CS329 Machine Learning(H) Project Proposal Report: Modification of Focal Loss in Object Detection

Zubin Zheng 12112328, Dahui Li 12112411

December 10, 2023

Abstract

In object detection, the main obstacle that makes one-stage detector achieve to the similar accuracy as two-stage detector is that the extreme foreground-background class imbalance. One solution is to reshape loss function. This report presents our exploration on improving the performance of object detection by modifying the loss function Focal Loss, which is our novelty of this work. The evaluation platform is HPC-AI service platform and we have train the model with Focal Loss as baseline. In the future, we will train another model with Modified Focal Loss* to compare with baseline, expecting improvement in performance(AP).

1 Introduction

In autonomous driving, object detection and object tracking are crucial tasks that play a significant role in the decision-making process during the operation of intelligent vehicles.

The current object detection and object tracking tasks can be categorized based on methods into Camera-based, LiDAR-based, and Camera-LiDAR Fusion-based approaches. They can also be classified based on tasks into detection and tracking. We are interested in Camera-based 2D Object Detection.

1.1 Background

Object detection is an important computer vision task that deals with detecting instances of visual objects of a certain class (such as humans, animals, or cars) in digital images. The goal of object detectiojn is to develop computational models and techniques that provide the knowledge: What objects are where? There are some metrics in object detection, which contains accuracy (including classification accuracy and localization accuracy) and speed[ZCS+23].

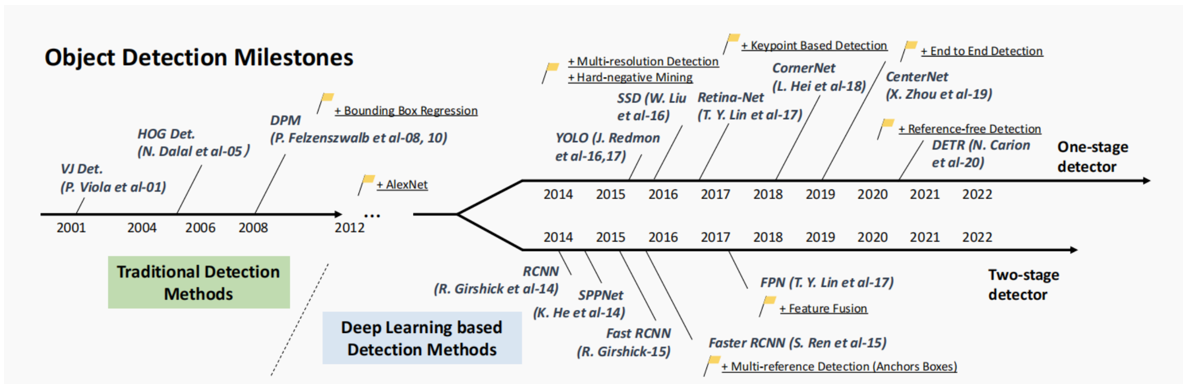

Object Detection Milestones
VJ Det. (P. Viola et al-01)
HOG Det. (N. Dalal et al-05)
DPM (P. Felzenszwalb et al-08, 10)
+ Bounding Box Regression
+ AlexNet
2001
2004
2006
2008
2012
...
Traditional Detection Methods
Deep Learning based Detection Methods
+ Multi-resolution Detection
+ Hard-negative Mining
YOLO (J. Redmon et al-16,17)
SSD (W. Liu et al-16)
Retina-Net (T. Y. Lin et al-17)
+ Keypoint Based Detection
CornerNet (L. Hei et al-18)
+ End to End Detection
CenterNet (X. Zhou et al-19)
+ Reference-free Detection
DETR (N. Carion et al-20)
One-stage detector
2014 2015 2016 2017 2018 2019 2020 2021 2022
RCNN (R. Girshick et al-14)
SPPNet (K. He et al-14)
Fast RCNN (R. Girshick-15)
Faster RCNN (S. Ren et al-15)
+ Multi-reference Detection (Anchors Boxes)
FPN (T. Y. Lin et al-17)
+ Feature Fusion
Two-stage detector

Fig. 1: A road map of object detection[ZCS+23].

