Reimplementation and Optimization of FCOS: Enhancing Object Detection with Lightweight Backbones

Gu Chengkang, BAI Huiyu, Liu Zhimeng, Bai Huiyu, Zhou Ruoheng

College of Computing and Data Science, Nanyang Technological University, Singapore
{CHENGKAN001, HUIYUBAI001, ZHIMENG001, YAOJ0014, ZH0121NG}@ntu.edu.sg

Abstract

In this report, we first re-implement the FCOS algorithm for object detection. To reduce the backbone's parameter count while maintaining accuracy, we replace the original ResNet50 backbone with lightweight alternatives, and we achieve an acceptable loss in accuracy through a series of fine-tuning steps. In our experiments, we test ***VoVNet-27-slim***, ***ResNet18***, ***MobileNetV2***, and evaluate on VOC 2007 test dataset. We found that ResNet18 based FCOS model achieves the highest mAP, our model allows deployment on devices with limited storage, such as mobile phones, drones, and embedded IoT devices, without compromising detection accuracy. Our source code is available at:https://github.com/ryougishiki1999/NTU_AI6103_Project_FCOS_2024fall

Introduction

Object detection is a basic computer vision task, the objective is to locate and identify the objects in images. Over the years, researchers have developed different approaches to tackle this problem, which can be broadly classified into anchor-based and anchor-free methods.

Anchor-Based Approaches

Traditional object detectors such as Faster R-CNN (Ren et al. 2016) and early versions of YOLOv3 (Redmon and Farhadi 2018) rely on predefined anchor boxes. Tuning of anchor sizes and ratios of the boxes are required to match the diverse scales of object in datasets. Anchor-based methods involve complicated pipelines and are computationally expensive due to the large number of anchors that need to be processed.

Anchor-Free Approaches

Anchor-free methods have been proposed to address the limitation of Anchor-free methods. This approach eliminates the need for predefined anchor boxed by directly predicting object locations and sizes from feature maps. Notable examples of anchor-free detectors include FCOS (Fully Convolutional One-Stage Object Detection) (Tian et al. 2019) and CenterNet (Duan et al. 2019). CenterNet utilizes keypoint estimation to locate the center of object and regress to bounding box sizes. FCOS simplifies the detection pipeline, through directly predicting the bounding box, this approach eliminates the need of computing the overlapping between the predicted anchor boxes and ground truth.

Dataset

The original FCOS paper used the COCO dataset (Lin et al. 2015), a large-scale object detection benchmark with over 80 categories and 330,000 images, to demonstrate the model's performance. However, due to our limited computational resources, we chose to train and evaluate our models on the PASCAL VOC dataset, specifically the VOC 2007 (Everingham et al. a) and VOC 2012 datasets (Everingham et al. b).

The PASCAL VOC dataset provides a more manageable alternative, containing 20 object categories with approximately 25,000 annotated images. For our experiments:

- The VOC 2012 train + val datasets were used for training and fine-tuning.
- The VOC 2007 test dataset was used for evaluation.

Model Architecture

Backbone

Deep networks may encounter the issues of gradient vanishing or gradient exploding during training, which results in the network performance not improving with the increase of depth. ResNet proposed the concept of residual learning, which directly jumps the input to the output by introducing a shortcut connection, and constructs the following structure:

$$y = F(x, \{W_i\}) + x, \tag{1}$$

This short-circuit connection enables the network to learn the residual directly, avoiding directly fitting the target value. Thus, training is accelerated, and gradient vanishing is alleviated.

ResNet-50 In our initial reimplementation, we utilize ResNet-50 (He et al. 2015) as backbone, ResNet-50 is a deep convolutional neural network consisting of 50 layers, with sequential bottleneck residual blocks that facilitate, skip connections help mitigate the gradient vanishing problem.

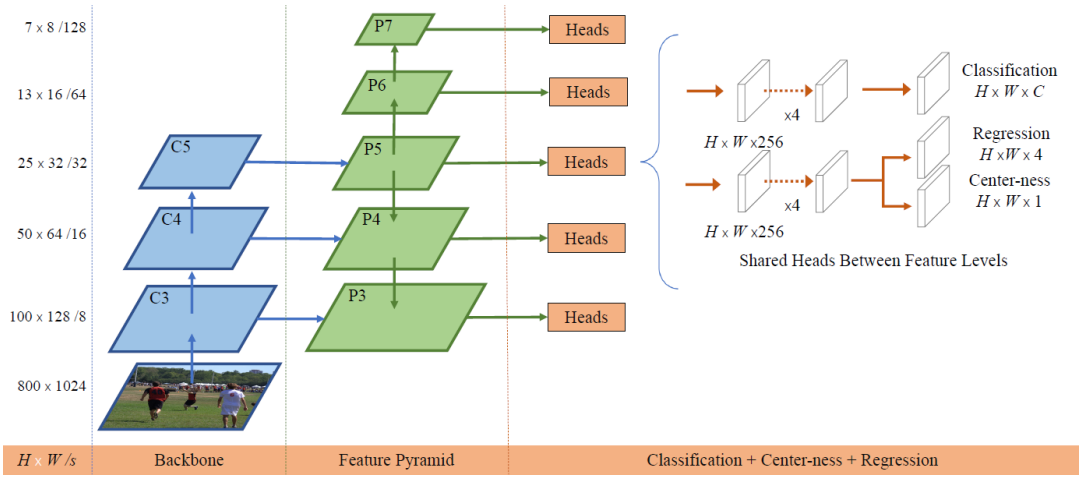


Figure 1: Pipeline of FCOS) (Tian et al. 2019)

