EE898: PA1
FCOS: Fully Convolutional One-Stage Object Detection

Geeta Kumari (20194155) **Dated: 12th May 2020**

Paper: FCOS: Fully Convolutional One-Stage Object Detection by Zhi Tian, Chunhua Shen, Hao Chen, Tong He

Introduction

In this assignment, we implement fully convolutional one-stage (FCOS) object detection to solve object detection in a per-pixel manner like semantic segmentation. In contrast to most of the state-of-the-art algorithms, the FCOS implementation is both anchor-box free and proposal free, which saves it from the computations and hyperparameters setting required for anchor box- and proposal-based object detection methods. This makes FCOS very simple yet competitive to the other methods.

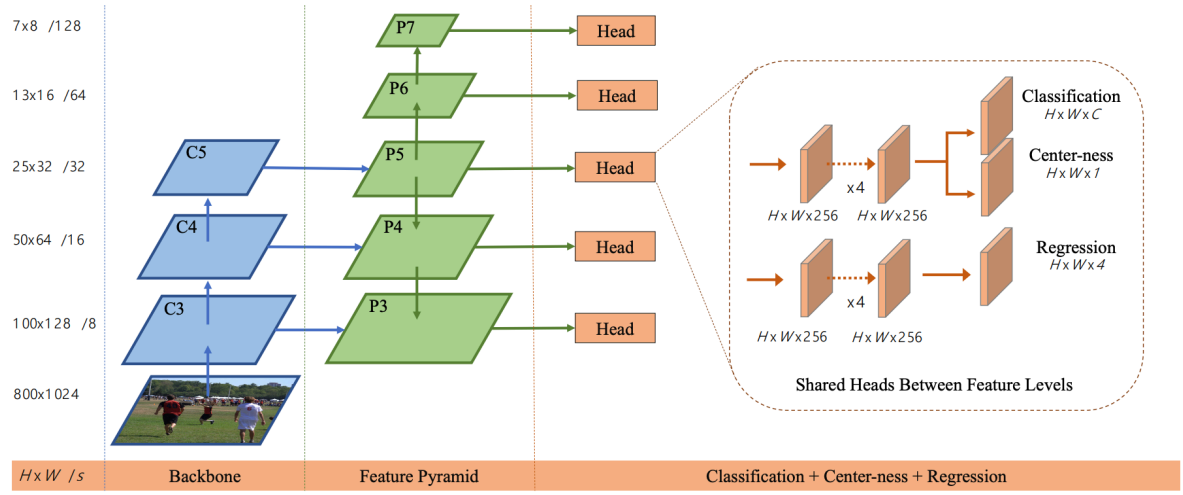


Figure 1: FCOS Network Architecture

FCOS Network Architecture:

Figure 1 shows the architecture design of FCOS. The ResNet-50 [1] is used as backbone – C_3, C_4 and C_5 denote the feature maps of the backbone network. It makes use of FPN (Feature pyramid network) [2], for it detects different sizes of objects on different levels of feature maps. The paper makes use of five levels of feature maps defined as {P_3, P_4, P_5, P_6, P_7}. P_3, P_4 and P_5 are produced by the backbone CNNs' feature maps C_3, C_4 and C_5 followed by a 1x1 convolutional layer with top-down connections. P_6 and P_7 are produced by applying one convolutional layer with the stride being 2 on P_5 and P_6 respectively. Thus, the feature levels P_3, P_4, P_5, P_6 and P_7 have strides 8, 16, 32, 64 and 128 respectively.

Implementation Steps:

1. **FCOS Head**
    - Four convolutional layers are added after the feature maps of FPN network respectively for classification and regression branches.
    - A single-layer branch is added in parallel with classification branch to predict centerness as shown in Figure 1.

- The heads are shared between different feature levels which makes it not only more parameter-efficient but also improves detection performance. However, different feature layers regress different size ranges and it is not reasonable to use identical heads for different feature layers. Thus, instead of using standard exp(x), the original paper uses exp(s_ix) with a trainable scalar s_i to automatically adjust the base of the exponential function for feature level P_i.
- In the improved version, group normalization (GN) is done after each conv layer before activation function in both classification and regression branch. Using GN slightly improves the performance.
- In the implementation part, I add four conv layers with GN under conditional statement and use ReLU as activation function.
- Function *Scale()*, which was already given, is used for adding the trainable scalar s_i as mentioned earlier.
- Centerness branch is added with classification branch in baseline, while in improved version it is added in regression branch.
- The weights of the network are initialized with normal distribution of given mean = 0 and standard deviation = 0.01.

2. FCOS Targets

In this part, we compute ground-truth targets required for training.

