training curves of the default model.

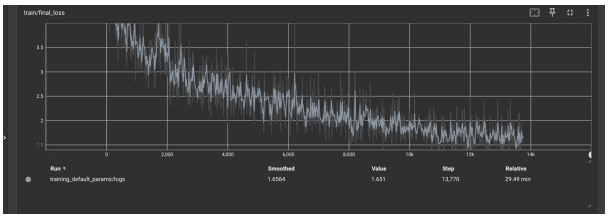


Figure 3: Final Loss

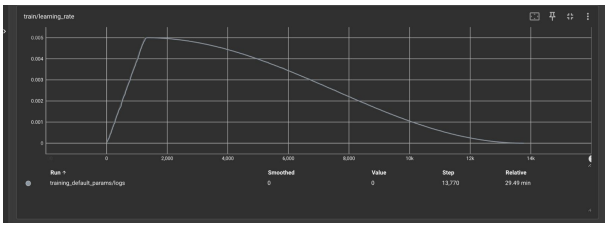


Figure 4: Learning Rate

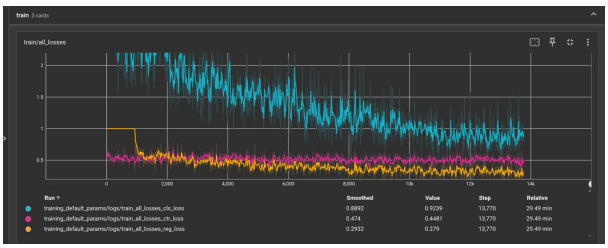


Figure 5: All Losses

For the default model we got the following mAP score : mAP@0.5 is 43.6%. Figure 6 shows the testing mAP scores for the default model.