From the ResNet-50 backbone, we extract three crucial feature maps, denoted as C3, C4, and C5 (see Figure 1). The feature maps are extracted with the following spatial dimensions, assuming an input image size of 800×1024 pixels:

- **C3**: 100×128
- **C4**: 50×64
- **C5**: 25×32

We also implement several different backbones with fewer parameter sizes for lightweight optimization.

ResNet-18 ResNet-18 simplifies the structure of deep neural networks by employing only an input layer, four residual module groups, and an output layer. This design achieves high performance with a shallower network and relatively fewer parameters. Due to its reduced depth, the optimization process becomes more straightforward. In total, ResNet-18 has a size of approximately 11.7 million parameters, which is more than half less than ResNet-50's 25.6 million parameters. Therefore, ResNet-18 is better suited for devices or tasks with limited resources, such as mobile or embedded development.

MobileNet-V2 MobileNet-V2 (Sandler et al. 2019) is a lightweight neural network architecture specialized for computing-limited devices. The mobileNet only contains around 3.4M parameters. Based on the success of its predecessor, MobileNet-V1, MobileNet-V2 achieves significant improvements in both accuracy and efficiency, making it a popular alternative for various mobile deep learning applications such as image classification, object detection, and semantic segmentation. Mobilenet-V2 involves several improvements, like inverted residuals, linear bottlenecks and depthwise separable convolutions.

- **Inverted residuals** The inverted residual block in MobileNet differs from ResNet by first expanding channels with a 1x1 convolution (expansion layer), typically by a factor of 6, followed by a depthwise convolution for feature extraction. Afterward, a 1x1 convolution (projection layer) reduces dimensions, with no ReLU applied to the output to avoid information loss. Shortcuts are used only when input and output channels are equal, and stride = 1. This structure preserves the manifold's information flow, minimizes feature loss, and ensures efficient feature extraction in a lightweight design.
- **Linear bottlenecks** Linear Bottlenecks leverage the idea that neural network features (manifold of interest) lie in a low-dimensional subspace within the channel dimension. The model preserves essential information while minimizing distortion by reducing channels with narrow bottleneck layers and removing non-linear activations (ReLU) in the output. This dimensionality reduction reduces computational cost and memory usage, enabling lightweight design. The width multiplier further controls channel dimensions, balancing accuracy and efficiency. Linear Bottlenecks allow MobileNet-V2 to efficiently encode features, achieving a trade-off between performance and resource constraints.

VoVNet-27-slim The VoVNet (Lee et al. 2019) architecture is based on the One-Shot Aggregation (OSA) module, which aggregates feature maps efficiently, promoting better feature reuse and improving accuracy while keeping the model computationally lightweight. The VoVNet-27-slim variant is a compact version of VoVNet optimized for smaller models, making it suitable for mobile and edge computing applications.

- One-Shot Aggregation (OSA) Module
- Improved Feature Representation

Feature Pyramid Network

Introduction It is well known that Feature Pyramids are a basic component in object detections tasks at different scales. Moreover, there are diverse variants through development of Feature Pyramids.

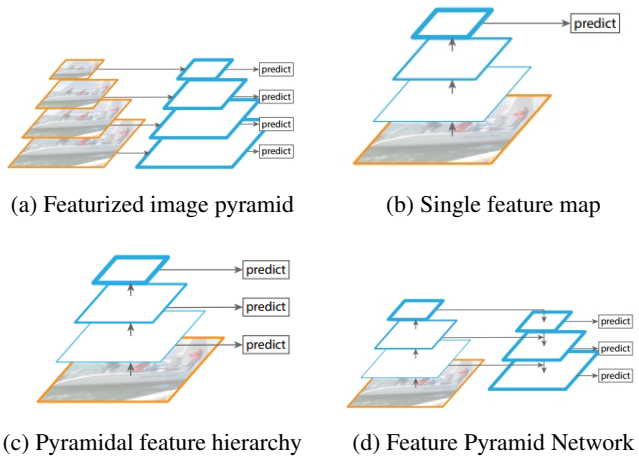
One of Predominant Feature Pyramids is Featurized Image Pyramid(Dalal and Triggs 2005) shown in figure 2a heavily used in the era of hand-engineered features. Through constructing complete multi-scale feature pyramids, this

architecture achieves robust representation and detection across object scales. However, its major limitation is the extensive computational cost of processing multiple full-scale feature maps, which constrains its practical deployment under computational and real-time requirements.