The history of research on object detection can date back to 2000[ZCS+23]. Object detection milestones are shown in Fig. 1, which is summarized in [ZCS+23]. In classic object detectors, the sliding-window paradigm, in which a classifier is applied on a dense image grid[LGG+17], is one of the traditional detection methods. VI detectors, HOG detectors and DPM are some traditional detectors. However, with the resurgence of deep learning since 2012, two-stage detectors and one-stage detectors come to the domain. Two-stage detectors is famous for its higher accuracy. In the first stage, it selects the potential candidate proposal regions from the original digital images, which reduces the proposal regions from ∼100k down to ∼1k or ∼2k, generating a sparse set of candidates. In the second stage, it classifies each potential candidate proposal regions. While one-stage detector classifies the potential candidate proposal regions in one stage, which is difficult on dense object detection. Hence, one stage detectors are usually with less accuracy than two-stage detectors. But one-stage detectors are faster and simple than the two-stage detectors. So people may ask a question: could a simple one-stage detector achieve similar accuracy as two-stage detector?

1.2 Motivation

The main obstacle that makes one-stage detector achieve to the similar accuracy as two-stage detector is that the extreme foreground-background class imbalance, which is founded by T.-Y. Lin et al[LGG+17]. The extreme foreground-background class imbalance has negative effects in two aspects: For the first as aspect, the existence of extreme foreground-background class imbalance makes training inefficient as most locations are easy negatives that contribute no useful learning signal. For the second aspect, the easy negatives can overwhelm training and lead to degenerate models.

Some researchers are dedicated to find some solutions to deal with the extreme foreground-background class imbalance problem. Some techniques such as heuristic re-sampling, hard negative mining, loss function reshaping are practical in object detection. Inspired by [LGG+17], we are interested in the design of loss function, named Focal Loss in RetinaNet. The design idea of Focal Loss is so subtle that we want to explore whether we can improve the performance of object detection by modifying the loss function Focal Loss.

2 Literature Review

2.1 Related Work

Loss function measures how well the model matches the data (i.e., the deviation of the predictions from the true labels)[ZCS+23]. A general form of the loss function[ZCS+23] is defined as :

$$\boxed{\begin{aligned} &L(p,p^*,t,t^*) = L_{cls.}(p,p^*) + \beta I(t) L_{loc.}(t,t^*) \\ &I(t) = \begin{cases} 1 & \text{IoU}\{a,a^*\} > \eta \\ 0 & \text{else} \end{cases} \end{aligned}} \tag{1}$$

where t and t^* are the locations of predicted and ground-truth bounding boxes, p and p^* are their category probabilities. $IoU\{a, a^*\}$ is the IoU between the reference box/point a and its ground-truth a^*. γ is an IoU threshold, say, 0.5. If an anchor box/point does not match any objects, its localization loss does not count in the final loss[ZCS+23]. Intuitively the loss function in object detection tasks is the weighted sum of classification loss and localization loss.

Classification loss contains two main categories: Mean-Square-Error(MSE)/L2 loss and Cross Entropy(CE) loss. But in object detection task, CE is more effective since CE is suitable for the classification task. One example is Label Smooth[SVI+16] which can deal with outliers:

$$\boxed{H(q',p) = -\sum_{k=1}^{K} logp(k)q'(k) = (1-\epsilon)H(q,p) + \epsilon H(u,p)} \tag{2}$$

As for localization loss, there is also one case:

$$\boxed{Smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |\text{x}| < 1 \\ |x| - 0.5 & \text{else} \end{cases}} \tag{3}$$

What we are interested in is the loss function Focal Loss mentioned in paper [LGG+17], which can be classified as one of classification losses. The design idea of Focal Loss is based on the traditional cross entropy loss with a weighting factor, i.e. balanced cross entropy, defined as

$$\boxed{CE(p_t) = -\alpha_t log(p_t)} \tag{4}$$

where p_t is the probability of samples being correctly classified. α_t is a weighting factor, which can be calculated as the inverse of the frequency of the true-label class or a tunable hyper-parameter. However, the balanced cross entropy still gives the well-classified examples non-trivial loss weight. The vast negative examples are usually considered as well-classified examples, increasing the total loss. But such training information is less useful than the hard examples, overwhelming the detector during training. By observation, T.-Y. Lin et al proposed Focal Loss defined as

$$\boxed{FL(p_t) = -\alpha_t (1-p_t)^\gamma log(p_t)} \tag{5}$$

where p_t and α_t are the same as the above definition, γ is a tunable focusing parameter. The term $(1-p_t)^\gamma$ is the so called modulating factor. We found that $(1-p_t)^\gamma$ is bounded in [0, 1] and it down-weights the loss assigned to well-classified examples. The following Fig. 2 shows the comparison of cross entropy and focal loss:

