《YOLO 代码复现框架》

详细分析与设计

V2.0

目　录

第一部分 引言

一、编写目的

编写本设计的目的是为了准确阐述 YOLO 代码复现框架改进系统的具体实现思路和方法,即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。预期读者为项目全体成员，包括用户代表、软件分析人员、开发管理人员和测试人员。

二、项目背景

系统名称：YOLO 代码复现框架

任务提出者：略。

开发者：略。

用户和运行该程序系统的计算中心：略。

三、定义

1、训练类

1）Dataset 类负责加载数据并对数据进行预处理

能够解析并兼容多种形式的数据集，如 COCO, VOC，支持从云端数据流或本地导入图像文件。

1、 loaddata 读取数据。

2、 prepocess 对数据进行预处理，裁剪至统一大小 448*448。

3、 利用 Mosaic, Mixup, Random Erasing, Random Crop, Flip, Rotate, Color Jitter 等算法进行数据增强。

4、 自动划分训练集、测试集和验证集，对于类别不平衡的数据集，支持分层抽样。

2）BackBoneNet 类负责特征提取

它有多个主干网络可供选择，包括 VGG, Darknet, Resnet 等，用于提取图像特征。包含一系列卷积、池化、残差连接、注意力机制等，能够有效地从原始图像中提取多层次的视觉特征，并逐步下采样，将输入图像的空间分辨率从原始尺寸（448*448）降低到更小的特征图（14*14 或 7*7），同时扩大感受野，捕捉更丰富的语义信息。

3）Model 类

这是本系统的核心，它将目标检测任务建模为一个端到端的回归任务，即直接从输入图像预测目标的位置和类别。模型将输入图像划分为 S*S 的网格，每个网格单元预测 B 个边界框（包含中心点坐标、宽度、高度、置信度）以及 C 个类别概率。PyTorch 的 nn.Conv2d、nn.Linear、nn. BatchNorm2d 、 nn. LeakyReLU 以 及 自 定 义 的 Anchor Box generation/matching 或

Anchor-Free assignment 策略，用于本部分是实现。

4）Training 类

本部分将数据与模型结合，是学习目标检测任务的核心阶段。

1、 参数设置部分，包括：learning rate, batch size, epochs, loss weights 等。

2、 优化器设置，包括：Adam， AdamW， SGD 等。

3、 损失计算，包含正样本中心点损失、正样本高宽度损失、正样本置信度损失、负样本置信度损失和正样本分类类别损失。

4、 训练过程支持 GPU 加速，高效进行前向传播、反向传播和梯度更新，全程记录 Loss 曲线，mAP，FPS 等指标，通过 TensorBoard 提供实时的监控和可视化，用户可以据此调整超参数或提前停止训练，以防止过拟合。

5） Evaluate 类

本部分用于评估模型检测能力和性能。

1、通过指定测试数据集、置信度阈值和 NMS（Non-Maximum Suppression）阈值来启动评估任务。

2、遍历测试集，对模型输出的预测框和真实标签进行比对，计算一系列关键指标，如 mAP（mean Average Precision），Precision，Recall，F1 Score，以及 IoU 的分布。

3、对模型进行性能剖析，测量其在不同硬件（CPU, GPU）上的推理速度（FPS）和延迟。

2、部署类

API Interface：用户与本系统交互的平台，通过 Python 和 Web 框架构建高性能 API 服务。该服务接收图像或视频输入，通过模型进行推理，并将检测结果以 JSON 形式返回，包含边界框、类别和置信度。同时提供 API 调用接口，用户可以直接将模型部署到需要目标检测的特定场景中。

四、参考资料

列出有关的参考资料，如：

a. 本项目的经核准的计划任务书或合同、上级机关的批文；

《03 项目库文档@YOLO 代码复现框架》

《You Only Look Once Unified, Real-Time Object Detection》

b. 属于本项目的其他已发表的文件：

《项目需求分析》

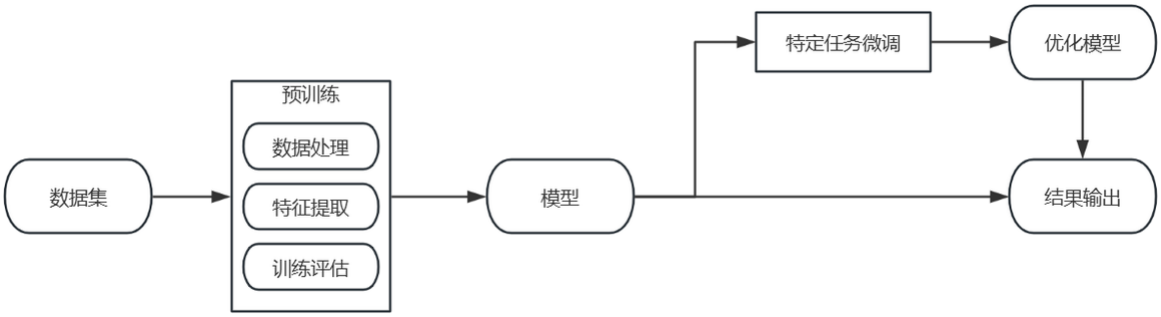
《系统概要设计》

c. 本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《深度学习（PyTorch 版）》

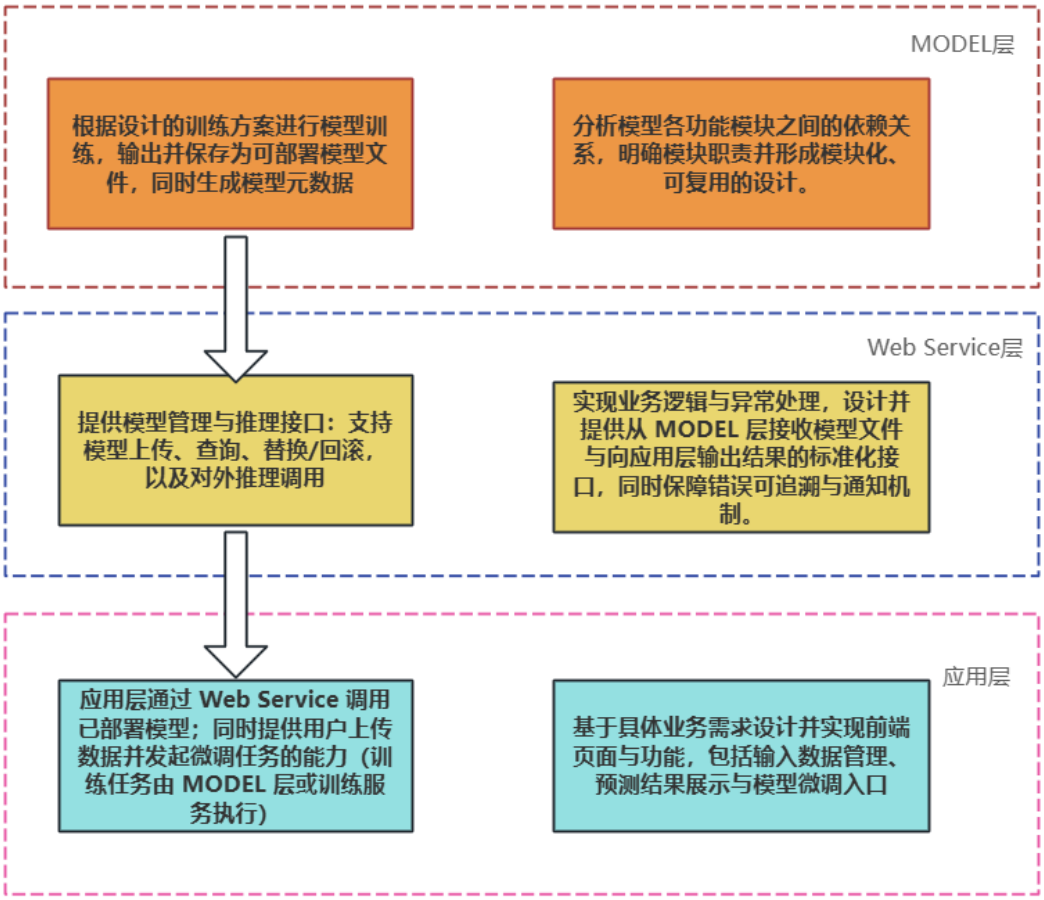
第二部分 项目概述

本项目旨在完成 YOLO 系列目标检测算法的代码复现、训练优化与模型部署，并提供一个可交互的演示系统。项目功能主要包括数据处理、模型训练、模型评估、推理部署以及系统管理五大功能模块。