Later, as shown in figure 2b, Single Shot Detector (SSD) came into existence as the first attempts using a pyramidal feature hierarchy(Liu et al. 2016). Single Feature Map detection employed by SSD, which utilizes a singular feature map for prediction, significantly simplifies computation and enhances efficiency. While its streamlined architecture facilitates implementation and deployment, this simplification introduces critical limitations: inadequate multi-scale object handling capability, relatively weak feature representation, and suboptimal detection performance in complex scenarios.

Further progression in this research direction, Pyramidal Feature Hierarchy shown in figure 2c achieves a balance between computational efficiency and multi-scale feature representation by utilizing the inherent hierarchical structure of convolutional neural networks. However, its primary limitation lies in the substantial semantic gap between hierarchical features, with insufficient semantic information in shallow features, which constrains its performance in complex visual tasks.

Looking across the developmental history of Feature Pyramids, the weak semantics at different scales in feature representations have become the most significant challenge impeding the development. As figure 2d indicates, Feature Pyramid Network (FPN)(Lin et al. 2017) innovatively addressed these issues through its architectural design of "top-down pathway + lateral connections". FPN achieves effective fusion of features across hierarchical levels while maintaining computational efficiency, enabling rich semantic information at each layer. Although its network architecture and training process are relatively complex, this structure has demonstrated significant advantages in multi-scale visual tasks such as object detection, becoming an integral component of modern vision systems.



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 2: (a)-(d): Variations in Feature Pyramid

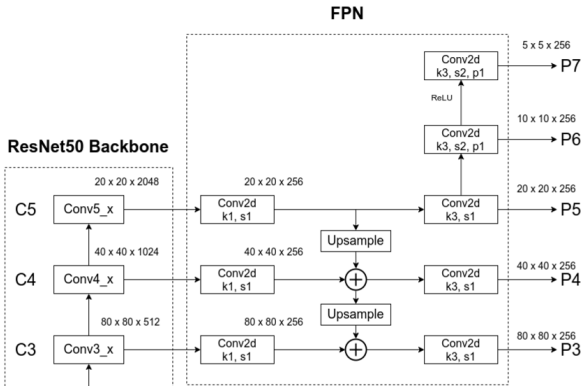


Figure 3: FPN workflow

Workflow of FPN in FCOS FPN in FCOS ensures the robust implementations of multi-level predictions as shown in Figure 3. As introduced in previous section, FPN contributes to detecting different size objects on different map levels. Besides, with the help of upsampling, the feature maps from different layers are integrated into a unified representation. The fused feature map is then utilized for prediction, aiming to enhance the network's detection performance.

As figure 3 states, CNN's feature maps C_3, C_4 and C_5 produced by the backbone will be imported into workflow of FPN to generate multi-level of fusion pyramid feature maps defined as $\{P_3, P_4, P_5, P_6, P_7\}$. Specifically, P_3, P_4, P_5 are produced from C_3, C_4 and C_5 with top-down connections. The remaining pyramid features P_6, P_7 are generated based on P_5 and P_6 through a convolutional layer with a same kernel size of 3×3 and a stride of 2. Initially, P_5 is applied to generate P_6. Then outcome P_6 is used to generate P_7 undergoing the same procedure. Finally, the set of pyramid feature maps $\{P_3, P_4, P_5, P_6, P_7\}$ will be passed along the workflow to the next stage, serving as input for MultiHead module to compute classification, regression and centerness.

The above describes the complete workflow of the Feature Pyramid Network (FPN). Returning to the implementation details, there are top-down pathways and lateral connections. In top-down pathway,s tarting from the deepest layer, feature maps are progressively upsampled. Nearest-neighbor upsampling is typically used to increase the spatial resolution of the feature maps by a factor of 2 thereby ensuring the height and width of feature maps are consistent in the fusion.

On the aspect of lateral connections, the convolutional layer before upsampling shown in left area of figure 3 actually consists of N 1×1 kernels. These 1×1 convolutional kernels target to adjust the number of output channel, N, which ensures the channels of feature maps are consistent in the fusion.

Throughout top-down and lateral connections, the two feature maps are fused through element-wise addition. Take the generation process of P_4 as the illustration. Firstly focus on C_4 of which shape is $80 \times 80 \times 512$ representing height, width and input channels passing through the lateral connection. $N = 256$ 1×1 convolutional kernels are applied to C_4 to shift 512 input channels to 256 output channels.

The outcome forms one of two feature maps in the element-wise addition. The other feature map comes from top-down connections. Returning to the initial step, C_5 of which shape is $20 \times 20 \times 256$ not only needs the lateral connections to make the output channels consistent but also requires the upsampling by factor of 2 to align height and width with the feature map from the previous layer, C_4. The outcome of C_5 through upsampling and unifying output channels meet up with the outcome of C_4 throught unifying output channels and execute the element-wise addition. The outcome of element-wise addition keeps further on lateral connections and is applied with the 3×3 convolutional layer mentioned in the previous paragraphs to generate the final pyramid feature map P_4.