- The ground truth bounding boxes for an input image are defined as $\{B_i\}$, where $B_i = (x_0^{(i)}, x_1^{(i)}, y_0^{(i)}, y_1^{(i)}, c^{(i)}) \in \mathbb{R}^4 \times \{1, 2, \ldots, C\}$. $(x_0^{(i)}, y_0^{(i)})$ and $(x_1^{(i)}, y_1^{(i)})$ represents the coordinates of the top-left and right-bottom corners of the bounding box and $c^{(i)}$ is the class that the object in the bounding box belongs to.
- Unlike other object detection works, FCOS directly predict a 4D vector plus a class category at each spatial location on a level of feature maps as shown in Figure 2. That is, it directly regresses the target bounding box at the locations. So, for each location (x, y) on the feature map P_i, first, it is mapped back onto the image as

$$\left(\left\lfloor \frac{s}{2} \right\rfloor + xs, \left\lfloor \frac{s}{2} \right\rfloor + ys\right),$$

  which is near the center of the receptive field of the location (x, y).
- Thus, the function get_points() implements the above formula and accumulates the results for all feature levels.
- Then, if the location (x, y) falls into any ground-truth box, it is considered as a positive sample and the class label of the location is the class of the ground-truth box. Otherwise, it a negative sample and class is just zero or background.
- Besides classification label, the 4D vector regression targets for the location are calculated as follows:

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

where *l*, b*, t*,* and *r** are the distances from the location to the four sides of the bounding box.

- So, the function fcos_targets() implements above equations to calculate regression targets for the locations.
- Since different feature levels are intended to detect different size objects, the range of bounding box regression for each feature level is limited. Thus, after computing the regression targets, if a location satisfies, max(*l*, b*, t*, r**) > m_i or max(*l*, b*, t*, r**) < m_{i-1}, where m_i is the maximum distance the feature level *i* needs to regress, it is set as a negative sample and hence, not required to regress a bounding box anymore.
- In the implementation, (m_2, m_3, m_4, m_5, m_6, m_7) = (0, 64, 128, 256, 512, INF)
- Thus, this way the problem of overlapping between objects is resolved, as different feature levels regress different sizes.
- In the improved version, center sampling is used in order to avoid the problem of low-quality bounding boxes – far from center of the objects. The idea is to use the central portion of ground-truth bounding box as positive samples and hence, computing the regression targets according to only the central portion. This requires defining the radius for the central portion consideration and it has to be different for different feature levels. Therefore, the radius = 1.5 and for each level, it is multiplied with its respective stride.

Figure 2. 4D vector (l, t, b, r) encoding the location of a bounding box at each foreground pixel

- The paper adds an additional branch to the head for centerness computation. The computed centerness is multiplied with scores of the classification branch in order to down-weight the scores of bounding boxes that are far from center of an object, thus, resolving the issue of low-quality boxes as discussed above for center sampling. The centerness is computed as follows,

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

- The centerness basically depicts the normalized distance from the location to the center of the object that the location is responsible for. The square root is

employed to slow down the decay of the centerness. It ranges from 0 to 1 and thus, the centerness branch is trained with binary cross entropy loss.

- The function compute_centerness_targets() computes the centerness targets by employing the above equation by taking the regression targets as inputs.

3. **FCOS Losses**
   - FCOS adopts two types of loss functions i.e. focal loss and IOU loss for classification and regression tasks respectively. It is defined as follows:

$$L(\{\boldsymbol{p}_{x,y}\},\{\boldsymbol{t}_{x,y}\}) = \frac{1}{N_{\text{pos}}}\sum_{x,y} L_{\text{cls}}(\boldsymbol{p}_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{\text{pos}}}\sum_{x,y} \mathbb{1}_{\{c^*_{x,y}>0\}} L_{\text{reg}}(\boldsymbol{t}_{x,y}, \boldsymbol{t}^*_{x,y}),$$

     where L_{cls} is focal loss and L_{reg} is the IOU loss, N_{pos} is the number of positive samples and $\lambda = 1$ is the balance weight for L_{reg} and is set to be 1 in the implementation.
   - In addition, the centerness loss described above is added to this loss for training FCOS. The implementation of losses was already given.

4. **Inference**

   Given an image and forwarding it through network gives classification scores and regression predictions for each location on the feature maps F_i. The function predict_proposals_single_image() implements the inference part per image. Given the class scores, box predictions and centerness scores with locations at all feature levels, the following steps are taken:
   - The location is chosen as positive sample if classification score, $p_{(x, y)} > 0.05$.
   - The scores are then multiplied with centerness to get updated scores, called as score thresholding to filter out low-quality predictions
   - Using the updated scores, top k (k = 1000 here) scores indices are found, which are used to obtain the corresponding classification scores and locations. This is called proposal ranking.
   - The bounding boxes are then obtained by inverting the regression target equations – decoding regressions.
   - Finally, non-maximum suppression is applied to remove duplicated boxes in each image. The post_nms_topk is set to be 100 here. So, there are 100 max detections per image.

