

**本科毕业设计（论文）**

**基于深度学习的病虫害检测系统的设计与实现**

|  |  |
| --- | --- |
| 学院（部、中心）： | 软件学院 |
| 专 业： | 软件工程 |
| 班 级： | 软件工程2103 |
| 学生姓名： | 谢曼思 |
| 学 号： | 2214424316 |
| 指导教师： | 李晨 |

2025年06月

（注：X均为阿拉伯数字，如2025年06月）

摘 要

论文摘要由摘要正文、关键词等部分组成。

摘要正文是论文的高度概括，是全文的缩影，是长篇论文不可缺少的组成部分。要求用中、英文分别书写，一篇摘要不少于400字。

摘要正文内容一般包括：从事这项研究工作的目的和意义；完成的工作（作者独立进行的研究工作及相应结果的概括性叙述）；获得的主要结论（这是摘要的中心内容）。

摘要中一般不宜使用图、表、化学结构式、非公知公用的符号和术语，不标注引用文献编号。

如果论文的主体工作得到了有关基金资助，应在摘要第一页的页脚处标注：本研究得到某某基金（编号：）资助。（五号）

……

**关 键 词**：XXX；XXX；XXX；XXX；XXX

关键词由3～5个词组成。关键词应从《汉语主题词表》中摘选，当《汉语主题词表》的词不足以反映主题时，可由申请人设计关键词，但须加注。每一关键词之间用分号分开，最后一个关键词后不打标点符号。由申请人设计的关键词，须在该关键词的右上角标注\*，并在该页的页脚处注明“\*表示非汉语主题词”。

**中文摘要格式规范：**

1. **标题部分：**居中编排“摘 要”二字（三号宋体），字间空两个半角字符。段落设置：段前2行，段后1行，行距1.2倍。
2. **摘要正文部分：**中文每段首行缩进2字符起排，段与段之间不空行（小四号，中文宋体，数字及英文Times New Roman）。段落设置：行距1.2倍。
3. **关键词部分：**关键词位于摘要正文下方，空一行，左对齐顶格编排“**关 键 词**”三字（小四号宋体，**加粗**），每两字间距为一个半角字符，后接冒号（全角），其后为具体关键词（小四号宋体，不加粗）。每一关键词之间用分号（全角）隔开，最后一个关键词后不打标点符号。段落设置：行距1.2倍。

ABSTRACT

The key parts in drip irrigation facilities are emitters. The structural design parameters of emitters can directly affect its performance and the function of the whole drip irrigation system ……

1. Because……

2. Only ……

3. To support ……

**KEY WORDS**: XXX; XXX; XXX; XXX; XXX

**英文摘要撰写要求如下：**

（1）用词准确，符合英文语法，内容须与中文摘要一致；

（2）关键词按相应专业的标准术语写出，尽量从《英语主题词表》中摘选；

（3）如果论文的主体工作得到了有关基金资助，应用英文在摘要第一页的页脚处标注：本研究得到某某基金（编号：）资助。

**英文摘要格式规范：**

1. **标题部分：**居中编排“ABSTRACT”（三号Times New Roman）。段落设置：段前2行，段后1行，行距1.2倍。
2. **摘要正文部分：**每段开头左对齐顶格编排，段与段之间空一行（小四号Times New Roman）。段落设置：行距1.2倍。
3. **关键词部分：**关键词位于摘要正文下方，空一行，左对齐顶格编排“**KEY WORDS**”（小四号Times New Roman，**加粗**），后接冒号（半角），其后为具体关键词（小四号Times New Roman，不加粗）。每个关键词组的第一个字母大写，其余为小写，每一关键词之间用分号（半角）隔开，最后一个关键词后不打标点符号。段落设置：行距1.2倍。例如：

**KEY WORDS**: Drip irrigation emitter; RP&M; Hydraulics; Labyrinth flow channel

目 录

摘 要.............................................................................................................................................................I