第三部分 总体设计

一、技术架构设计



1. 各层职责

1.1 Model 层职责

主要职责：设计并实现模型训练流程、构建模型组件（如 backbone、head、loss 等模块），以及导出训练结果为可部署格式（例如 PyTorch 的 .pt）。

输出物：训练脚本、模型文件（.pt）、模型元数据（版本、训练超参、数据集信息、评估指标）。

质量要求：模型需带版本号、哈希/签名，且附带最小依赖说明与示例推理代码。

1.2 Web Service 层职责

主要职责：作为模型的托管与服务层，提供模型上传、模型替换/回滚、模型推理接口（REST/ gRPC）、以及文件接入/输出的标准化接口。实现业务逻辑、异常处理与审计。

功能要点：模型元信息管理（版本、状态）、推理接口（同步/异步）、批量预测支持、模型热替换与灰度发布、日志与监控。

可扩展性：应支持水平扩展（容器化 + 自动伸缩）、GPU/CPU 调度与资源隔离。

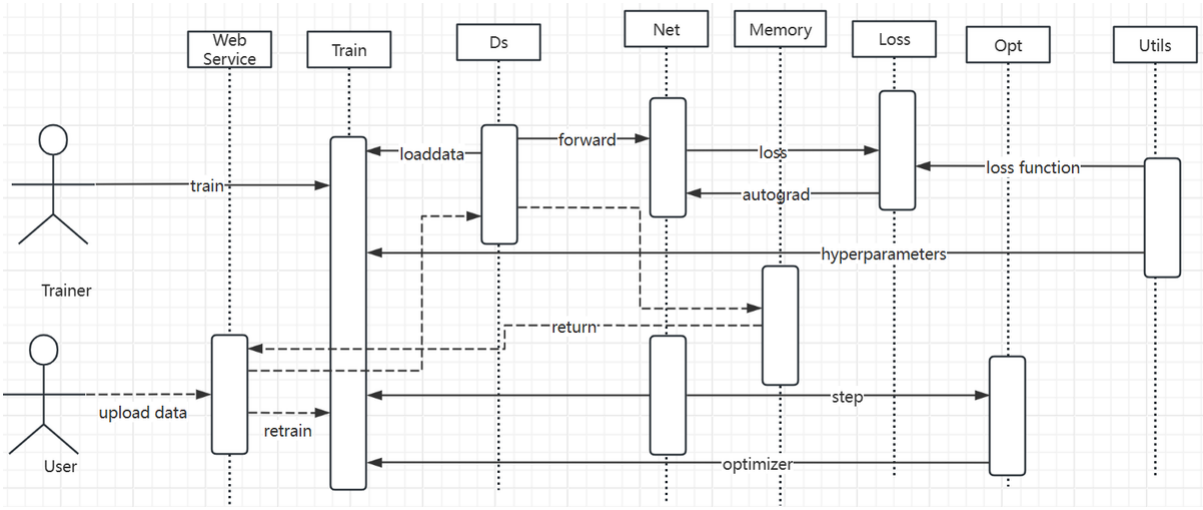
1.3 应用层职责

主要职责：面向最终用户/客户端，提供特定页面与业务功能；通过 Web Service 调用模型推理；支持用户上传数据进行微调（Fine-tune）或触发离线训练任务。

用户流程示例：上传图片或数据 -> 请求推理 -> 显示结果；或上传数据集 -> 请求微调 -> 收到训练完成通知并使用新模型。

二、核心控制流程

1、核心控制流程图



2、核心控制流程说明

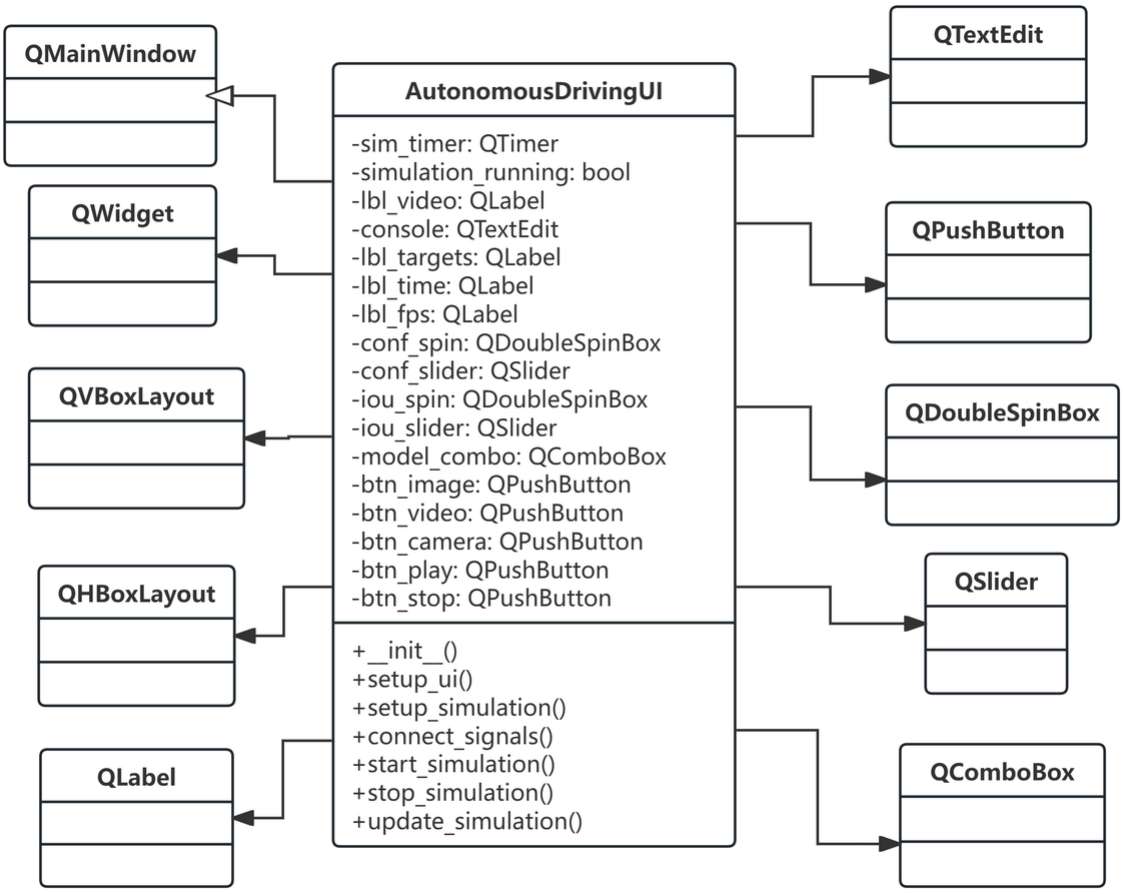
结合上图，针对每一个消息请求，按照消息请求/回复顺序，作如下说明。

消息名称	消息说明
train	训练师（Trainer）通过 Web Service 向 Train 模块发起训练请求，启动训练流程。
load data	Train 模块调用 Ds（Dataset）模块，加载一个批次（batch）的数据用于训练。
forward	Train 模块将数据送入 Net（神经网络）模块，执行前向传播，获得模型预测结果。
loss function	Loss 模块从 Utils 模块获取预定义的损失函数（例如交叉熵损失等）。
loss	Train 模块将 Net 的预测输出和真实标签传递给 Loss 模块，计算两者间的损失值。

autograd	Loss 模块执行反向传播（自动求导），计算出网络中所有参数的梯度。
hyperparameters	Train 模块从 Utils 模块获取学习率等超参数,用于配置优化器。
step	Train 模块调用 Opt（Optimizer）模块，触发参数更新步骤。
optimizer	Opt 模块根据梯度和超参数执行优化算法，更新 Net 模块中的模型权重。
return	子模块（如 Ds, Net）处理完成后，将结果（如数据批次、计算结果）返回给调用者（Train 模块）。
upload data	普通用户（User）通过 Web Service 上传自定义数据集以供后续训练。
retrain	用户上传数据后，通过 Web Service 向 Train 模块发起一个基于新数据的再训练任务。