Training Details

For my implementation, I trained the network on 2 GPUs (GeForce GTX 1080 Ti), with the learning rate set to 0.005, batch size to 8 and total iterations to 180k with reduction of learning rate by factor of 10 after 120k and 160k iterations respectively.

Quantitative Results

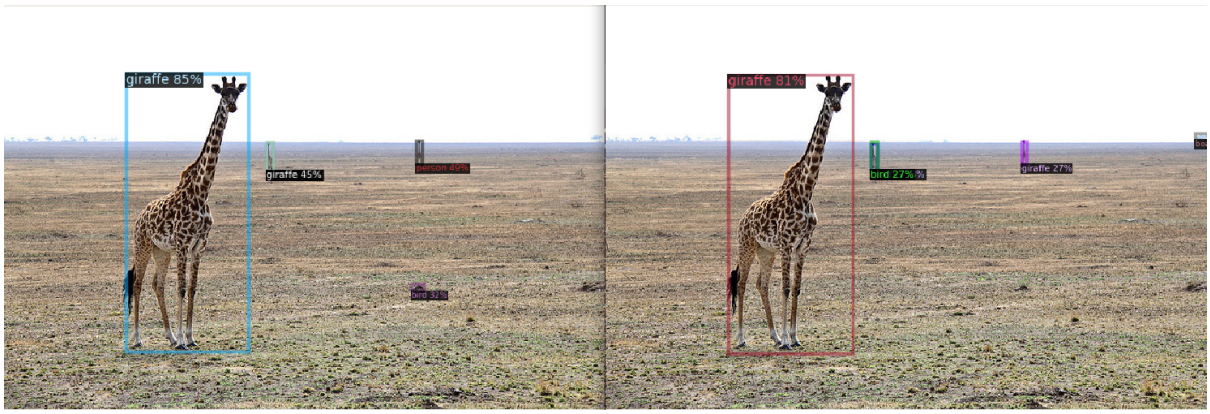
- Baseline-paper denotes the original paper implementation results
- Reproduced Baseline denotes my implementation results
- Improved denotes the results of improvements added to baseline that include group normalization, center on regression layer and center sampling

- Results are generated for two NMS values i.e. 0.5 and 0.6
- The improved model's AP is better than the baseline model which suggests the added improvements have a positive effect on the overall performance
- The reproduced APs is lower than the original paper's, most likely due to the limited GPU environment used for the reproduced models