ABSTRACT................................................................................................................................................II

[1 绪论（一级标题） 1](#_Toc197617896)

[1.1 标题2（二级标题） 1](#_Toc197617897)

[1.1.1 标题3（三级标题） 1](#_Toc197617898)

[1.1 研究背景和意义 1](#_Toc197617899)

[1.2 国内外农业病虫害识别研究现状 2](#_Toc197617900)

[1.3 论文研究内容 3](#_Toc197617901)

[1.3.1 标题3（三级标题） 3](#_Toc197617902)

[1.4 论文组织结构 4](#_Toc197617903)

[1.4.1 标题3（三级标题） 4](#_Toc197617904)

[2 相关理论与技术 5](#_Toc197617905)

[2.1 基于深度学习的目标检测算法 5](#_Toc197617906)

[2.2 系统开发Web技术 7](#_Toc197617907)

[2.2.1 后端开发技术 7](#_Toc197617908)

[2.2.2 前端开发技术 8](#_Toc197617909)

[2.3 本章小结 9](#_Toc197617910)

[3 基于改进YOLOv8n的番茄叶片病虫害检测模型 10](#_Toc197617911)

[3.1 应用场景分析与问题描述 10](#_Toc197617912)

[3.2 基于 GhostNet 网络的轻量化改进 10](#_Toc197617913)

[3.3 基于模型剪枝的轻量化改进 15](#_Toc197617914)

[3.4 实验设计和结果分析 17](#_Toc197617915)

[3.4.1 实验数据集及实验环境 17](#_Toc197617916)

[3.4.2 评价指标 17](#_Toc197617917)

[3.4.3 对比试验分析 17](#_Toc197617918)

[3.4.4 消融实验分析 18](#_Toc197617919)

[3.5 本章小结 19](#_Toc197617920)

[4 番茄叶片病虫害识别系统的需求与设计 20](#_Toc197617921)

[4.1 系统需求分析 20](#_Toc197617922)

[4.1.1 系统功能性需求分析 20](#_Toc197617923)

[4.1.2 系统非功能性需求分析 21](#_Toc197617924)

[4.2 系统概要设计 22](#_Toc197617925)

[4.2.1 系统整体体系结构 22](#_Toc197617926)

[4.2.2 系统功能模块设计 23](#_Toc197617927)

[4.2.3 识别搜索模块设计 24](#_Toc197617928)

[4.2.4 识别记录管理模块设计 25](#_Toc197617929)

[4.2.5 账户管理模块设计 27](#_Toc197617930)

[4.3 数据库设计 28](#_Toc197617931)

[4.4 本章小结 30](#_Toc197617932)

[5 番茄叶片病虫害识别系统的实现与测试 31](#_Toc197617933)

[5.1 标题2 31](#_Toc197617934)

[5.1.1 标题3 31](#_Toc197617935)

[6 结论与展望 32](#_Toc197617936)

[6.1 标题2 32](#_Toc197617937)

[6.1.1 标题3 32](#_Toc197617938)

[7 XXXXXXX（标题1） 33](#_Toc197617939)

[7.1 标题2 33](#_Toc197617940)

[7.1.1 标题3 33](#_Toc197617941)

[8 XXXXXXXX（标题1） 34](#_Toc197617942)

[8.1 标题2 34](#_Toc197617943)

[8.1.1 标题3 34](#_Toc197617944)

[9 XXXXXXXXX（标题1） 35](#_Toc197617945)

[9.1 标题2 35](#_Toc197617946)

[9.1.1 标题3 35](#_Toc197617947)

[10 XXXXXXXXXX（标题1） 36](#_Toc197617948)

[10.1 标题2 36](#_Toc197617949)

[10.1.1 标题3 36](#_Toc197617950)

[11 XXXXXXXXXXX（标题1） 37](#_Toc197617951)

[11.1 标题2 37](#_Toc197617952)

[11.1.1 标题3 37](#_Toc197617953)