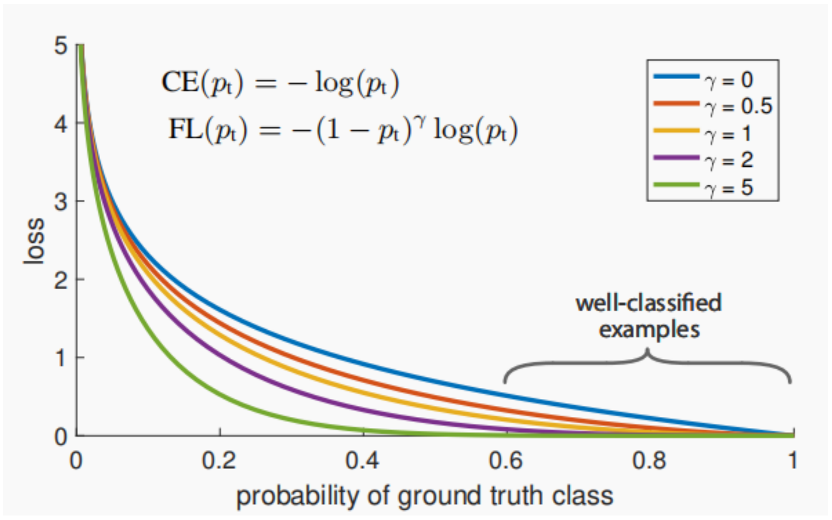


Fig. 2: Comparison of Cross Entropy and Focal Loss[LGG+17]

Focal Loss is used on the output of the classification subnet in RetinaNet[LGG+17]. RetinaNet forms a single FCN comprised of a ResNet-FPN backbone, a classification subnet, and a box regression subnet. The following Fig. 3 shows the one-stage RetinaNet network architecture:

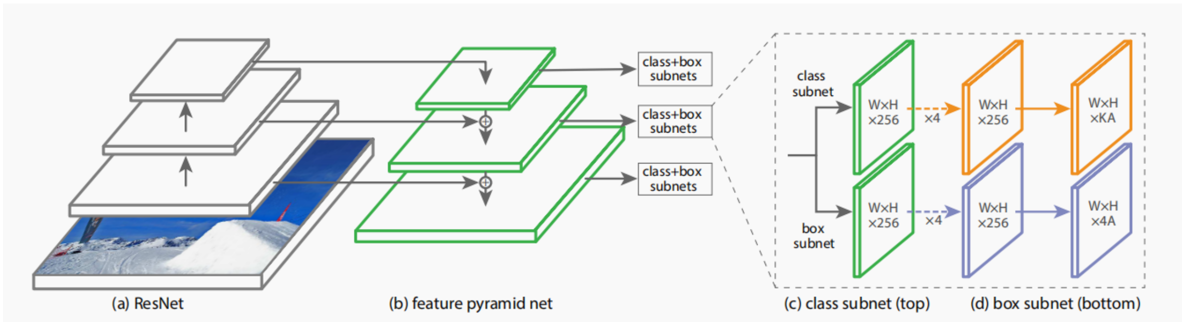


Fig. 3: The one-stage RetinaNet network architecture[LGG+17]

RetinaNet is efficient and accurate: the best model, based on a ResNet-101-FPN backbone, achieves a COCO test-dev AP of 39.1 while running at 5 fps, surpassing the previously best published single-model results from both one and two-stage detectors[LGG+17].

3 Methodology

3.1 Novelty

Although the reshaped standard cross entropy loss will put more focus on hard, misclassified examples during training, there are some possible drawbacks: We found that Focal Loss down-weights all samples, for easy samples, the weight reduction is greater. However, for hard samples, it is not necessary to down-weight them.

Hence the novelty of this project is that we have modified the definition of Focal Loss to hold the following properties:

- down-weights easy samples
- increases the loss weight of hard samples

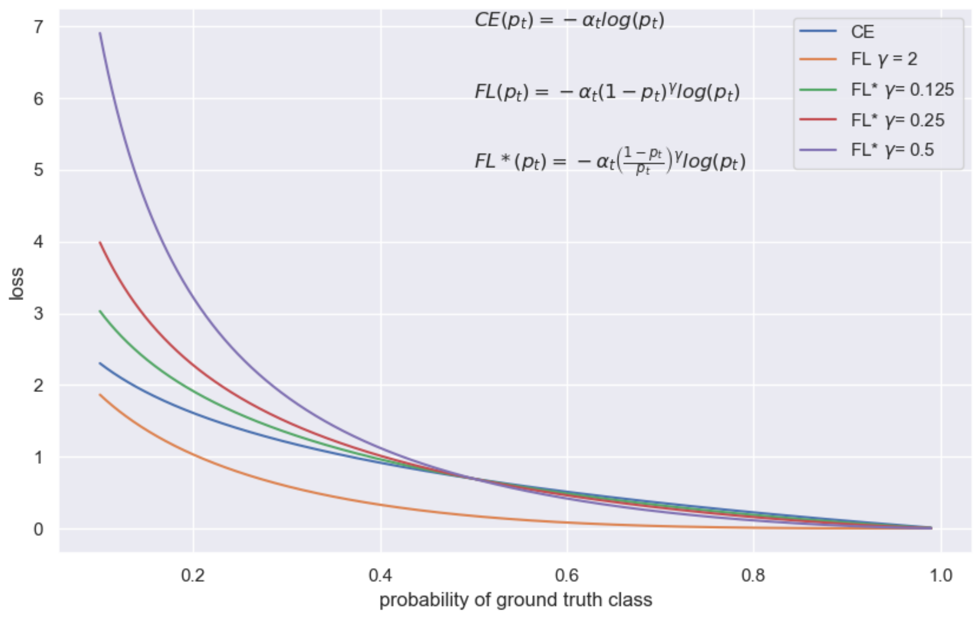


Fig. 4: Comparison of Cross Entropy, Focal Loss and Focal Loss*

3.2 Proposed method

For the system setup, we still use the framework of RetinaNetNet, the only difference is that we have used the modified Focal Loss, denoted as Focal Loss* or FL*, and the settings of other parameters of the RetinaNet remain unchanged.

Focal Loss* is defined as:

$$\boxed{FL^*(p_t) = -\alpha_t \left(\frac{1-p_t}{p_t}\right)^\gamma log(p_t)} \tag{6}$$

where $\left(\frac{1-p_t}{p_t}\right)^\gamma$ is the modified modulating factor.

We observed that when $p_t = 0.5$, the modified modulating factor $\left(\frac{1-p_t}{p_t}\right)^\gamma = 1$, which do not affect the original balanced cross entropy. For the $p_t > 0.5$, the well-classified examples or easy examples, the modified modulating factor $\left(\frac{1-p_t}{p_t}\right)^\gamma < 1$, which down-weighs the loss but do not down-weigh as much as Focal Loss. For the $p_t < 0.5$, the hard examples, the modified modulating factor $\left(\frac{1-p_t}{p_t}\right)^{\gamma}$ > 1, especially when p_t nears to 0, the modified modulating factor is huge, which increases the loss weight. Fig. 4 shows the comparison of cross entropy, focal loss and focal loss*. Here we set $\alpha_t = 1$, $\gamma = 2$ in Focal Loss, $\gamma = 0.125, 0.25, 0.5$ in Focal Loss* respectively, independent variable is probability of ground truth class, i.e the probability of samples being correctly classified, p_t, dependent variable is loss.

4 Experiments

In this section, we introduce the experiments we have conducted at present. Our project can be accessed at https://github.com/0SliverBullet/CS329-Machine-Learning-H-Project. We have found the RetinaNet source code on the GitHub repository(https://github.com/open-mmlab/mmdetection) and completed model training using Focal Loss on the coco dataset used in the paper. We compare the running results with the experimental results in the paper, compare the performance of the pretrained model provided by mmdetection and our trained model on the digital image, test our trained model in SUSTech campus. Note that the models mentioned above use the original Focal Loss as the loss funtiction during training, and our trained model with Focal Loss will be regarded as baseline.

4.1 Experiments Setting

The experiment platform we used in our experiments is the HPC-AI service platform from the Department of Computer Science and Engineering(http://172.18.34.4/). GPU node has the specifications: 2*Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz, cuda = 12.2.

The dataset we use is COCO. COCO (Common Objects in Context) is a widely used dataset for object detection tasks. It is a large-scale dataset that contains a diverse range of object categories and complex scenes. COCO dataset consists of images from everyday scenes, captured in various real-world contexts.

The dataset contains over 200,000 images, with more than 1.5 million labeled instances. These instances belong to 80 different object categories, including common objects like cars, people, animals, and household items. The dataset is carefully annotated with bounding boxes around the objects of interest, providing precise localization information.

One of the unique aspects of the COCO dataset is its complexity. The images in the dataset often contain multiple objects with varying scales, occlusions, and overlapping instances. This makes the dataset challenging for object detection algorithms, as they need to accurately detect and classify objects in cluttered scenes.

In addition to object detection annotations, the COCO dataset also includes other annotations such as segmentation masks, keypoints, and captions. These additional annotations enable researchers to explore more advanced tasks like instance segmentation and pose estimation.

Table 1 summarizes some characteristics of the COCO dataset.

