# BMI / CS 771: Homework Assignment 3 Report

Qinxinghao Chen Handan Hu Chongwei Liu Bohan Wen

November 15, 2025

## 1 Team Contributions

- **Qinxinghao Chen**: Responsible for the model implementation and training on VOC dataset.
- **Handan Hu**: Responsible for training and evaluating on COCO dataset.
- **Chongwei Liu**: Responsible for the model inference.
- **Bohan Wen**: Responsible for final model training data processing and report writing.

## 2 Implementation Details and Challenges

During the implementation of the FCOS model, we encountered several key technical challenges, which we resolved through a series of strategic adjustments and debugging:

1. **Training Stability**: In our initial training attempts, we observed gradient explosion, which caused the loss to become `NaN`. We successfully suppressed the `NaN` values by adopting a **lower initial learning rate (0.01)**, ensuring a stable training launch.
2. **Center-ness Loss Convergence**: We found that the *ctr_loss* converged much more slowly than other losses (see Figure 4). To address this, we adjusted the loss function by doubling its weight ($final\_loss = cls\_loss + reg\_loss + 2 \times ctr\_loss$), thereby forcing the model to pay more attention to the quality of predicted box centers.
3. **Inference Top-k Logic**: In the `inference` function, we discovered that one cannot simply flatten all predictions and take the global top-k. Instead, we modified the logic to select candidates **per-class**, which ensured that rare categories could be correctly detected.
4. **Coordinate Format Confusion**: We experienced confusion between the $(x, y)$ and $(y, x)$ formats when handling `points`, which led to significant debugging difficulties. We spent extra time unifying the coordinate system to resolve this issue.
5. **Weight Initialization**: To aid the convergence of *ctr_loss*, we applied specific initializations to the bias terms in the `RegressionHead` (e.g., initializing `bbox_ctrness.bias` to 0.0).

## 3 Model Inference

We first validated the correctness of our inference code using the provided pre-trained model (`voc_res18.pth.tar`).

- **mAP Score**: Our implementation achieved an mAP@0.5 of **60.1%**, which is consistent with the assignment's benchmark of 60.9%.
- **Inference Efficiency**: With a batch size of 32, the average inference time was approximately 1.16 seconds per batch, and the total evaluation time was 191.19 seconds.
- **Sample Detection Results**: As shown in Figures 1 and 2, the pre-trained model demonstrated strong detection capabilities, accurately identifying and localizing objects of various classes, such as horses, people, and cars.


Figure 1: Pre-trained model detection result.


Figure 2: Pre-trained model detection result.

# 4 Model Training

After validating our inference code, we trained our own FCOS model on the VOC 2007 dataset using a `ResNet-18` backbone.

## 4.1 VOC Dataset

### 4.1.1 Key Parameters & Training Efficiency

**Baseline (as reported).**

- **Configuration**: `ResNet-18`, 10 epochs, Learning Rate: 0.01.
- **Efficiency**: We completed the training using a **single T4 GPU** in **25 minutes**. This is "well under" the 30-minute reference standard mentioned in the assignment, proving that our `compute_loss` implementation is efficient.

**Extended Training on Top of the Baseline.** On top of the above baseline setting, we further trained with a lower learning rate and more epochs while keeping all other hyperparameters unchanged unless noted:

- **Configuration**: `ResNet-18`, 50 epochs, Learning Rate: $8 \times 10^{-4}$ (`0.0008`).
- **Results**:

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=5.87s).
Accumulating evaluation results...
DONE (t=1.45s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.313
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.574
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.307
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.044
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.157
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.406
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.309
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.424
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.431
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.082
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.270
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.534
All done! Total time: 124.10 sec
```

Figure 3: Test result on extended model.

### 4.1.2 Training Curves

As shown in Figure 3, the total loss steadily decreased while the mAP on the test set steadily increased. As shown in Figure 4, the component losses (cls_loss, reg_loss) decreased significantly, while the ctr_loss decreased more slowly, validating our decision to increase its weight.

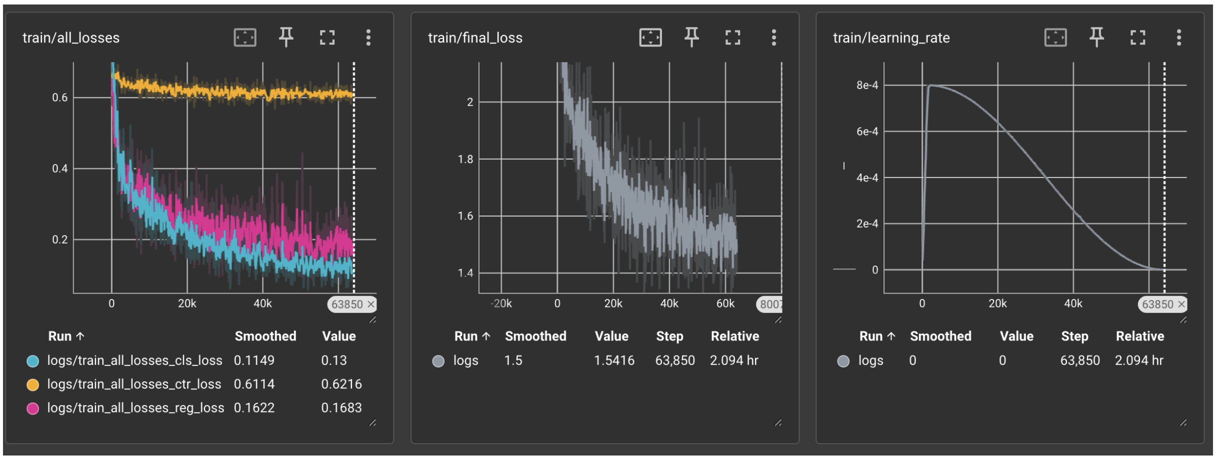


Figure 4: Tensorboard result

### 4.1.3 Final mAP Score (Testing mAP)

Our final model achieved an mAP@0.5 of **57.4%** on the test set.

### 4.1.4 Sample Detection Results

As shown in Figures 5 and 6, our model is capable of accurately detecting and localizing multiple objects in an image, such as people, dogs, sofas, and dining tables.

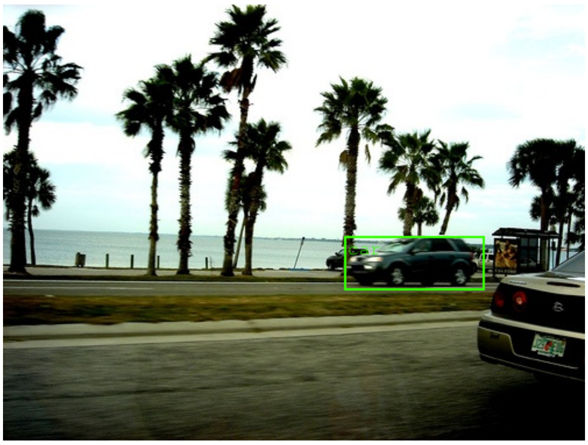

Figure 5: Our final model's detection result.


Figure 6: Our final model's detection result.

## 4.2 COCO Dataset

### 4.2.1 Implementation Description

- **Dataset** (`libs/dataset.py`):

– Added `COCODetection` class
– Implemented `get_cls_names()` returning 80 COCO class names
– Modified `build_dataset()` to handle COCO's directory structure and annotation format.

- **Configuration** (`configs/coco_fcos.yaml`):
  – Updated paths of images and annotations
  – Used ResNet-18 as the lightweight backbone
  – Set training to 4 epochs with SGD optimizer and learning rate 0.0005
  – Set num_classes: 80

#### 4.2.2 Experimental Procedures

1. **Data Preparation:** Downloaded COCO 2017 using `download_coco.sh` and extracted files into data/coco/, including training, validation, test images and annotations.

2. **Training Process:** Train the model using `configs/coco_fcos.yaml`. Training curves show convergence: classification and regression losses decrease from $\tilde{0}.7$ to $\tilde{0}.3$–0.4, while center-ness loss stabilizes around 0.6–0.7.