[12 结论与展望 38](#_Toc197617954)

[12.1 标题2 38](#_Toc197617955)

[12.1.1 标题3 38](#_Toc197617956)

[致 谢 39](#_Toc197617957)

[参考文献 40](#_Toc197617958)

[附 录 44](#_Toc197617959)

主要符号表

|  |  |
| --- | --- |
|  | 灌水器流量偏差系数 |
|  | 管道内径/mm |
|  | 灌水器流道当量直径/mm |
|  | 管长/m |
|  | 迷宫流道单元个数/个 |
|  | 灌水器流量/L·h-1 |
|  | 灌水器额定流量/L·h-1 |
| *Re* | 雷诺数 |
|  | 灌水器流量标准偏差 |
|  | 流体的运动粘性系数 |
|  | 流态指数 |

如果论文中使用了大量的物理量符号、标志、缩略词、专门计量单位、自定义名词和术语等，应将全文中常用的这些符号及意义列出。如果上述符号和缩略词使用数量不多，可以不设专门的主要符号表，但在论文中出现时须加以说明。

论文中主要符号应全部采用法定单位，特别要严格执行GB3100～3102—93有关“量和单位”的规定。单位名称的书写，可以采用国际通用符号，也可以用中文名称，但全文应统一，不得两种混用。

缩略词应列出中英文全称。

主要符号表正文统一左缩进一个字符。

符号表排序方法：先按拉丁字母大写、小写排序，再按希腊字母大写、小写排序，如下表所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| **Ⅰ** | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| **Ⅱ** | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| **Ⅲ** | Α | Β | Γ | Δ | Ε | Ζ | Η | Θ | Ι | Κ | Λ | Μ | Ν | Ξ | Ο | Π | Ρ | Σ | Τ | Υ | Φ | Χ | Ψ | Ω |  |  |
| **Ⅳ** | α | β | γ | δ | ε | ζ | η | θ | ι | κ | λ | μ | ν | ξ | ο | π | ρ | σ | τ | υ | φ | χ | ψ | ω |  |  |

**Ⅰ**：拉丁字母大写；**Ⅱ**：拉丁字母小写；**Ⅲ**：希腊字母大写；**Ⅳ**：希腊字母小写。

本部分内容非强制性要求，如果论文中所用符号不多，可以省略《主要符号表》。章的MathType的章标记（打印前将其字体颜色变为白色，在打印预览中看不见即可）：

# 绪论（一级标题）

绪论相当于论文的开头，它是三段式论文的第一段（后二段是本论和结论）[[1]](#footnote-1)。绪论与摘要写法不完全相同，摘要要写得高度概括、简略，绪论可以稍加具体一些，文字以1000字左右为宜。绪论一般应包括以下几个内容：

（1）为什么要写这篇论文，要解决什么问题，主要观点是什么。

（2）对本论文研究主题范围内已有文献的评述（包括与课题相关的历史的回顾，资料来源、性质及运用情况等）。

（3）说明本论文所要解决的问题，所采用的研究手段、方式、方法。明确研究工作的界限和规模。

（4）概括论文的主要工作内容。

## 标题2（二级标题）

### 标题3（三级标题）

#### 1）标题4

##### （1）标题5

###### a）标题6

###### b）标题6

（a）标题7

**下一行为分节符。每章末尾均有分节符，请勿删除分节符行。**

## 研究背景和意义

在我国农业发展过程中，番茄作为种植面积广、经济价值高的重要蔬菜作物，扮演着促进农民增收、保障蔬菜供应的重要角色。番茄生长周期长、管理要求高，且容易受到多种病虫害的侵袭，轻则影响产量与品质，重则导致大面积减产甚至绝收。为了有效识别和控制病虫害，保障番茄产业的可持续发展，番茄病虫害检测成为了种植管理中不可或缺的重要环节。目前传统的检测方法主要有两种：一是依靠农户或农技人员的经验进行目测识别，但此方式受限于个人经验差异大、判断标准不一，容易出现识别误差；二是通过常规施药手段进行预防，虽然使用农药可以一定程度上避免番茄出现病虫害现象，但农药滥用会严重污染与破坏辣椒生长的环境，这违背了绿色高质量农产品的理念，也不利于番茄可持续化发展。