Assigning objects to FPN Objects here refer to ground truth instances implying the annotations which includes the positional information, category, etc. The issues of assigning objects to different levels of FPN are as follows.

- **Semantics of features varies across different scales.** A well-designed allocation strategy ensures that each object is assigned to the most suitable feature level.
- **Class imbalance in training samples.** Improper allocation may lead to an imbalance in the number of training samples across feature layers. For instance, if all objects are assigned to a single layer, other layers may lack positive samples to learn from.

Compared to the assigning strategy for object proposals presented in (Lin et al. 2017) according to the formulation $k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$ where $k \in \{3, 4, 5, 6, 7\}$, we finally employ the $\max(l^*, t^*, r^*, b^*)$ strategy where l^*, t^*, r^*, b^* represent the distances from a point (the feature map's projection onto the original image) to the left, top, right, and bottom boundaries of the Ground Truth (GT) box, respectively. The process of assigning strategy is illustrated as following steps.

- Firstly, compute l^*, t^*, r^*, b^* for each location on all feature levels i, $i \in \{2, 3, 4, 5, 6, 7\}$
- Afterwards, for each location at feature level i, if it satisfies $\max(l^*, t^*, r^*, b^*) \leq m_{i-1}$ or $\max(l^*, t^*, r^*, b^*) \geq m_i$, it will be set as negative sample and not required to compute regression results.

m_i represents the maximum distance for location from a point in feature map in need of computing regression in feature level i. In our work, we set $m_2, m_3, m_4, m_5, m_6, m_7$ as 0,64,128,256,512,∞ to address the issues mentioned above.

Effects of FPN on ambiguous samples There is a concern that plethora of ambiguous samples may arise due to overlap among ground-truth boxes. The presence of ambiguous samples will inevitably interfere with the network's learning and prediction processes. The default approach is to assign the point to the ground-truth (GT) box with the smallest area.

However, FPN effectively helps us address this issue. The FPN structure, along with the FPN object assigning strategy, significantly reduces the number of ambiguous samples

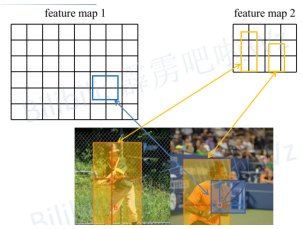


Figure 4: example on FPN for ambiguous samples

as experiments conducted in (Detector 2022). In FPN, multiple prediction feature maps are utilized, with each feature map at a different scale responsible for predicting objects of various sizes. For example, p_3 is responsible for predicting small objects, p_5 handles medium-sized objects, and p_7 is tasked with predicting large objects. Take figure 4 as illustration. For small objects, such as paddles, they are assigned to feature map 1 according to the objects assigning strategy. On the other hand, large objects, such as people, are assigned to feature map 2. This approach helps separate overlapping objects when matching positive and negative samples, effectively addressing most of the ambiguous samples problem.

Multi-Head

The Head module in FCOS functions as the output component responsible for receiving multi-scale feature maps from the Feature Pyramid Network (FPN) and producing the results for classification, regression, and centerness estimation.

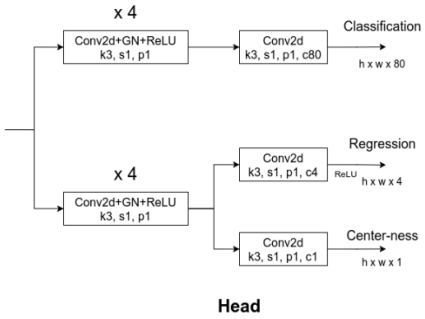


Figure 5: The Structure of the Head in detail

This model employs five identical Heads to process the feature maps at each corresponding scale. All features utilize this shared Head structure for the final prediction operations. Each Head is divided into a classification and a regression part, each composed of four convolutional modules connected in series. Each convolutional module uses a 3×3 convolutional kernel with a stride of one and padding of one. Group normalization is applied during computation, and ReLU is used as the activation function.

In the classification output layer, the original feature map of size $H \times W \times 256$ is reduced to $H \times W \times 80$, where each channel corresponds to the probability data of a specific category. The maximum value among these channels

is selected as the target probability. In the regression part, the final output layer has two outputs: one provides the regression results, and the other outputs the centerness predictions (which will be introduced in the subsequent loss function discussion). Specifically, the bounding box regression reduces the feature map dimensions from 256 to 4, with each value representing the distance from the center point to the top, bottom, left, and right edges of the bounding box. The centerness estimation directly reduces the dimension to 1, where each value indicates how close that point is to the center. After obtaining the results from all five Heads, it is necessary to aggregate all the outputs. All the predicted boxes from the five output Heads are combined into a single list, forming the complete set of all prediction results. Based on this comprehensive aggregation, various composite $Scoreconfidence\text{Score}_{\text{confidence}}$ are calculated. Predicted boxes with confidence scores lower than a specified threshold are filtered out, retaining a subset of high-confidence bounding boxes. Finally, Non-Maximum Suppression (NMS) is applied with a confidence threshold T. The Intersection over Union (IoU) between the retained boxes and other boxes is computed. If the IoU exceeds the predetermined threshold, that box is suppressed (removed). This process is repeated until the best bounding box is obtained as the final result.