Feature	**Value**
Number of Images	200,000
Number of Instances	1.5 million
Number of Object Categories	80
Average Objects per Image	7.5
Image Resolution	Varied
Complexity	High
Additional Annotations	Segmentation masks, keypoints, captions

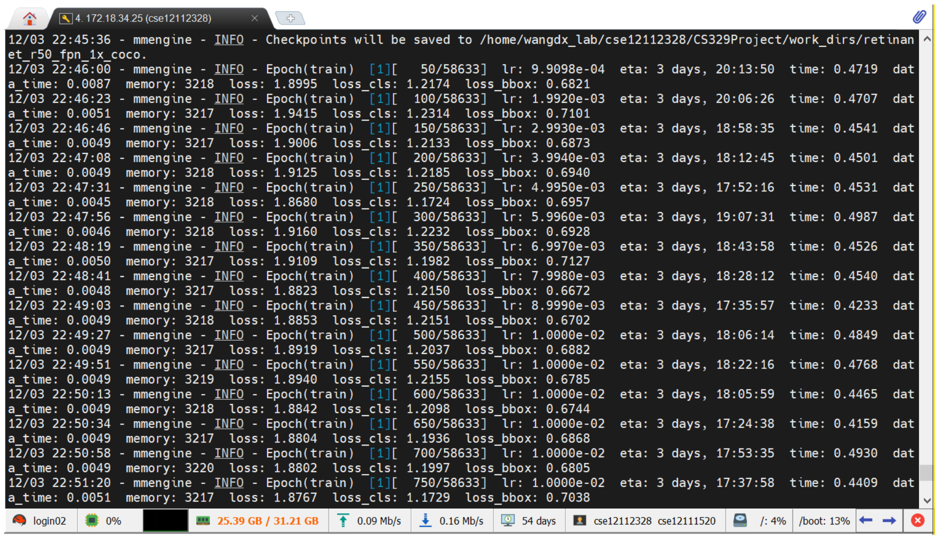
Table 1: Characteristics of the COCO dataset

In our experiments, we use COCO2017train containing 58633 instances as train set, and use COCO2017val containing 5000 instances as test set. Parameter settings is described as following: epochs=12, optimizer='SGD', γ=2.0, α_t=0.25.

4.2 Initial Results

Fig. 5 6 7 8 and Table 2 show the initial results of our project experiments.