第四部分 界面设计

一、类图设计



二、详细设计描述

1、AutonomousDrivingUI 主界面类

1.1、职责

目标检测系统的核心控制类，负责整个应用程序的界面布局、功能协调和用户交互。

1.2、核心属性

定时器系统：包含一个模拟定时器和运行状态标志，用于控制检测过程的模拟。

显示组件组：包括视频显示标签、系统日志文本框、目标数量标签、推理时间标签和帧率显示标签。

参数控制组：包含置信度和 IOU 阈值的数字输入框和滑块控件，以及模型选择下拉框。

功能按钮组：包含图像检测、视频检测、摄像头检测、开始检测和停止检测五个功能按钮。

1.3、主要方法

初始化方法：构建界面框架，设置窗口属性和基本配置。

界面设置方法：创建和布局所有 UI 组件，应用样式主题和视觉效果。

模拟环境设置：初始化定时器系统和状态变量，准备模拟运行环境。

信号连接方法：建立所有用户交互事件与处理函数之间的连接关系。

模拟控制方法：启动和停止检测模拟过程，管理定时器运行状态。

数据更新方法：定时回调函数，负责更新模拟数据和刷新界面显示。

2、QMainWindow 主窗口基类

2.1、职责

提供应用程序的主窗口框架，作为整个界面的容器和基础平台。

2.2、核心功能

窗口管理：设置中心窗口部件，管理窗口标题、尺寸和位置。

样式控制：支持 CSS 样式表设置，实现界面美化和主题定制。

框架结构：提供标准的应用程序窗口结构，支持菜单栏、工具栏等扩展组件。

3、QWidget 控件基类

3.1、职责

所有 UI 控件的共同基类，提供基本的窗口功能和通用接口。

3.2、核心功能

布局管理：支持各种布局管理器的设置和应用。

尺寸控制：提供宽度、高度等尺寸属性的设置方法。

样式支持：允许为每个控件单独设置视觉效果和样式规则。

4、 QVBoxLayout 垂直布局管理器

4.1、职责

将子控件按照垂直方向进行排列和组织。

4.2、核心功能

垂直排列：自动将添加的控件从上到下依次排列。

间距控制：支持设置控件之间的垂直间距。

边距管理：可以调整布局与容器边界的距离。

嵌套支持：允许在垂直布局中嵌入其他布局管理器。

5、 QHBoxLayout 水平布局管理器

5.1、职责

将子控件按照水平方向进行排列和组织。

5.2、核心功能

水平排列：自动将添加的控件从左到右依次排列。

弹性空间：支持添加弹性空白区域，实现灵活的空间分配。

对齐控制：提供多种对齐方式的设置选项。

6、 QLabel 标签显示类

6.1、职责

用于显示文本内容、图像信息或动态数据。

6.2、核心功能

文本显示：支持静态文本和动态文本内容的显示。

图像展示：能够显示图片和图形内容。

对齐方式：提供多种文本和内容的对齐选项。

样式定制：支持字体、颜色、背景等视觉样式的设置。

7、 QTextEdit 文本编辑类

7.1、职责

提供多行文本的显示和编辑功能。

7.2、核心功能

多行文本：支持大量文本内容的显示和操作。

内容追加：允许在现有内容基础上添加新的文本。

高度控制：可以设置控件的最大高度限制。

只读模式：支持设置为只读状态，用于信息展示。

8、QPushButton 按钮交互类

8.1、职责

提供用户点击交互功能，触发相应的操作和命令。

8.2、核心功能

点击事件：产生点击信号，触发相应的处理函数。

文本显示：支持按钮文字的设置和显示。

样式定制：可以设置按钮的颜色、边框、背景等视觉效果。

状态反馈：提供按下、悬停、禁用等不同状态的视觉反馈。

9、QDoubleSpinBox 精度输入类

9.1、职责

提供浮点数值的精确输入和调整功能。

9.2、核心功能

数值范围：支持设置最小值和最大值的输入范围。

步长控制：允许设置每次调整的数值变化量。

值变化事件：当数值发生变化时发出信号。

精确输入：支持浮点数的高精度输入和显示。

10、QSlider 滑块调节类

10.1、职责

通过滑动操作来选择和调节数值。

10.2、核心功能

范围设置：支持滑块的最小值和最大值设置。

值设置：可以编程设置滑块的当前值。

值变化事件：当滑块值变化时发出信号通知。

视觉反馈：提供滑块的当前位置和范围的视觉指示。

11、 QComboBox 下拉选择类

11.1、职责

提供从预定义选项中选择值的功能。

11.2、核心功能

选项管理：支持添加和管理选择项列表.

