# 目标检测需求文档

## 对方小组人员说明

第三组：李航、赵凌云、朱卫明、王子谦

## 一、需求概述

目标检测（Object Detection）是计算机视觉中的重要任务，旨在识别图像或视频中的物体，并确定其位置。它结合了分类（确定物体是什么）和定位（确定物体在图像中的位置）两个子任务。

分类任务的目标是识别图像中的物体类别，主要通过卷积神经网络（CNN）提取图像特征，并利用Softmax层输出每个类别的概率，常用的损失函数是交叉熵损失（Cross-Entropy Loss）。定位任务的目标是确定物体的位置，通常用边界框表示。这涉及边界框回归（Bounding Box Regression）来预测边界框的坐标，以及使用IoU（Intersection over Union）评估预测边界框与真实边界框的重叠程度。锚框（Anchor Box）是一种预定义框，用于处理不同尺度和形状的物体。

## 二、需求目标

1.使用yolov5-s模型进行训练

2.使用VOC07+12数据集或能够使用VOC格式数据集进行训练，实现对以下20类别进行目标检测任务。20类别如下：aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor。

例如VOC2007格式的数据集如下

VOCdevkit

--VOC2007

----Annotations

------xml files (eg. 000001.xml)

----JPEGImages

------image files (eg. 000001.jpg)

----ImageSets

------Main

--------train.txt

--------test.txt

--------trainval.txt

3.模型性能在NVIDIA 4060显卡下至少能够达到60fps

4.模型平均精确度mAP至少达到0.5

5.能够提供模型测试数据的可视化结果

6.系统要求：win10或win11

7.环境要求：python3.8 + pytorch1.14

## 三、项目代码结构

--Project -项目目录

----logs -用于存放训练日志

----predict\_out -用于存放预测与测试的结果

------detection\_results -检测结果，一张图片使用一个txt

------ground\_truth -标准结果

------results -测试结果

----model\_data -数据相关

------classes.txt -数据集分类类别

------anchors.txt -备选框大小

------pre\_model.pth -预训练模型

----VOCdevkit -VOC格式数据集，详细目录参考需求目标中VOC格式数据集

----nets -模型相关定义

----utils -工具函数和辅助功能

----predict.py -模型推理脚本

----test.py -模型测试脚本

----train.py -模型训练脚本

----requirement.txt -项目依赖的第三方库列表

----readme.md -项目说明

## 四、模块结构要求

1. 数据处理

需要能够对下载到本地的VOC07+12数据集进行读取，使用pytorch的dataset自行定义。

dataloader.py

import torch

import numpy

import cv2

from PIL import Image

from torch.utils.data.dataset import Dataset

"""

自定义的数据集，继承torch中的Dataset并实现\_\_len\_\_方法与\_\_getitem\_\_方法

"""

class MyDataset(Dataset):

"""

Dataset初始化方法，记录数据读取与建立dataset时需要用到的参数

annotion\_lines: 所有数据的集合，包括每张图像的位置以及框

input\_shape: 读取图像大小

num\_class: 类别数量

等等其他可能用到的参数

"""

def \_\_init\_\_(self, annotation\_lines, input\_shape, num\_classes, …)

"""

返回数据集的大小

"""

def \_\_len\_\_(self):

return len

"""

实现从数据集中获取一组数据的方法

需求：训练时需要能够对数据进行随机增强，而验证时不需要进行数据的随机增强

index: 返回的数据序号

"""

def \_\_getitem\_\_(self, index):

raise NotImplementedError

1. 模型定义

定义用于训练的模型

MyModel.py

from torch import nn

class MyModel(nn.Module):

"""

定义模型结构

"""

def \_\_init\_\_(self,

"""

前向传播

"""

def forward(self, x):

raise NotImplementedError

1. 模型训练

整体训练过程

train.py

'''

加载数据集

相关可能会用到的参数

annotation\_path: 数据集路径

classes\_path: 类别路径

anchors\_path: 先验框路径

'''

# 首先读取数据

# 使用先前自定义的数据集进行建立

train\_dataset = MyDataset(voc,…)

# 之后使用pytorch的DataLoader对数据集进行加载

train\_loader = DataLoader(dataset, batch\_size=1, shuffle=False, sampler=None,...)

'''

加载模型

需要从定义好的模型中进行模型的创建，如果有预训练的权重，加载预训练权重

anchors\_mask： 预选框

num\_classes : 类别数量

phi： 所使用的YoloV5的版本

backbone： 特征提取网络

pretrained： Boolean 是否预训练

input\_shape： 输入形状

'''

model = YoloBody (anchors\_mask, num\_classes, phi, backbone, pretrained=pretrained,

input\_shape=input\_shape)

'''

获取损失函数

input\_shape： 输入形状

anchors\_mask： 预选框

num\_classes : 类别数量

等等其他需要的参数

'''

loss = YoloLoss(anchors, num\_classes, input\_shape, Cuda, anchors\_mask, …)

'''

选择优化器，包括adam或sgd

'''

学习率下降

'''

'''

训练代码

'''

for epoch in range(start, stop)

# 调整学习率

# 进行一轮训练

for iteration, batch in enumerate(train\_loader):

# 清零梯度

# 前向传播

# 计算损失

# 反向传播

# 一轮训练完成后进行验证

for iteration, batch in enumerate(val\_loader):

# 清零梯度

# 前向传播

# 计算损失

# 记录与保存