Fig. 5 shows the training process in the HPC-AI service platform from the Department of Computer Science and Engineering. Note that we run retinanet_r50_fpn_1x_coco.py. Retinanet is the network name we used, r50 means resNet with 50 layers, fpn means feature pyramid network, 1x means epoch = 12, coco means the train set is coco. Note that training a such model consuming around 3 days, total loss = loss_cls + loss_bbox. Fig 6 shows the classification loss(loss_cls) and localization loss(loss_bbox) fluctuate and decrease as the number of iterations increases.



```
12/03 22:45:36 - mmengine - INFO - Checkpoints will be saved to /home/wangdx_lab/cse12112328/CS329Project/work_dirs/retinan
et_r50_fpn_1x_coco.
12/03 22:46:00 - mmengine - INFO - Epoch(train)  [1][   50/58633]  lr: 9.9098e-04  eta: 3 days, 20:13:50  time: 0.4719  dat
a_time: 0.0087  memory: 3218  loss: 1.8995  loss_cls: 1.2174  loss_bbox: 0.6821
12/03 22:46:23 - mmengine - INFO - Epoch(train)  [1][  100/58633]  lr: 1.9920e-03  eta: 3 days, 20:06:26  time: 0.4707  dat
a_time: 0.0051  memory: 3217  loss: 1.9415  loss_cls: 1.2314  loss_bbox: 0.7101
12/03 22:46:46 - mmengine - INFO - Epoch(train)  [1][  150/58633]  lr: 2.9930e-03  eta: 3 days, 18:58:35  time: 0.4541  dat
a_time: 0.0049  memory: 3217  loss: 1.9006  loss_cls: 1.2133  loss_bbox: 0.6873
12/03 22:47:08 - mmengine - INFO - Epoch(train)  [1][  200/58633]  lr: 3.9940e-03  eta: 3 days, 18:12:45  time: 0.4501  dat
a_time: 0.0049  memory: 3218  loss: 1.9125  loss_cls: 1.2185  loss_bbox: 0.6940
12/03 22:47:31 - mmengine - INFO - Epoch(train)  [1][  250/58633]  lr: 4.9950e-03  eta: 3 days, 17:52:16  time: 0.4531  dat
a_time: 0.0045  memory: 3218  loss: 1.8680  loss_cls: 1.1724  loss_bbox: 0.6957
12/03 22:47:56 - mmengine - INFO - Epoch(train)  [1][  300/58633]  lr: 5.9960e-03  eta: 3 days, 19:07:31  time: 0.4987  dat
a_time: 0.0046  memory: 3218  loss: 1.9160  loss_cls: 1.2232  loss_bbox: 0.6928
12/03 22:48:19 - mmengine - INFO - Epoch(train)  [1][  350/58633]  lr: 6.9970e-03  eta: 3 days, 18:43:58  time: 0.4526  dat
a_time: 0.0050  memory: 3217  loss: 1.9109  loss_cls: 1.1982  loss_bbox: 0.7127
12/03 22:48:41 - mmengine - INFO - Epoch(train)  [1][  400/58633]  lr: 7.9980e-03  eta: 3 days, 18:28:12  time: 0.4540  dat
a_time: 0.0048  memory: 3217  loss: 1.8823  loss_cls: 1.2150  loss_bbox: 0.6672
12/03 22:49:03 - mmengine - INFO - Epoch(train)  [1][  450/58633]  lr: 8.9990e-03  eta: 3 days, 17:35:57  time: 0.4233  dat
a_time: 0.0049  memory: 3218  loss: 1.8853  loss_cls: 1.2151  loss_bbox: 0.6702
12/03 22:49:27 - mmengine - INFO - Epoch(train)  [1][  500/58633]  lr: 1.0000e-02  eta: 3 days, 18:06:14  time: 0.4849  dat
a_time: 0.0049  memory: 3217  loss: 1.8919  loss_cls: 1.2037  loss_bbox: 0.6882
12/03 22:49:51 - mmengine - INFO - Epoch(train)  [1][  550/58633]  lr: 1.0000e-02  eta: 3 days, 18:22:16  time: 0.4768  dat
a_time: 0.0049  memory: 3219  loss: 1.8940  loss_cls: 1.2155  loss_bbox: 0.6785
12/03 22:50:13 - mmengine - INFO - Epoch(train)  [1][  600/58633]  lr: 1.0000e-02  eta: 3 days, 18:05:59  time: 0.4465  dat
a_time: 0.0049  memory: 3218  loss: 1.8842  loss_cls: 1.2098  loss_bbox: 0.6744
12/03 22:50:34 - mmengine - INFO - Epoch(train)  [1][  650/58633]  lr: 1.0000e-02  eta: 3 days, 17:24:38  time: 0.4159  dat
a_time: 0.0049  memory: 3217  loss: 1.8804  loss_cls: 1.1936  loss_bbox: 0.6868
12/03 22:50:58 - mmengine - INFO - Epoch(train)  [1][  700/58633]  lr: 1.0000e-02  eta: 3 days, 17:53:35  time: 0.4930  dat
a_time: 0.0049  memory: 3220  loss: 1.8802  loss_cls: 1.1997  loss_bbox: 0.6805
12/03 22:51:20 - mmengine - INFO - Epoch(train)  [1][  750/58633]  lr: 1.0000e-02  eta: 3 days, 17:37:58  time: 0.4409  dat
a_time: 0.0051  memory: 3217  loss: 1.8767  loss_cls: 1.1729  loss_bbox: 0.7038
```