选择设置：可以设置当前选中的项目。

下拉显示：点击时显示所有可选项目。

选择事件：当选择变化时产生相应的事件。

12、QTimer 定时触发类

12.1、职责

提供定时触发功能，用于定期执行特定操作。

12.2、核心功能

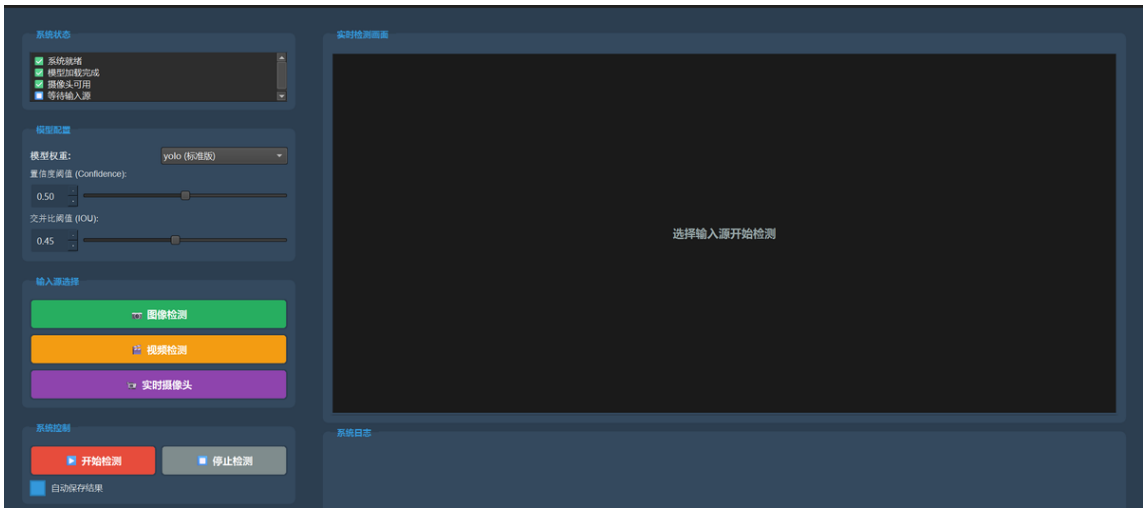
定时触发：按照设定的时间间隔定期发出超时信号。

启停控制：支持启动和停止定时器运行。

间隔设置：可以设置定时触发的时间间隔。

周期性执行：支持连续性的定期操作执行。

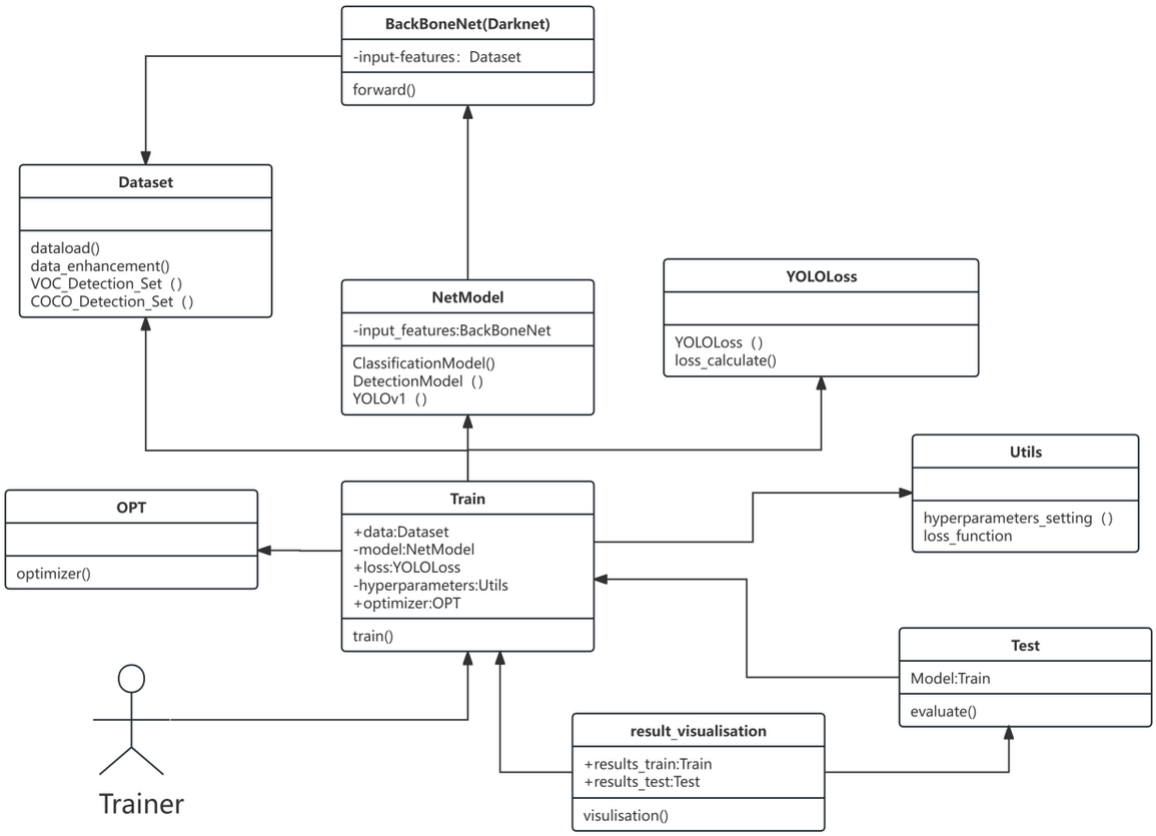
三、界面设计示例



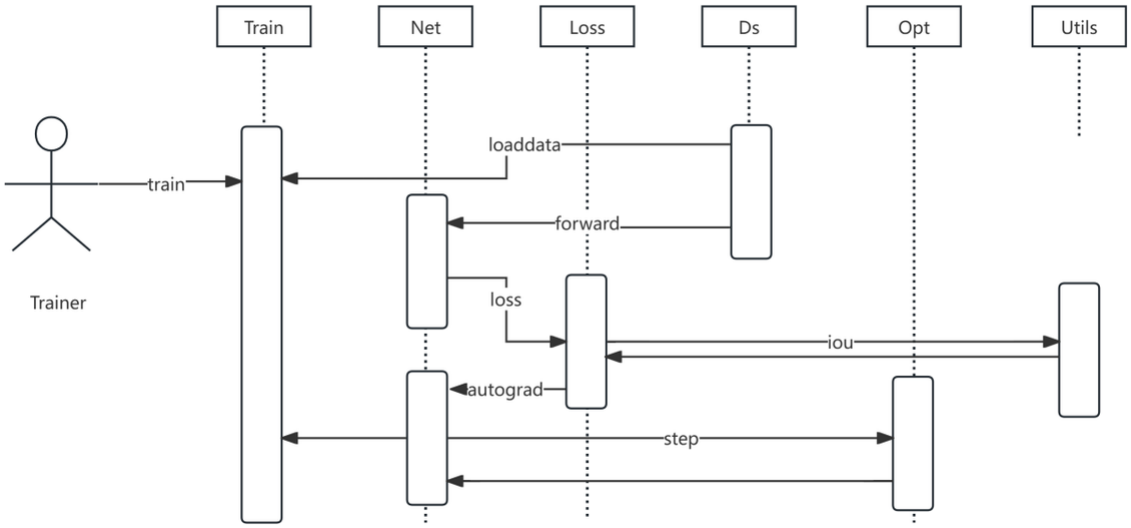
第五部分 单元模块设计

一、模型训练与评估设计

1、类图设计

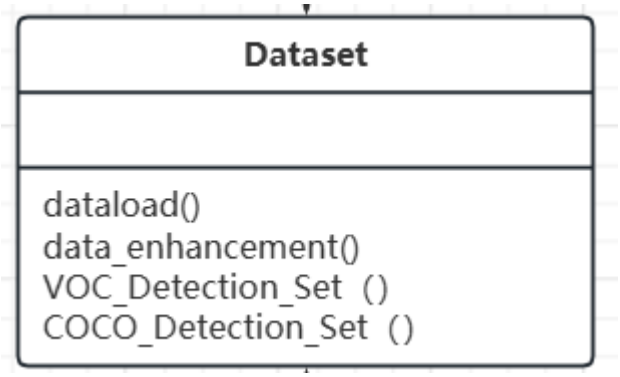


2、时序图



3、类的详细设计描述

3.1 Dataset 数据处理设计

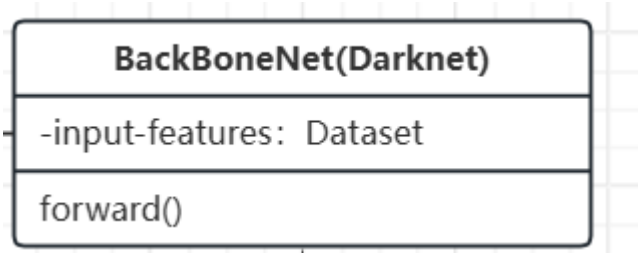


详细描述：

Dataset 数据处理操作 该模块将对选中的数据集下载并对数据集中的图片进行一系列处理
dataload(); 若为公开标准数据集，则通过 torchvision 的 datasets 模块下载；如果是自己使用的数据集，则需要提供文件路径，并在下载时进行一定的数据清洗，剔除不合适的图片。
data_enhancement(); 对传入的图片进行图片旋转、随机裁剪、颜色变换等数据增强方式处理
VOC_Detection_Set（）； 对传入的 VOC 数据集进行专门处理

COCO_Detection_Set（）； 对传入的 COCO 数据集进行专门处理

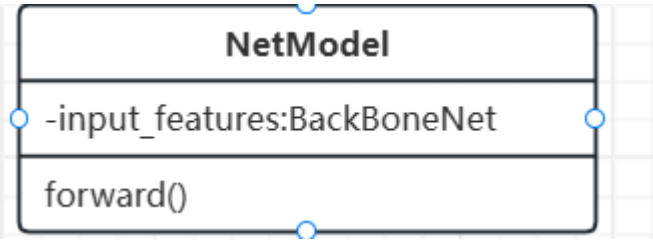
3.2 BackBoneNet 设计



详细描述：

BackBoneNet 图像处理操作 该模块将对 Dataset 处理得到的图像块进行卷积池化处理得到一个张量
forward(); 按照预设的卷积网络结构依次进行卷积与池化操作得到一个张量

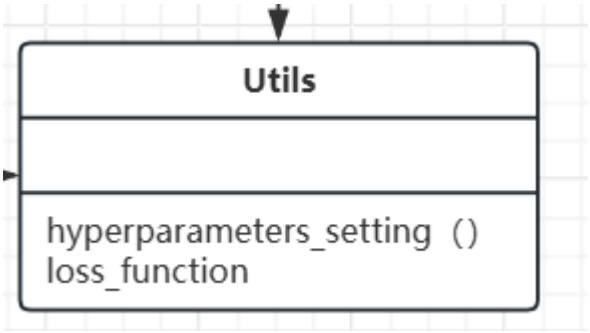
3.3 NetModel 设计



详细描述：

NetModel 图像检测操作 本模块将在 BackBoneNet 输出向量的基础上通过加入数个全连接层实现目标检测功能
ClassificationModel()分类模型 对检测框内的物体做分类，输出每个物体的类别概率
DetectionModel()检测模型 定位目标的位置信息，输出(x, y, w, h, c)，即中心位置坐标、宽度、高度和置信度
YOLOv1() YOLOv1 模型的框架

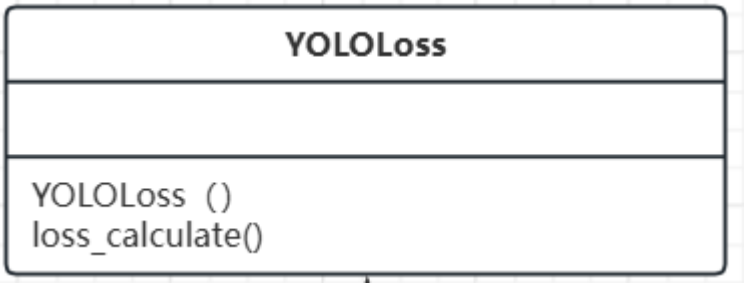
3.4 Utils 设计



详细描述：

Utils 辅助性模块设置 该模块作为辅助性的模块，提供一些辅助性的函数
positive_sample_centre_point_loss（）； 计算正样本中心点损失
negative_sample_confidence_loss（）； 计算负样本置信度损失
positive_sample_high_width_loss（）； 计算正样本高宽度损失
positive_sample_confidence_loss（）； 计算正样本置信度损失
positive_sample_classification_loss（）； 计算正样本分类类别损失
hyperparameters_setting（）； 设定训练所需要的超参数