Loss Function

The model's output comprises two main heads. The classification head generates both the classification results and the centerness scores, while the regression head produces the regression predictions. Accordingly, the loss function must consider the outputs from both of these components and combine their losses for gradient descent optimization. Therefore, the overall loss function is as follows:

$$L(\{p_{x,y}\},\{t_{x,y}\}) = \frac{1}{N_{\text{pos}}}\sum_{x,y} L_{\text{cls}}(p_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{\text{pos}}}\sum_{x,y} 1_{\{c^*_{x,y}>0\}} L_{\text{reg}}(t_{x,y}, t^*_{x,y}) \tag{2}$$

Classification Loss: The classification loss primarily consists of the results from both category prediction and centerness prediction. In the context of category prediction, the vast majority of regions in the input image are background (areas without targets), while the foreground (target) regions constitute a relatively small proportion. This imbalance can lead to a series of issues, such as weakened learning effectiveness for positive samples due to class imbalance. To address this problem, FCOS employs the Focal Loss to compute the classification loss. The Focal Loss enhances the standard cross-entropy loss by incorporating a dynamically adjustable weighting factor, which increases the focus on hard-to-classify samples (e.g., foreground samples with low prediction confidence) and decreases the emphasis on easy-to-classify samples (such as background samples).

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{3}$$

(P_t is the estimated probability of the model for classes at t steps. α_t is a weighting factor that balances the positive and negative examples, we set it to 0.25. γ represents the focusing parameter for decreasing the relative loss for the examples that already classified, we set it to 2)

Regression Loss: When performing bounding box regression, the primary objective is to ensure that the model's predicted bounding boxes are as close as possible to the true target boxes. In this approach, we do not train the model by inputting the ROI (Region of Interest) coordinate parameters of the bounding boxes. FCOS introduces a loss function based on the Intersection over Union , which divides the image into small regions and treats the areas where the target is located as positive samples. This method facilitates better alignment between the predicted boxes and the ground truth boxes, providing more stable and effective loss values. The calculation method is as follows:

$$\text{Loss}_{\text{GIoU}} = 1 - \text{GIoU} \tag{4}$$

$$\text{GIoU} = \text{IoU} - \frac{|C \setminus (A \cup B)|}{|C|}, \quad \text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{5}$$

In this formula, A is the area of the predicted bounding box. While B respects the Area of the ground truth box. The symbol $A \cap B$ shows the overlap Area between the predicted box and the real box. $A \cup B$ is the Area of the union of the A box and the B box. Therefore, the Intersection over Union (IoU) represents the degree of overlap area; higher values indicate that the expected box is closer to the ground truth. The Generalized IoU (GIoU) introduces the minimal enclosing area C and its remaining region $|C \setminus (A \cup B)|$. This allows for effective gradient descent during training, even in cases where no overlap occurs. Finally, subtracting the GIoU from 1 yields the final regression loss value.

Centerness: In addition to computing the regression loss, we also need to calculate the loss for predicting centerness. Predicting centerness is crucial for anchor-free detection, and the authors have proposed a novel method for computing the centerness target value.

$$c^* = \sqrt{\left(\frac{\min(l^*, r^*)}{\max(l^*, r^*)}\right) \times \left(\frac{\min(t^*, b^*)}{\max(t^*, b^*)}\right)} \tag{6}$$

In this context, l,t,r, and b represent the distances from a specific point in the image to the left, top, right, and bottom edges of the target bounding box. Variables $\frac{\min(l^*,r^*)}{\max(l^*,r^*)}$ and $\frac{\min(t^*,b^*)}{\max(t^*,b^*)}$ measure the relative centerness of the point between the left-right and top-bottom edges; values closer to 1 indicate more excellent proximity to the center. After multiplying and rooting, The final centeredness score ranges between 0 and 1, where values closer to 1 signify points nearer to the centre, and values closer to 0 indicate points nearer to the edges. The centerness score lies within this interval, so it can be regarded as a binary classification problem. Therefore, we employ the binary cross-entropy loss to optimize the predicted centerness scores, as given by the following formula:

$$\text{Loss}_{\text{centerness}} = -\left[c^* \log(c) + (1-c^*)\log(1-c)\right] \tag{7}$$

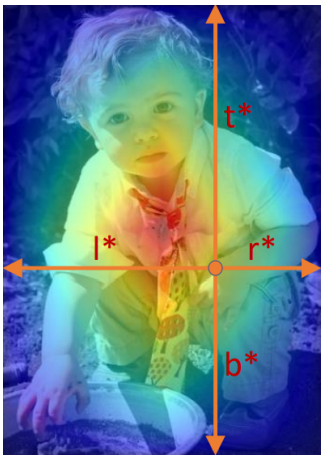


Figure 6: Heat map represented the value of centerness(Tian et al. 2019): The red value is very close to 1 in the center, while the blue value is close to 0 at the edge of the image.