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.195
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.436
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.140
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.014
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.087
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.267
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.235
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.363
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.391
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.057
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.264
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.493
All done! Total time: 161.18 sec
```

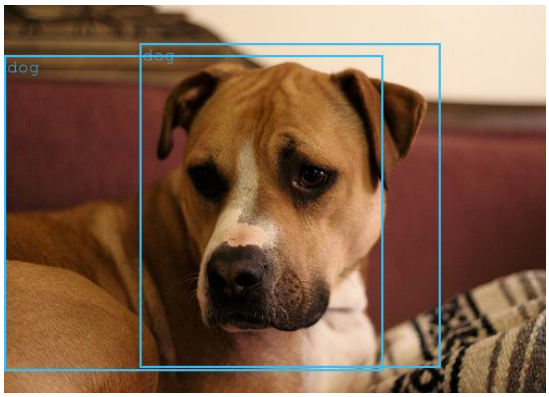
Figure 6: mAP scores for the default model

Figure 7 shows test times for some of the mini batches.

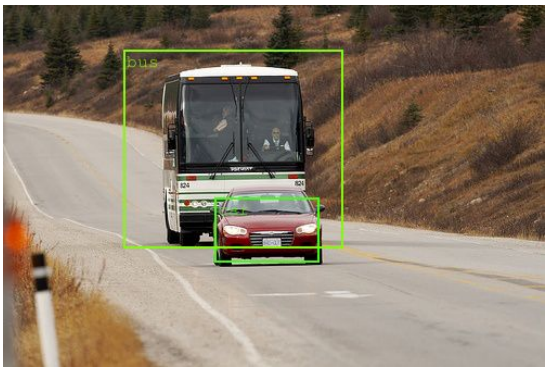
```
  warnings.warn(
Test: [00010/01238]     Time 0.17 (0.17)
Test: [00020/01238]     Time 0.11 (0.14)
Test: [00030/01238]     Time 0.10 (0.13)
Test: [00040/01238]     Time 0.10 (0.12)
Test: [00050/01238]     Time 0.10 (0.12)
Test: [00060/01238]     Time 0.10 (0.11)
Test: [00070/01238]     Time 0.11 (0.11)
Test: [00080/01238]     Time 0.10 (0.11)
Test: [00090/01238]     Time 0.10 (0.11)
Test: [00100/01238]     Time 0.09 (0.11)
Test: [00110/01238]     Time 0.10 (0.11)
Test: [00120/01238]     Time 0.12 (0.11)
Test: [00130/01238]     Time 0.10 (0.11)
Test: [00140/01238]     Time 0.10 (0.11)
Test: [00150/01238]     Time 0.10 (0.11)
Test: [00160/01238]     Time 0.10 (0.11)
Test: [00170/01238]     Time 0.09 (0.10)
```

Figure 7: Test time for some mini batches with default model

Figure 8 shows some samples object detection results on the test set.



(a) Single Object in Image



(b) Multiple Objects in Image

Figure 8: Detection results with default model.

ResNet - 34

We experimented with hyperparameters and found the following model gave better results. The following is the improved model's configuration:

- **Backbone:** Resnet-34
- **Epochs:** 12
- **Learning rate:** 0.005

Figure 9, 10 and 11 shows the training curves for the ResNet-34 model.

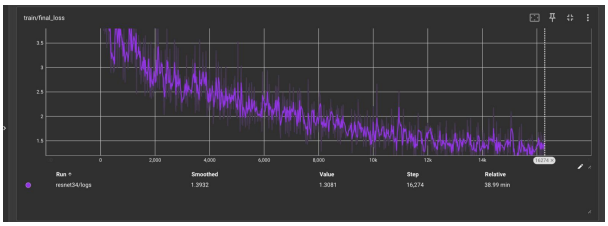


Figure 9: Final Loss

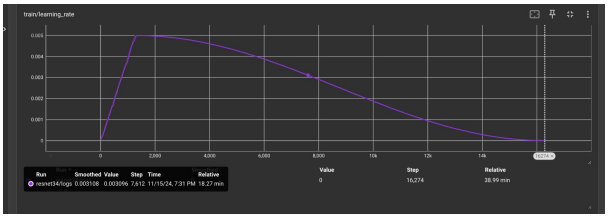


Figure 10: Learning Rate

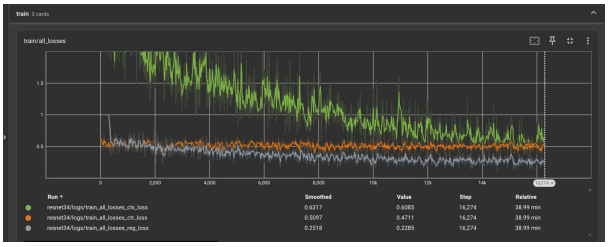


Figure 11: All Losses

For the improved model we got the following mAP score : mAP@0.5 is 53.4%. Almost a 10% increase in score from the default model. Figure 12 shows the testing mAP scores for the improved model.

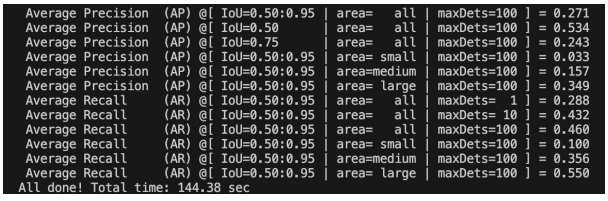


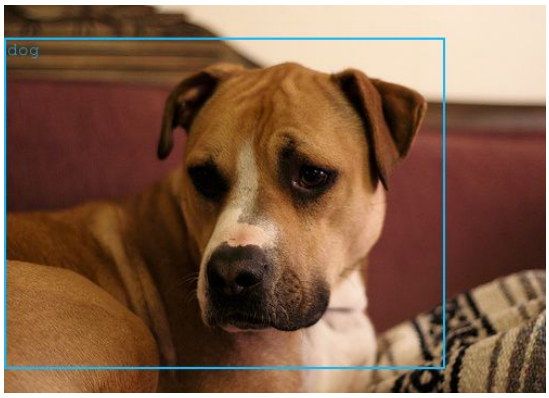
Figure 12: mAP scores for the improved model

Figure 13 shows test times for some of the mini batches.

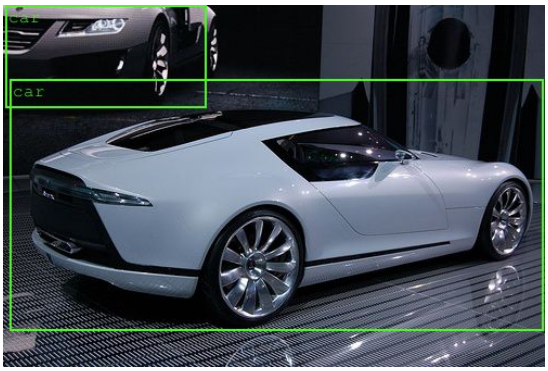
```
Test: [00010/01238]     Time 0.20 (0.20)
Test: [00020/01238]     Time 0.08 (0.14)
Test: [00030/01238]     Time 0.09 (0.12)
Test: [00040/01238]     Time 0.08 (0.11)
Test: [00050/01238]     Time 0.09 (0.11)
Test: [00060/01238]     Time 0.08 (0.10)
Test: [00070/01238]     Time 0.08 (0.10)
Test: [00080/01238]     Time 0.08 (0.10)
Test: [00090/01238]     Time 0.09 (0.10)
Test: [00100/01238]     Time 0.08 (0.10)
Test: [00110/01238]     Time 0.10 (0.10)
Test: [00120/01238]     Time 0.08 (0.10)
Test: [00130/01238]     Time 0.09 (0.09)
Test: [00140/01238]     Time 0.08 (0.09)
Test: [00150/01238]     Time 0.08 (0.09)
Test: [00160/01238]     Time 0.08 (0.09)
Test: [00170/01238]     Time 0.08 (0.09)
Test: [00180/01238]     Time 0.08 (0.09)
Test: [00190/01238]     Time 0.09 (0.09)
Test: [00200/01238]     Time 0.08 (0.09)
Test: [00210/01238]     Time 0.09 (0.09)
Test: [00220/01238]     Time 0.08 (0.09)
```

Figure 13: Test time for some mini batches with improved model

Figure 14 shows some samples object detection results on the test set.



(a) Single Object in Image



(b) Multiple Objects in Image

Figure 14: Detection results with improved model.

Name	Contributed Sections
Sujay Chandra Shekara Sharma	Understanding FCOS paper, 4.1, 4.3
Jiangyi Liu	Understanding FCOS paper, 4.2
Venkata Abhijeeth Balabhadruni	Understanding FCOS paper, 4.1, 4.3

Table 1: Contributions of each member