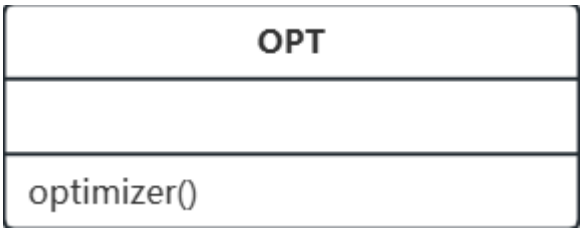
3.5 YOLOLoss 损失函数计算设计



详细描述：

YOLOLoss 损失函数计算 通过调用 utils 中实现的损失函数进行 loss 计算
loss_calculate(); 计算 YOLO 目标检测的损失
YOLOLoss（）； 规定损失函数的类别以及他们的计算方法

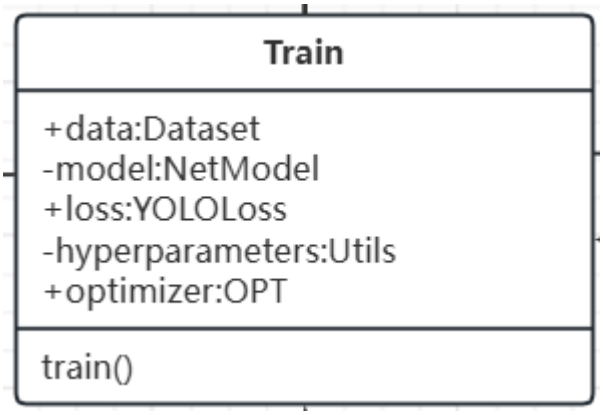
3.6 OPT 优化函数设定



详细描述：

OPT 优化函数设定操作 选择优化函数的类型
optimizer（）； 调用选定的优化函数类型并设定相应的超参数

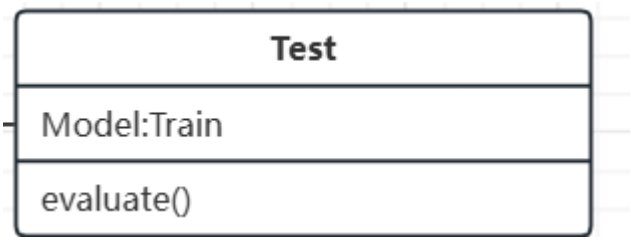
3.7 Train 训练模块设计



详细描述：

Train 训练函数操作 这是训练任务的主要模块，通过调用 NetModel 设定的模型，在 Dataset 提供的数据上进行训练，并用 YOLOLoss 提供的损失函数计算损失，并从 Utils 中获得训练所需要的部分超参数
train（）； 训练进行的主函数，包含训练的输出和训练器的优化

3.8 Test 评估模块设计



详细描述：

Test 训练模块 此模块通过获取 Train 模块训练后的模型通过测试集进行评估
evaluate（）； 将模型设定为评估模式，评估模型的检测性能。

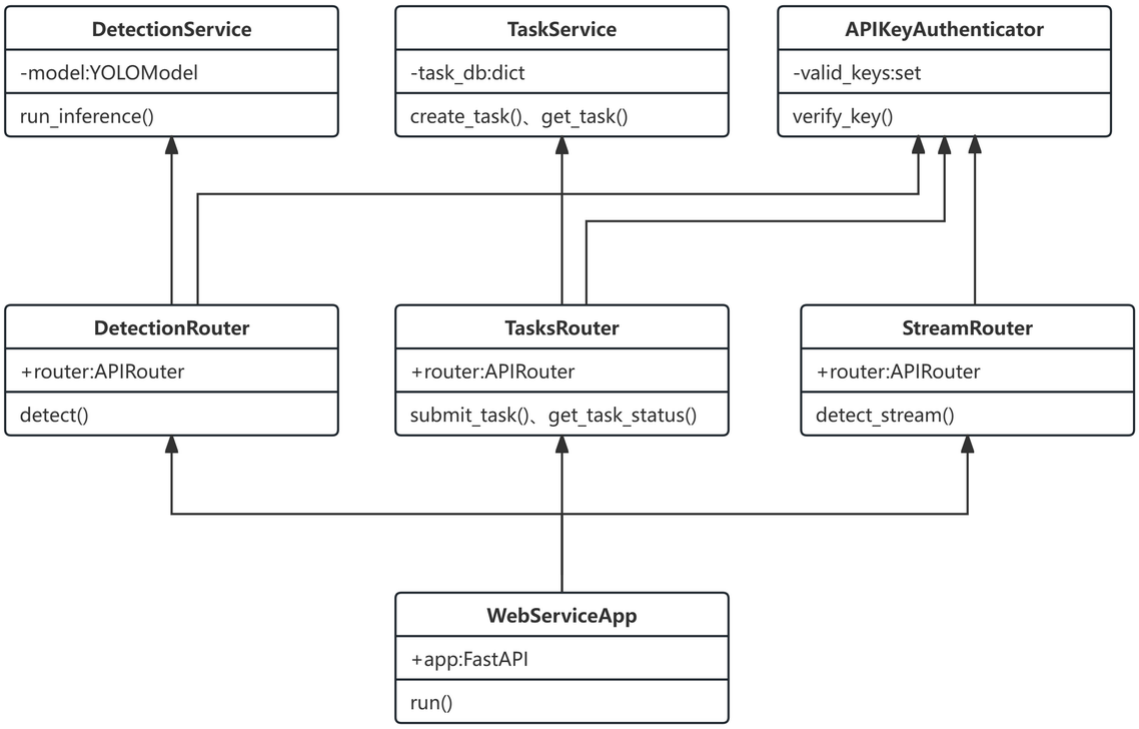
3.9 result_visualisation 结果可视化设计

详细描述：

result_visualisation 结果可视化操作 通过模型输出的最终结果在原图片上显示目标检测的结果
visulisation()； 目标检测结果可视化函数

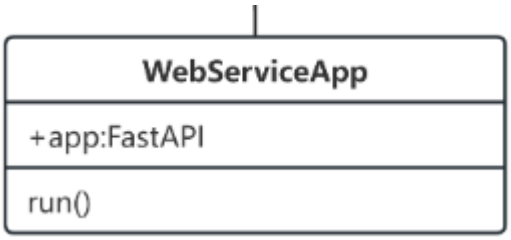
二、WebService

1、 类图设计



2、 类的详细设计描述

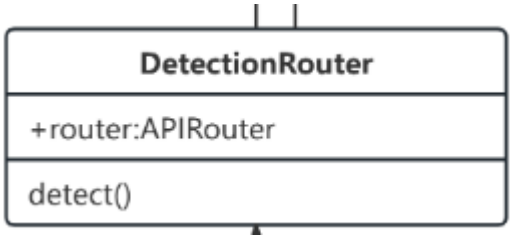
2.1 WebServiceAPP 设计



详细描述：

run（）： 初始化并启动整个 Web 应用服务，作为所有 API 的入口。

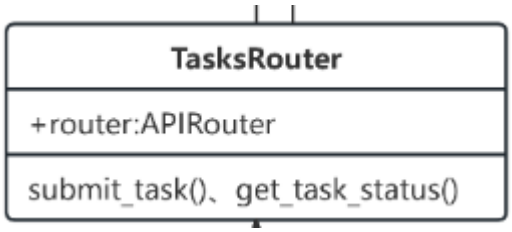
2.2 DetectionRouter 设计



详细描述：

detect（）： 处理同步图形检测请求，调用服务并返回 JSON 结果。

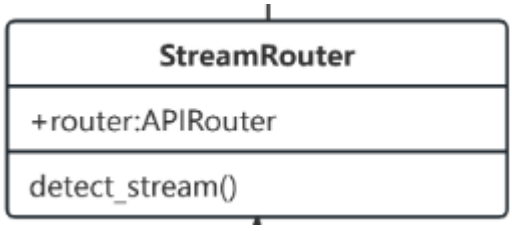
2.3 TasksRouter 设计



详细描述：

submit_task（）： 提交异步任务（如视频分析），并立即返回任务 ID。
get_task_status（）： 根据任务 ID 查询异步任务的状态和结果。