Fig. 5: Training Process

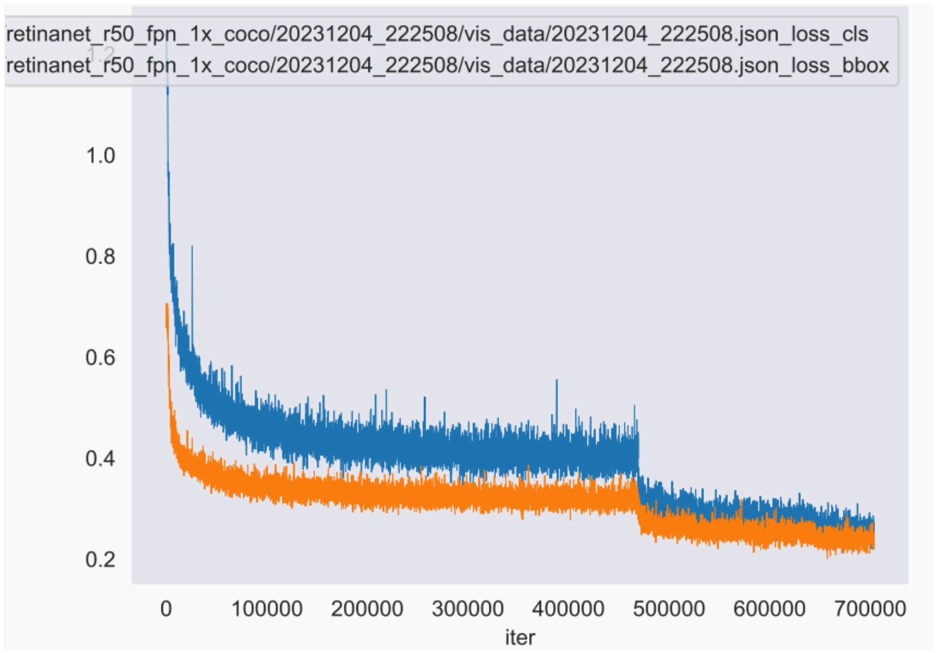


Fig. 6: The loss varies with the number of iterations

Table 2 shows mean average precision of our train model in test set COCO2017val. In the task of object detection, AP (Average Precision) is a commonly used evaluation metric to measure the detection performance of a model on different classes. mAP (mean Average Precision) is the average of AP values and is used to assess the overall detection performance on the entire dataset.

The calculation formula for AP is as follows:

$$\boxed{AP = \frac{1}{n_{\text{pos}}} \sum_{k=1}^{n_{\text{pos}}} \text{TP}_k} \tag{7}$$

Here, n_{pos} represents the number of ground truth objects, and TP_k represents the number of detection results in the top k predictions that have an IoU (Intersection over Union) with the ground truth object greater than a certain threshold. Typically, the IoU threshold is set to 0.5.

mAP is the average of AP values for all classes, and the calculation formula is as follows:

$$\boxed{mAP = \frac{1}{n_{\text{class}}} \sum_{i=1}^{n_{\text{class}}} AP_i} \tag{8}$$

Here, n_{class} represents the number of classes, and AP_i represents the AP value for the ith class.

In addition to mAP, different mAP values can be calculated based on different IoU thresholds. For example, mAP_{50} represents the mAP at an IoU threshold of 0.5, and the calculation formula is as follows:

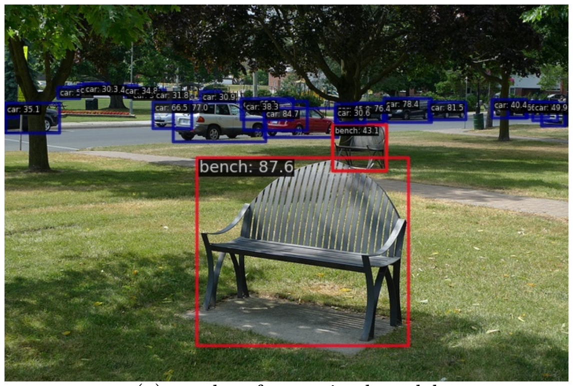
$$\boxed{mAP_{50} = \frac{1}{n_{\text{class}}} \sum_{i=1}^{n_{\text{class}}} AP_{i_{50}}} \tag{9}$$

Here, $AP_{i_{50}}$ represents the AP value for the ith class at an IoU threshold of 0.5.

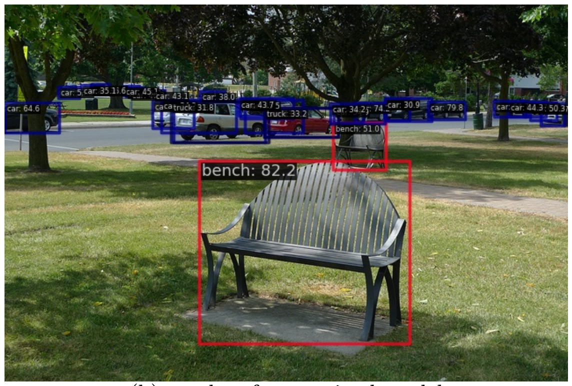
Similarly, mAP_{75} represents the mAP at an IoU threshold of 0.75, while mAP_S, mAP_M, and mAP_L represent the mAP for objects of small, medium, and large sizes, respectively.

detector	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
retinanet_r50_fpn_1x_coco	0.3280	0.5010	0.3500	0.1740	0.3530	0.4380

Table 2: AP of our trained model



(a) results of pretrained model



(b) results of our trained model

Fig. 7: Results Comparison of pretrained model and our trained model

Fig. 7 shows the results comparison of pretrained model and our trained model. Their performances are similar in the object detecion. Fig. 8 shows using our trained model to perform object detection in SUSTech Campus. Note that although the object "chair", "bench", "table" are relative smaller than the objects in Fig. 7, the model still detected them precisely. However, there are some drawbacks. It detected too many chairs in the left side, but actually there are only three chairs. What's more, it detected some chairs as benches, some tables as chairs. As for the obstructed chair on the right, the model is also powerless to detect that it is a partially obstructed chair. These insufficient detections indicate that there is still a lot of room for improvement in the model.