Here, c denotes the ground truth centerness score, and c^* represents the predicted value; during the logarithmic computation, samples with a more significant discrepancy between c and c^* are penalized more heavily.

By summing the loss obtained from the classification prediction and the loss from the centerness prediction, we will obtain the final Classification Loss Function.

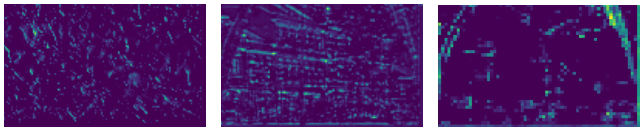
Experiment

Data Pre-processing

First, the images undergo preprocessing. Each image is resized so that its shorter side is standardized to 800 pixels, and the longer side is adjusted proportionally based on this short side. This approach maintains the original aspect ratio of the images, preventing distortion of the geometric shapes of the target objects. Additionally, zero-padding is applied to the input images to ensure that both the width and height are multiples of 32, accommodating the downsampling architecture of the Feature Pyramid Network (FPN). The next step is to perform data augmentation on the image to enhance the variety of samples and thus improve the generalization of the model. We defined three different methods, and each image has a 30% probability of performing a certain enhancement. The first enhancement method is to flip the image horizontally. The second one will randomly perturb the color of the image, such as randomly adjusting the brightness, contrast, saturation, etc. of the image. The third method performs random cropping, removing certain regions of the image while correspondingly adjusting the boundaries of the target bounding boxes. Finally, the pixel values of the images are normalized from a range of 0–255 to a range of 0–1 before being input into the model for training.

Train for original backbone ResNet50

FCOS uses ResNet50 as the backbone network to extract multi-scale basic features of images. Three feature maps of different scales, C3, C4, and C5, are output during the extraction process.



(a) C3 (b) C4 (c) C5

Figure 7: 3 scale feature maps

C3 is the output of the third residual stage of ResNet, with a feature map size of $\frac{H}{8} \times \frac{W}{8}$ and a depth of 512. C3 has a larger scale and contains more spatial details to capture small target information. C4 is the output of the fourth residual stage of ResNet, with a feature map size of $\frac{H}{16} \times \frac{W}{16}$ and a depth of 1024. C5 is a deeper feature map with a scale more $\frac{H}{32} \times \frac{W}{32}$ and a depth of 2048, which contains vital semantic information suitable for detecting large targets.

After obtaining the feature maps of C3 to C5, they will be passed to the FPN layer for further convolution operations to extract the P Feature map.

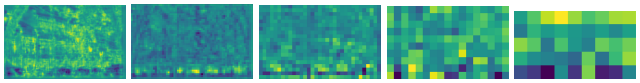


Figure 8: Feature maps output by FPN. From left to right as P3, P4, P5, P6 and P7.

Among them, P3, P4, and P5 are obtained from C3, C4, C5 in the backbone network, and their depth is unified to 256. Then, starting from the P5 layer, upsampling and downsampling are performed. P4 and P3 will be fused with the upsampling of the P5 layer to form the final output and passed to their respective heads. P6 and P7 are generated by downsampling of P5, not directly from the backbone network. Therefore, the spatial resolution of P7 and P6 is significantly reduced, and they can no longer represent the detailed features of the image, only containing the most abstract features in the picture. They are usually used to detect a small number of large targets.

Finally, these results will be passed into the five heads of the same architecture to predict the final results and obtain the final prediction results through non-maximum suppression (NMS) with a set confidence threshold T.

Finally, these results are fed into five Heads with identical architectures. Each Head outputs three results: the probability distribution of target categories (classification results), the positions of the bounding boxes (regression results), and the centerness scores. These three outputs are incorporated into the loss functions mentioned earlier to perform backpropagation, continuously updating the model parameters. In our training process, we employed an initial learning rate of 0.1 to facilitate rapid updates and utilized a "constant" learning rate warm-up strategy to ensure stability at a relatively low learning rate during the initial stages. After the warm-up phase, a weight decay of $\lambda = 0.0001$ was applied during training for regularization to prevent overfitting. We

set the batch size to 16 and conducted 90,000 iterations, totalling $16 \times 90,000 = 1,440,000$ samples, to train the model thoroughly. This amount is sufficient to cover the entire COCO training dataset. The final training loss results are shown in Figure X.