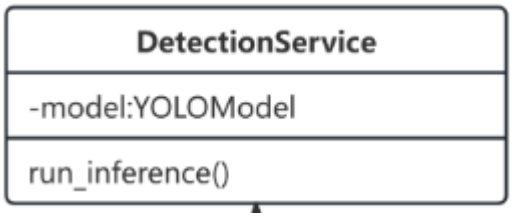
2.4 StreamRouter 设计



详细描述：

detect_stream（）： 通过 WebSocket 处理实时视频流，实现帧的接受和结果的实时回传。

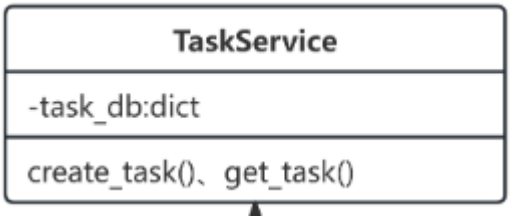
2.5 DetectionService 设计



详细描述：

run_inference（）： 封装核心推理逻辑，供 API 调用。

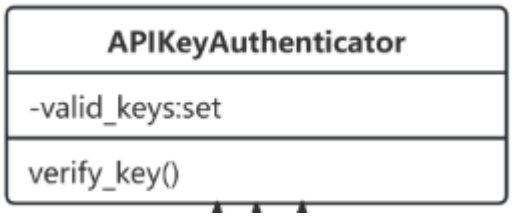
2.6 TaskService 设计



详细描述：

create_task（）： 后台任务的创建。
get_task（）： 后台任务的状态查询。

2.7 APIKeyAuthenticator 设计

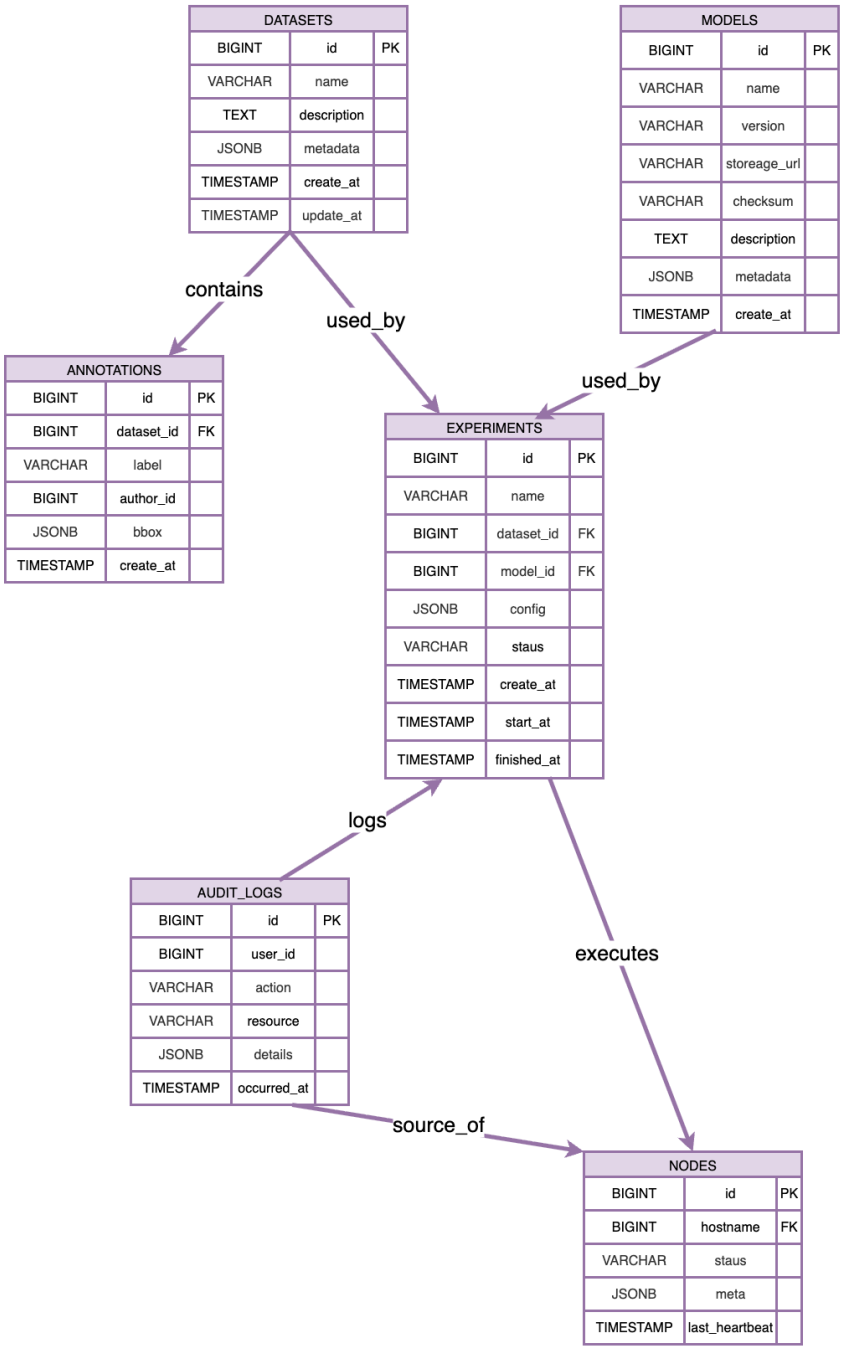


详细描述：

verify_key（）： 作为安全依赖，验证所有传入请求的 API 密钥。

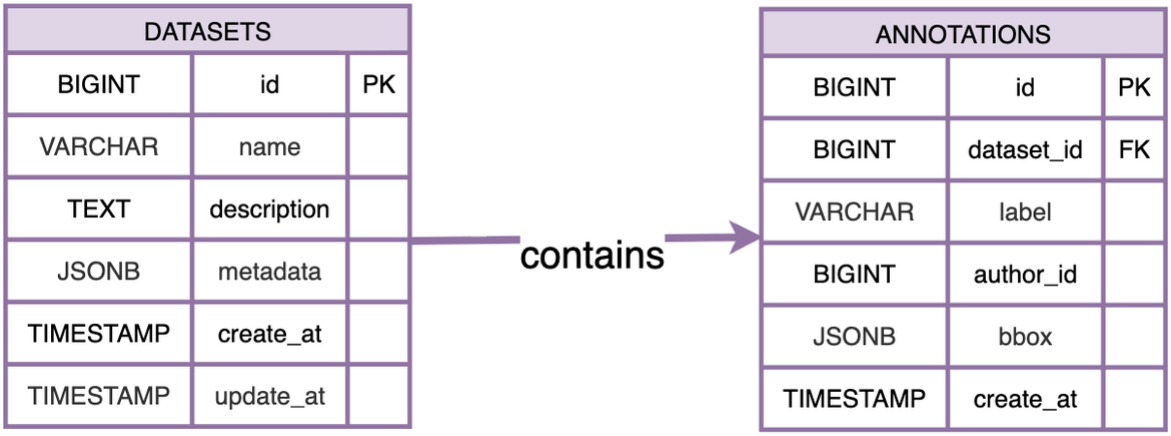
第六部分 数据库设计

一、 数据库整体结构图



数据库整体结构 ER 图

二、数据管理



数据管理 ER 图

数据管理表格清单

序号	表名	注释
1	DATSETS	数据集调用与维护记录
2	ANNOATIONS	图片的标注信息

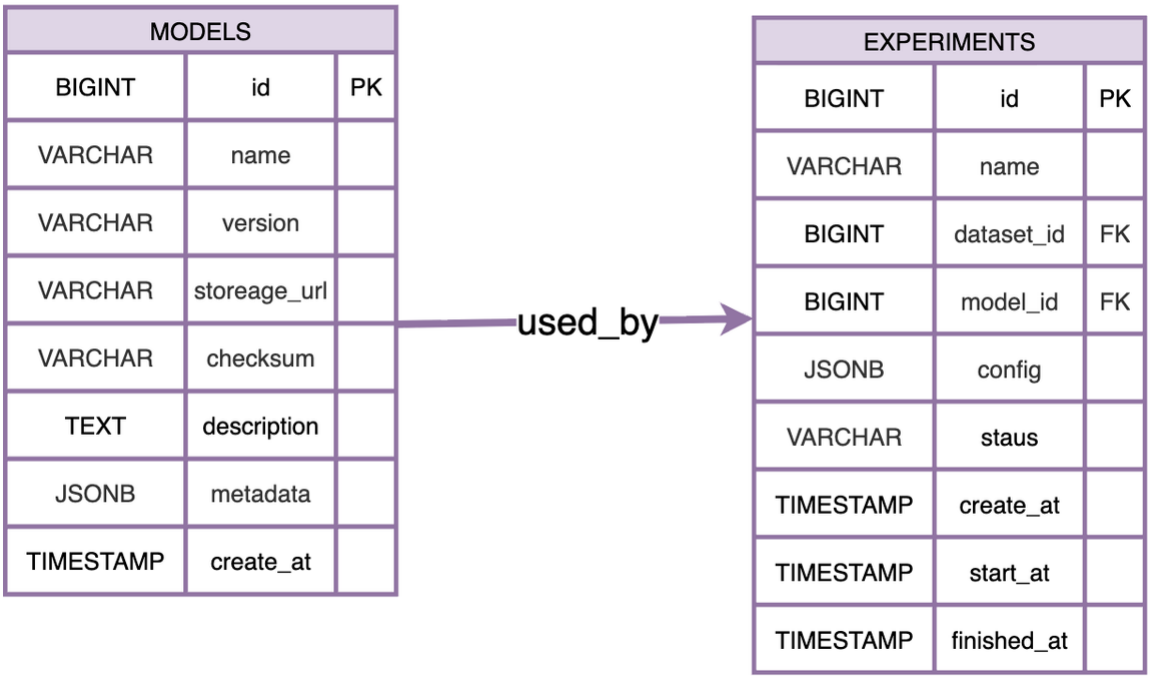
1、DATASETS 表结构

序号	列名	数据类型	注释
1	id	BIGINT (PK)	数据集 ID
2	name	VARCHAR(255)	数据集名称
3	owner_id	BIGINT (FK)	所有者 id
4	format	VARCHAR(32)	数据格式
5	storage_path	TEXT	存储路径
6	status	VARCHAR(32)	状态
7	num_images	INT	图片数量
8	description	TEXT	说明
9	created_at	TIMESTAMP	创建时间

2、ANNOTATIONS 表结构

序号	列名	数据类型	注释
1	id	BIGINT(PK)	标注 ID
2	image_id	BIGINT(FK->images.id)	图片 ID
3	annotator_id	BIGINT(FK->users.id)	标注人 ID
4	boxes	JSON/TEXT	标注框
5	status	VARCHAR(32)	状态
6	created_at	TIMESTAMP	创建时间
7	verified_at	TIMESTAMP NULL	审核时间