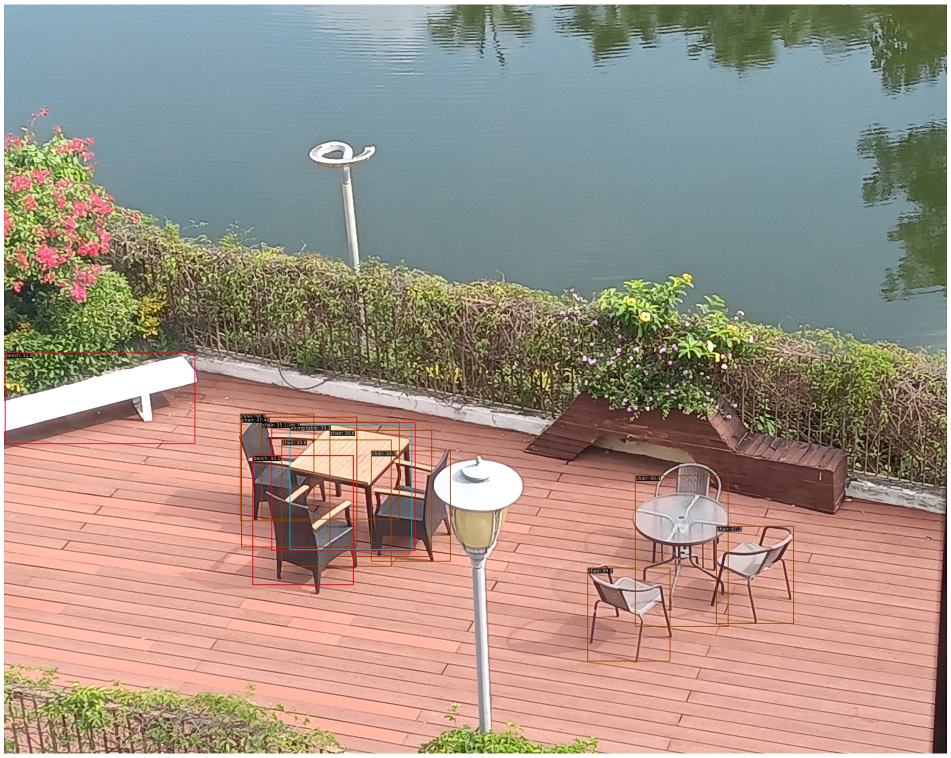


Fig. 8: Object Detection in SUSTech Campus

5 Conclusion

We have finished some works mentined in Goals and Objectives. We expect the modified Focal Loss can improve the performance in objection detecion. The initial results show the baseline of our project. In the future, we will train another model with Focal Loss* to compare with the baseline, expecting the improvement in the performance(AP). Prof. Hao suggested that we can consider safety as metric in the performance comparison. We will take it into consideration in the future works if possible or we have free time.

5.1 Goals and Objectives

- What we have done:
  - Literature review
  - Modified the Focal Loss
  - Run the source code in coco and get a trained model as baseline
- Goal: Performance comparison between original and modified loss functions, expect an improvement in performance(AP).

5.2 Task assignments and Project schedule

Our work schedule for the next three weeks is as follows: In week13, we will find other different datasets and test on them. In week14, we will do performance comparison between original and modified loss functions, tune parameters, analyze, go on to modify loss function if performance is not as our expectation. In week15, we will test our code in SUSTech campus, summarize this project and write the final report. Table 3 shows the whole project schedule since the final project is issued, indicating that our group project is well-organized.

Dates	Tasks	Remarks
Week4	Read the final project requirements	Final project is issued
Week5	Assess difficulty level, decide which job to do and which task to be done:	
Week6	**Camera-based 2D Object Detection**	
Week7	Roughly read the papers in the task we have chosen	
Week8	Choose the one that we are interested in: **RetinaNet**	
Week9	Search for the source code on the GitHub repository	
Week10		
Week11	Thoroughly read the paper: **Focal Loss for Dense Object Detection** Read the source code Download the original dataset coco used in the paper Compare the running results with the experimental results in the paper	
Week12	Modify the definition of loss function and run on **coco** Prepare for the proposal presentation and write proposal report	Final project proposal
Week13	Find other different datasets and test on them	
Week14	Performance comparison between original and modified loss functions tune parameters, analysis, modification on loss function	
Week15	Test our code in SUSTech campus Summarize this project and write the final report	Final project submission DDL

Table 3: Task assignments and Project schedule

References

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[ZCS+23] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.