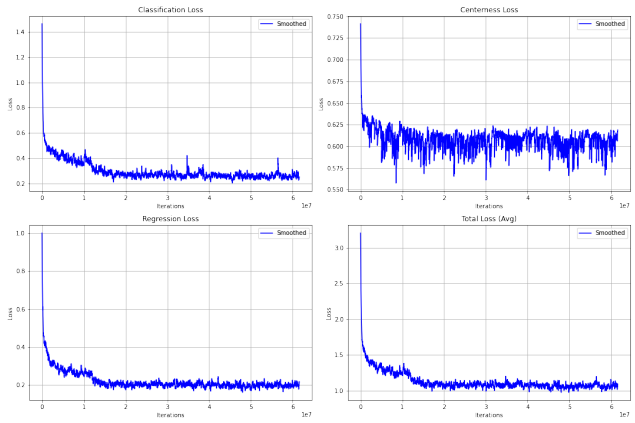


Figure 9: Classification loss, Regression loss, Centerness loss and Total loss decrease with the training processing.

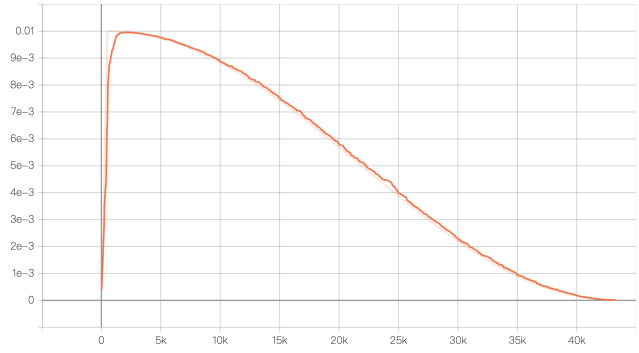
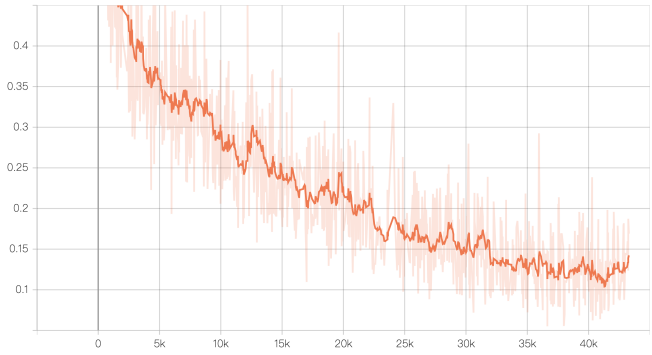


Figure 10: LR schedule using cosine annealing. The x-axis represents the total training steps, and the y-axis represents the learning rate, which decreases from 1×10^{-2} to 1×10^{-5} with a warm-up phase over 0.01 of the total steps.

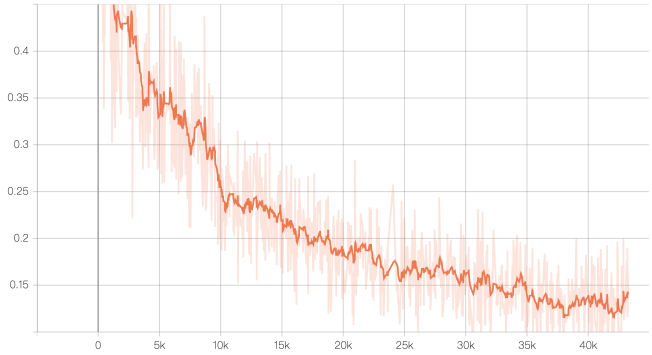
Backbone Replacement and Fine-Tuning

This section will explain experiment setup starting from the initial configuration, and final configuration after hyperparameter tuning and optimizations.

Initial Setup Initially, we replaced the ResNet-50 backbone in FCOS with three lightweight alternatives: VoVNet,ResNet-18, and MobileNetV2. These backbones are chosen to reduce parameter count and computational cost. The models are first trained on the VOC 2012 train dataset. Fine-tuning is conducted on the VOC 2012 val dataset. BN layers in the backbones are frozen throughout the entire experiments to preserve pre-trained feature statistics. Initial Hyperparameters are:



(a) Classification Loss for ResNet-18.



(b) Regression Loss for ResNet-18.

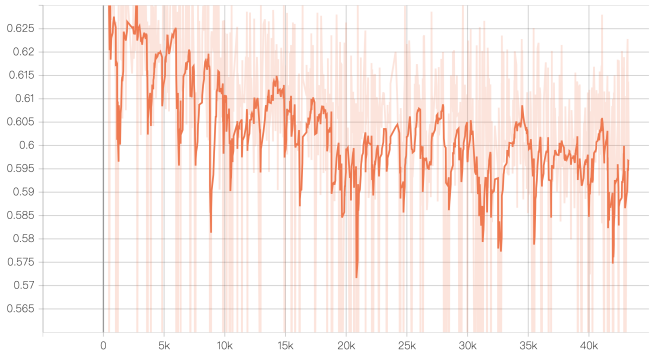
Figure 11: Loss curves for ResNet-18 during training: (a) classification loss and (b) regression loss.