8	notes	TEXT	备注

三、模型与实验管理



模型与实验管理 ER 图

模型与实验管理表格清单

序号	名称	注释

1	MODELS	模型
2	EXPERIMENTS	训练与评估实验

1、MODELS 表结构

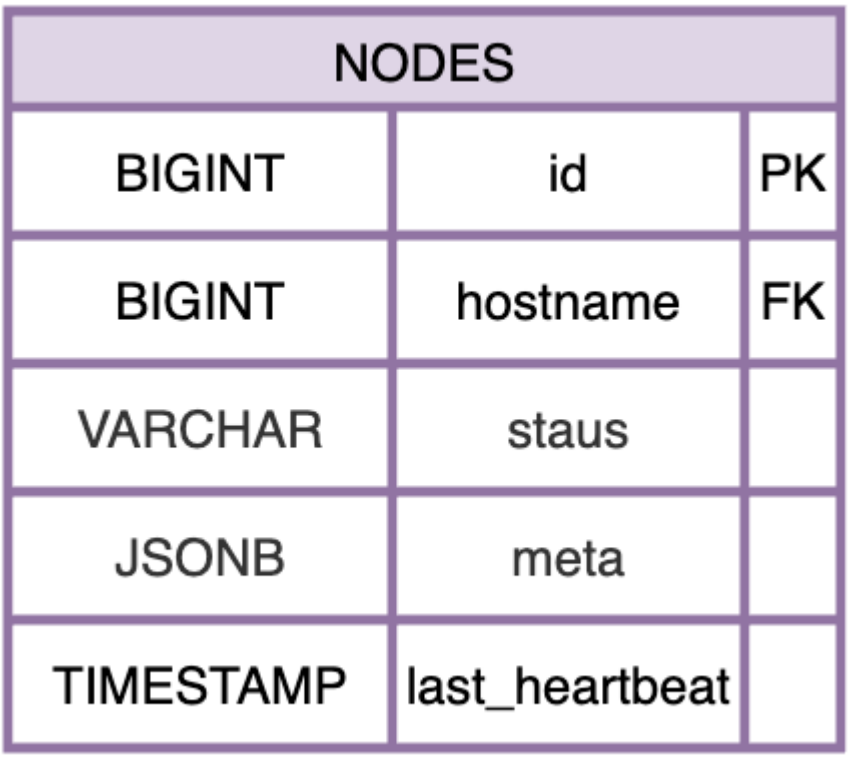
序号	列名	数据类型	注释
1	id	BIGINT (PK)	模型 ID
2	experiment_id	BIGINT (FK)	来源实验 ID
3	version	VARCHAR(64)	版本号
4	format	VARCHAR(32)	格式
5	storage_path	TEXT	存储路径
6	size_bytes	BIGINT	大小（字节）
7	metrics	JSON	评估指标
8	checksum	VARCHAR(128)	校验和
9	signed_by	VARCHAR(64)	签名用户
10	created_at	TIMESTAMP	创建时间

2、EXPERIMENTS 表结构

序号	列名	数据类型	注释
1	id	BIGINT (PK)	实验 ID
2	name	VARCHAR(255)	实验名称
3	user_id	BIGINT (FK）	提交用户 ID
4	dataset_id	BIGINT (FK）	训练数据集 ID
5	config	JSON	训练配置
6	status	VARCHAR(32)	状态
7	metrics	JSON	训练/评估指标

8	checkpoint_path	TEXT	checkpoint 路径
9	start_at	TIMESTAMP	开始时间
10	end_at	TIMESTAMP	结束时间
11	created_at	TIMESTAMP	创建时间

四、执行/调度管理



执行/调度管理 ER 图

业务管理表格清单

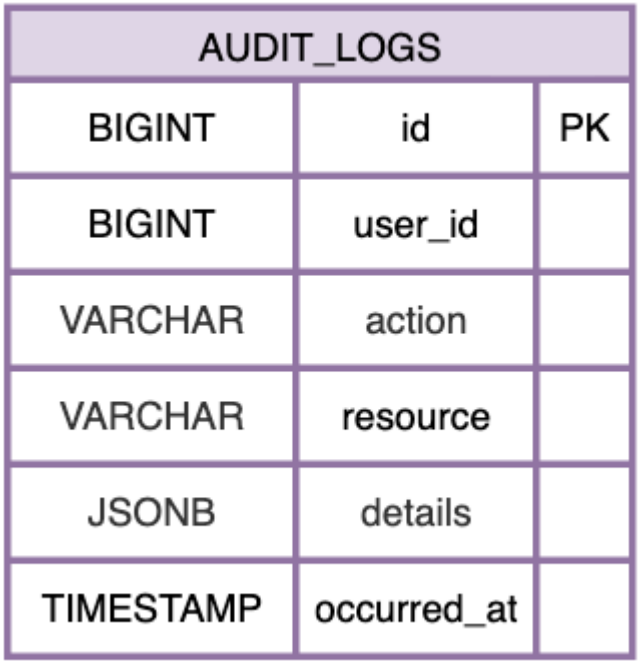
序号	*名称*	*注释*
1	NODES	管理各推理节点的注册、能力与心跳状态。

1、NODES 表结构

序号	列名	数据类型	注释
1	id	BIGINT (PK)	节点 ID

2	node_key	VARCHAR(128)	节点唯一标识
3	hostname	VARCHAR(255)	主机名
4	ip	VARCHAR(64)	IP 地址
5	capabilities	JSON	能力
6	public_key	TEXT	公钥 / 证书信息
7	status	VARCHAR(32)	状态
8	last_time	TIMESTAMP	最后使用时间
9	registered_by	BIGINT (FK)	注册用户 ID
10	created_at	TIMESTAMP	注册时间