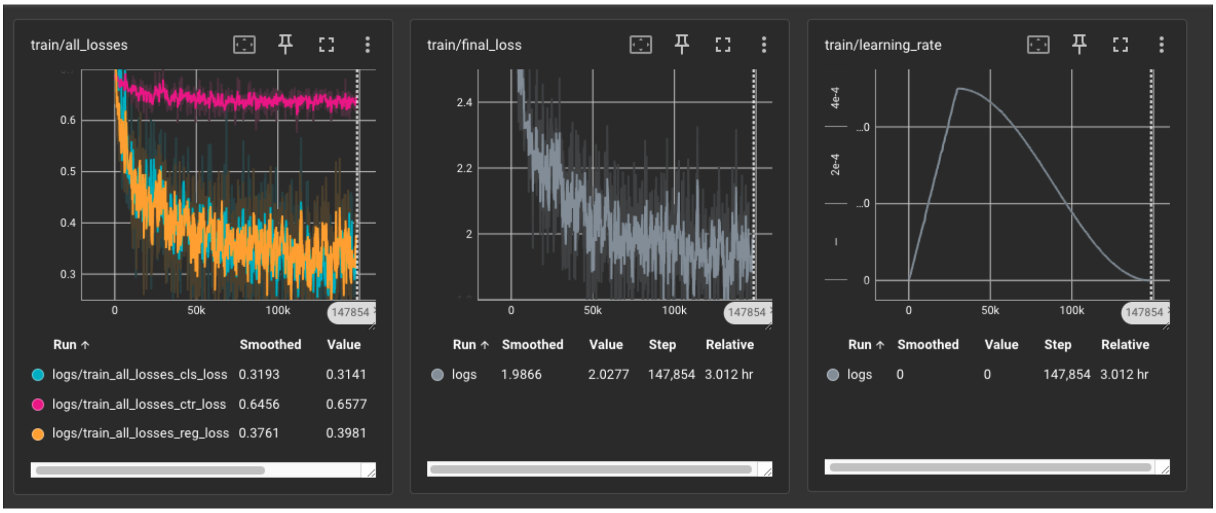


Figure 7: Training loss curves on COCO

#### 4.2.3 Performance

- **Accuracy:** mAP@0.5:0.95 = 0.161, mAP@0.5 = 0.291, mAP@0.75 = 0.159. As shown in Figure 8.

- **Efficiency:**
  – Training time: 3.01 hours for 4 epochs (147,854 steps)
  – Evaluation time: 272.31 seconds on the validation set

- **Analysis:** The model achieves reasonable performance given the limited training (4 epochs). mAP@0.5 of 29.1% demonstrates feasibility on COCO with minimal training. With more epochs and hyperparameter tuning, performance would likely improve.

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=24.88s).
Accumulating evaluation results...
DONE (t=4.79s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.161
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.291
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.159
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.050
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.174
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.260
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.171
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.271
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.285
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.072
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.319
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.462
All done! Total time: 272.31 sec
```

Figure 8: Evaluation result on COCO

# 5 Conclusion and Future Work

**Summary.** We implemented an end-to-end FCOS detector, validated inference correctness against the provided `voc_res18.pth.tar` and reproduced a strong baseline (mAP@0.5 = 60.1%). On VOC, the baseline training (`ResNet-18`, 10 epochs, LR= 0.01) finished in **25 minutes** on a single T4, meeting the efficiency target. Building on this, an extended schedule (**50 epochs**, LR= $8\times10^{-4}$) delivered stable convergence; our final model achieved **mAP@0.5 = 57.4%** on the test set. On COCO (4 epochs, `ResNet-18`), we obtained **mAP@[.5:.95] = 0.161**, **mAP@0.5 = 0.291**, showing reasonable performance under a short schedule.

**Key takeaways.** (1) Reweighting the centerness loss (factor 2×) accelerates its convergence and improves localization quality; (2) Per-class candidate selection in inference avoids head-class domination and benefits rare categories; (3) Unifying $(x, y)$ vs. $(y, x)$ conventions eliminated several failure cases.