Backbone	mAP	Parameters (M)
MobileNetV2	0.707	3.7
ResNet-18	0.725	11.7
ResNet-50	0.780	23.5

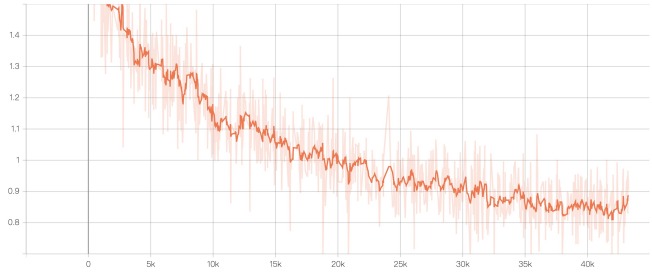
Table 1: Comparison of mAP and backbone parameters for different backbones evaluated on the VOC 2007 test dataset.

- **Learning Rate:** The LR was initially set between 1×10^{-3} and 1×10^{-5}, following a cosine annealing schedule.
- **Batch Size:** A batch size of 64 was used, which was appropriate for GPUs with sufficient memory.
- **Epochs:** Training was conducted over 180 epochs, providing ample time for convergence.
- **Warm-up Phase:** The learning rate was warmed up over 0.01 of the total training steps, starting from a very low value and gradually increasing to 0.01 of the maximum learning rate.

Observations from Initial Setup: We attempt to fine-tune the hyper-parameter of VoVNet-27-slim based FCOS, but VoVNet-27-slim struggled to converge and exhibited poor performance on both VOC 2012 val and VOC 2007 test datasets. The classification loss plateaued between 0.4 and 1.5, and the total loss remained within the range of 1.4 to



(a) Centerness Loss for ResNet-18.



(b) Total Loss for ResNet-18.

Figure 12: Loss curves for ResNet-18 during training: (a) centerness loss and (b) total loss.

1.5. As a result, we decided to discontinue fine-tuning for this backbone. After fune-tuning the other 2 backbone, we find the following hyper-parameters achieve the best performance:

- **Final Hyperparameters:**
  - **Learning Rate:** 1×10^{-2} to 1×10^{-5}, following a cosine annealing schedule. Refer to Figure 10 for the learning rate curve.
  - **Batch Size:** 8.
  - **Epochs:** 30, with 25 epochs being sufficient to fit the model.

  The mAP results are shown in Table 1.The loss curves for ResNet-18 during training are shown in Figures 11 and 12. Classification and regression losses are presented in Figure 11, while centerness and total losses are shown in Figure 12.

The original ResNet-50 backbone, trained on the combined VOC 2012 and VOC 2007 datasets, achieved a mAP of 0.78. Our MobileNetV2 backbone, designed for efficiency, achieved a mAP of 0.707, demonstrating a good trade-off between performance and computational cost. The ResNet-18 backbone achieved a mAP of 0.725 under the same conditions, providing a balance between accuracy and model size. Additionally, we ran inference on two sample images using the ResNet-18 and MobileNetV2 backbones to compare their detection performance qualitatively. The results, shown in Figures 13 and 14, highlight the effectiveness of both models in real-world scenarios.

Figure 13: Inference results on a sample image using the ResNet-18 backbone. Detected bounding boxes and classification scores are shown.

Figure 14: Inference results on a sample image using the MobileNetV2 backbone. Detected bounding boxes and classification scores are shown.

Conlusion and Future Work

In this report, we re-implimented the FCOS algorithm, and in order to deploy on resource-constrained device, we integrate the lightweight backbone with the FCOS algorithm, by replacing ResNet-50 backbone with MobileNetV2 and ResNet-18. We greatly reduce the backbone parameter while maintaining detection accuracy, through a series of fine-tuning experiments, we obtain optimal hyperparameters, such as learning rate schedules and training epochs, to achieve the best balance between accuracy and model efficiency. For future work, we can explore other backbone like MobileNetV3, and apply advanced methods such as quantization and pruning to further improve the efficiency of the model.

References

Dalal, N.; and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer soci-*

ety conference on computer vision and pattern recognition (CVPR'05), volume 1, 886–893. Ieee.

Detector, A.-F. O. 2022. Fcos: a simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4).

Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; and Tian, Q. 2019. CenterNet: Keypoint Triplets for Object Detection. arXiv:1904.08189.

Everingham, M.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. ????a. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Everingham, M.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. ????b. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Lee, Y.; won Hwang, J.; Lee, S.; Bae, Y.; and Park, J. 2019. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. arXiv:1904.09730.

Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37. Springer.

Redmon, J.; and Farhadi, A. 2018. YOLOv3: An Incremental Improvement. arXiv:1804.02767.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:1801.04381.

Tian, Z.; Shen, C.; Chen, H.; and He, T. 2019. FCOS: Fully Convolutional One-Stage Object Detection. arXiv:1904.01355.