五、设计与合规管理



设计与合规管理 ER 图

过户管理表格清单

序号	*名称*	*注释*
1	AUDIT_LOGS	记录关键操作与数据变更

1、AUDIT_LOGS 表结构

序号	列名	数据类型	注释
1	id	BIGINT (PK)	日志 id
2	user_id	BIGINT (FK)	操作用户 ID
3	action	VARCHAR(64)	操作类型
4	target_table	VARCHAR(64)	目标表名
5	target_id	BIGINT	目标记录 ID
6	details	JSON / TEXT	BIGINT
7	ip	VARCHAR(64)	操作 IP
8	created_at	TIMESTAMP	发生时间

第七部分 补充设计和说明

一、编译运行环境设计

1、 服务器端：

操作系统： Ubuntu 20.04 LTS 或更高版本

GPU： NVDIA GPU

CUDA Toolkit： 11.x, cuDNN: 8.x, cuBLAS

CPU: Intel Xeon E5-2680 v4, 建议>=8 核，支持数据预处理和多进程调度

内存： 64GB DDR4 RAM 或更高

存储： NVMe SSD 阵列，支持高速数据读取、模型存储和日志记录

Python >=3.8

包管理：Conda 或 Pip

2、 本地开发环境：

操作系统： Windows 10/11, macOS, Ubuntu

GPU: 建议配置 NVIDIA GPU

Python >=3.8

IDE: VS Code 或 PyCharm

版本控制： Git

容器化： Docker Desktop

3、 客户端：

Web 端： 推荐 Chrome/Firefox/Edge 最新版浏览器

二、包路径与 WEB 目录结构设计

1、 包路径设计

项目根目录 ：

README.md: 项目说明、安装指南、使用说明。

requirements.txt: Python 依赖包列表。

config/: 存放各种配置文件。

config.yaml: 主要系统配置（如数据库连接、模型服务地址）。

data_config.yaml: 数据集路径、格式、预处理配置。

model_config.yaml: 模型结构、默认预训练权重配置。

train_config.yaml: 默认训练参数配置模板。

deploy_config.yaml: 模型导出和推理服务配置。

data/: 存放示例数据集、下载的预训练模型权重。

datasets/: 用户上传或平台提供的数据集。

dataset_name_1/: 数据集目录。

images/: 图像文件。

annotations/: 标注文件（COCO.json, VOC.xml, YOLO.txt）。

pretrained_weights/: 存放下载的 YOLO 预训练权重。

src/: 存放平台核心源代码。

__init__.py

data/: 数据处理相关模块。

__init__.py

dataset.py: Dataset 类定义，数据加载器。

transforms.py: 数据增强算法实现。

utils.py: 数据处理辅助函数。

models/: YOLO 模型结构定义。

__init__.py

yolo_v1.py, yolo_v5.py, yolo_v8.py ... (按版本划分的模型文件)。

backbones/: 骨干网络定义。

necks/: Neck 结构定义。

heads/: Detection Head 定义。

losses/: 损失函数实现。

__init__.py

yolo_losses.py: 包含中心点、尺寸、置信度、类别损失。

train/: 模型训练相关模块。

__init__.py

train.py: 训练主循环、参数管理。

optimizer.py: 优化器与学习率调度器配置。

metrics.py: 训练指标计算（mAP, Loss）。

logger.py: 日志记录与 TensorBoard 集成。

evaluate/: 模型评估模块。

__init__.py

evaluate.py: 评估主流程。

metrics.py: 评估指标计算（mAP, Precision, Recall）。

report.py: 报告生成。

export/: 模型导出模块。

__init__.py

export.py: 统一导出接口。

onnx_exporter. py, tflite_exporter. py, coreml_exporter. py, tensorrt_exporter.py...

deploy/: 模型部署与服务化模块。

__init__.py

predict.py: 推理接口。

api_server.py: RESTful API 服务实现。

model_manager.py: 模型加载与版本管理。

utils/: 通用工具函数。

__init__.py

visualize.py: 可视化相关函数（如绘制边界框）。

tools.py: 通用辅助函数。

cli/: 命令行接口脚本。

__init__.py

main.py: CLI 入口。

data_cli.py: 数据集相关 CLI 命令。

train_cli.py: 训练任务 CLI 命令。

eval_cli.py: 评估 CLI 命令。

deploy_cli.py: 部署 CLI 命令。

web/: Web 前端项目.

public/: 静态资源（HTML, Favicon）。

src/: 源代码。

App.vue (或其他主入口)。

components/: UI 组件。

views/: 页面组件。

services/: API 服务调用。

utils/: 前端辅助函数。

package.json: npm/yarn 配置文件。

docker/: Dockerfile 及相关配置。

Dockerfile: 构建平台镜像。

docker-compose.yml: 编排容器。

notebooks/: Jupyter notebooks，用于模型演示、数据分析、实验。

tests/: 单元测试和集成测试。

docs/: 项目文档。

2、Web 路径结构（API 服务）：

api/v1/: API 版本 v1。

datasets/: 数据集相关 API。

POST /datasets: 上传数据集。

GET /datasets/{dataset_id}: 获取数据集信息。

POST /datasets/{dataset_id}/preprocess: 配置并启动预处理。

models/: 模型管理 API。

GET /models/architectures: 获取可用模型架构列表。

GET /models/pretrained: 获取可用预训练权重列表。

POST /models/train: 提交训练任务。

GET /models/train_jobs/{job_id}: 获取训练任务状态。

POST /models/export: 提交模型导出任务。

POST /models/deploy: 提交推理服务部署任务。

inference/: 推理服务 API。

POST /inference/predict: 提交推理请求（接受图像/视频数据）。

GET /inference/models/{model_id}/status: 查询模型服务状态。

evaluate/: 评估 API。

POST /evaluate/run: 提交评估任务。

GET /evaluate/jobs/{eval_job_id}: 获取评估任务状态和结果。

analytics/: 数据分析 API。

GET /analytics/detection_summary: 获取检测结果统计。

三、Python 集成代码库及其版本

1、深度学习框架：

PyTorch: torch>=2.0.0 (包括 torch.nn, torch.optim, torch.utils.data, torchvision 等)。

torchvision: torchvision>=0.15.0 (用于数据变换、模型定义、预训练模型)。

2、模型优化与部署：

ONNX Runtime: onnxruntime>=1.14.0 (用于 ONNX 模型推理)。

ONNX: onnx>=1.14.0 (用于 ONNX 模型格式)。

TensorRT: (NVIDIA 官方提供，需独立安装，通常通过 pycuda 间接使用或与 PyTorch 集成)。

Core ML Tools: coremltools>=6.0 (用于模型转换为 Core ML 格式)。

3、数据处理与增强：

OpenCV: opencv-python>=4.7.0 (用于图像/视频读写、基础图像处理、部分增强操作)。

NumPy: numpy>=1.23.0 。

Pillow (PIL): Pillow>=9.5.0 (图像处理)。

Albumentations: albumentations>=1.3.0 (提供更丰富、高性能的数据增强算法，可与 PyTorch 集成)。

4. 训练与评估：

TensorBoard: tensorboard>=2.10.0 (用于训练过程可视化)。

Scikit-learn: scikit-learn>=1.2.0 (用于评估指标计算，如 Precision, Recall)。

SciPy: scipy>=1.10.0 (科学计算库，可能用于某些优化或评估辅助)。

Tqdm: tqdm>=4.65.0 (用于显示进度条)。

5、部署与服务化：

Flask: Flask>=2.2.0 或 FastAPI: fastapi>=0.90.0 (用于构建 RESTful API 服务)。

Uvicorn: uvicorn>=0.20.0 (FastAPI 的 ASGI 服务器)。

Gunicorn: gunicorn>=20.0.0 (WSGI HTTP 服务器，用于 Flask 等)。

CLI 与 GUI：

Argparse / Click: Python 标准库或第三方库，用于构建 CLI。

Streamlit / Gradio: streamlit>=1.18.0 / gradio>=3.20.0 (用于快速构建 Web GUI)。

6、配置管理：

PyYAML: PyYAML>=6.0 (用于解析 YAML 配置文件)。

7、其他工具：

Pandas: pandas>=1.5.0 (用于数据分析和报告生成)。

Matplotlib / Seaborn: matplotlib>=3.7.0, seaborn>=0.12.0 (用于生成数据分析图表)。