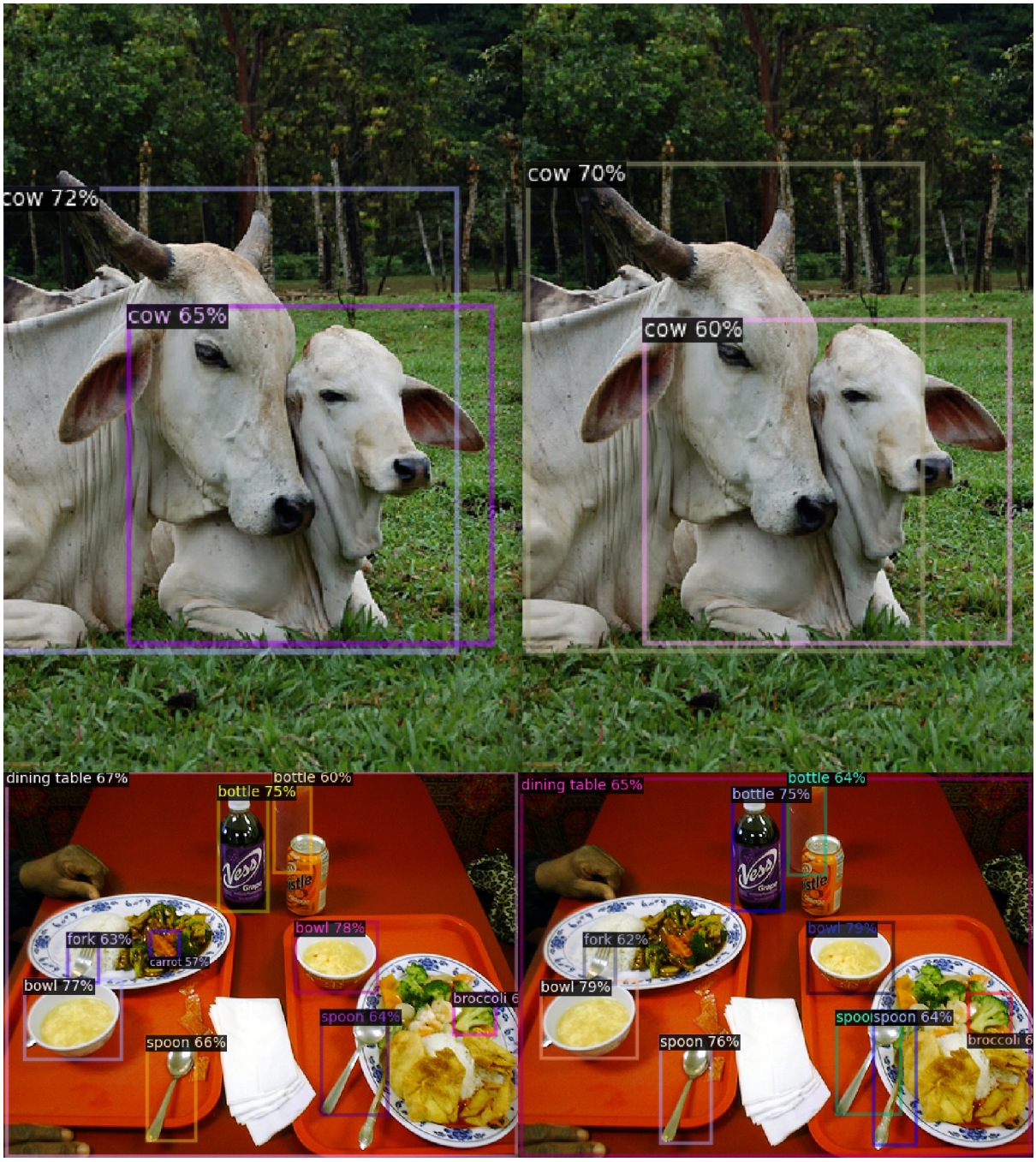
Model	NMS thr.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
Baseline - Paper	0.5	36.4	54.9	38.8	19.7	39.7	48.8	31.4	50.6	53.4
Baseline - Paper	0.6	36.5	54.5	39.2	19.8	40.0	48.9	31.3	51.2	54.5
Reproduced Baseline	0.5	35.6	54.0	37.7	19.8	39.0	47.0	31.3	50.0	52.7
Reproduced Baseline	0.6	35.7	53.7	38.2	20.0	39.3	47.0	31.3	50.5	53.8
Improved	0.5	36.2	55.5	38.5	21.6	39.5	45.8	31.1	50.7	53.9
Improved	0.6	36.3	55.1	39.0	21.8	39.8	45.9	31.1	51.5	55.1

Qualitative Results

- Images on left side are from baseline model
- Images on right side are from improved model – with GN, center sample and regression on center layer
- It is bit hard to make comparisons qualitatively for in both cases there were some wrong predictions about small size objects
- In general, most of the predictions were correct and same in both models if top predictions were considered
- In some cases, improved model did give better predictions considering very small objects as shown in picture with giraffes on farm and one with dining table




person 85%
baseball glove 75%
person 82%
person 81%
baseball glove 65%
person 79%
person 81%
person 36%
person 85%
backpack 43%
person 76%
person 84%
person 83%
person 65%
person 77%
person 27%
tie 68%
tie 60%
tie 77%
person 64%
person 85%
person 83%
person 78%
person 65%
person 79%
tie 68%
tie 72%
boat 64%
boat 39%
boat 47%
boat 64%
boat 42%
boat 64%
boat 41%
boat 56%
boat 58%
boat 60%


cow 72%
cow 65%
cow 70%
cow 60%
dining table 67%
bottle 75%
bottle 60%
fork 63%
carrot 57%
bowl 78%
bowl 77%
spoon 64%
spoon 66%
dining table 65%
bottle 75%
bottle 64%
fork 62%
bowl 79%
bowl 79%
spoon 76%
spooispoon 64%


person 73%
bowl 42%
bowl 38%
bowl 65%
cup 41%
knife 37%
remote 43%
person 75%
sink 36%
cup 39%
bowl 68%
cup 47%
sink 50%
knife 41%7%
person 65%
person 57%
remote 55%
cup 81%
chair 69%
person 64%
person 66%
remote 62%
cup 74%
chair 69%
chair 58%
person 54%
couch 52%
person 62%
dining table 57%
remote 59%
cup 65%
cup 75%
person 74%
couch 59%
person 72%
dining table 56%
cup 66%
cup 63%

Conclusion

In this assignment, the FCOS was implemented, which solves the object detection task in per-pixel fashion. It is anchor-free and proposal-free as opposed to many of other state-of-the-art methods. It is computationally very efficient, avoiding all the hyper-parameters related to anchor boxes and hence, making it a simpler framework to work with.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pages 770–778, 2016.
[2] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, ´ Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pages 2117–2125, 2017.