番茄叶片作为病虫害最早发生和最易观测的部位，其表面纹理、色泽及形态特征的变化往往能直接反映病害的种类和程度。与健康叶片相比，受病虫害侵扰的区域在纹理图像中通常表现为局部像素值突变或不规则的边缘特征，为基于图像识别技术进行检测提供了重要依据。由于自动检测系统具有低人力成本、检测速度快、可大规模推广等优点，近年来已成为农业信息化领域的重要研究方向。

传统的基于图像处理的检测方法，如Sobel算子、Canny边缘检测器等，虽在理想环境下能够实现较高精度，但在实际应用中，番茄叶片常因拍摄角度、光照变化、背景干扰等因素导致边界模糊、噪声增加，使得常规方法难以稳定工作。此外，基于固定阈值的分割方法虽然简单直观，但对光照、色差的变化极为敏感，容易出现漏检或误检的情况。面对复杂自然环境下的实际检测需求，仅依靠传统图像处理方法已难以满足高准确率与强鲁棒性的要求。近年来，随着深度学习技术，特别是卷积神经网络（Convolutional Neural Network, CNN）的快速发展，目标检测与图像分割领域取得了突破性进展。CNN能够通过多层次特征提取，自适应捕捉病虫害区域的复杂纹理和形态变化，显著提升了病虫害检测的准确性与泛化能力。

本文结合深度学习技术，围绕番茄叶片病虫害识别展开研究，构建基于YOLOv8模型的自动检测系统，该系统可以帮助农户快速了解发病番茄叶片病虫害信息并找到解决方法，在患病初期就及时精准防治、对症下药，从而保障植株的健康、优质和高产。这项技术不仅能带来显著的经济效益，还为其他农作物害虫识别研究提供了参考，有助于促进智慧农业建设与乡村振兴政策的推进。

## 国内外农业病虫害识别研究现状

农业病虫害识别技术一直是国内外农业工程领域的研究热点，其发展大致经历了三个阶段：依赖人工目视经验、应用传统机器学习方法进行识别，以及采用深度学习技术实现识别。

现今，卷积神经网络（Convolutional Neural Network，CNN）成为特征提取的主流工具。例如，ZENG等人[4]通过CNN提取图像特征，对水果和蔬菜进行分类训练，在自建数据集上实现了95.6%的准确率；毛锐等[5]针对小麦病害检测任务，优化了深度残差网络（ResNet-50）结构，采用卷积核拆解和下采样延迟策略，并改进了ROI Align以降低特征量化误差，最终使Faster-RCNN在小麦条锈病、黄矮病等识别任务中平均精度达到了98.74%。Helong Yu 等（2022）针对番茄病虫害问题，采用了6种网络并改进了果蝇优化算法，同时优化了网络的学习率。实验结果表明，IResNet50在识别7种番茄害虫时的准确率达到94.4%，优于其他模型。石晨宇等（2021）提出了一种结合常规卷积和Ghost瓶颈层的轻量化模型，减少了模型的复杂度，并使用GELU激活函数替代ReLU，提高了识别精度。李文逵等（2022）则使用改进的轻量级MobileNetV2网络进行植物叶片图像识别，其模型准确率达到了91.41%。

尽管现有卷积神经网络通过增加层数不断提升模型性能，但这也导致了计算资源和存储需求的显著增长，使得模型训练速度变慢、参数量庞大，限制了其在移动端和嵌入式应用中的推广。

为了解决这一问题，单阶段目标检测算法如SSD（Single Shot Multibox Detector）[6]和YOLO（You Only Look Once）系列[7]因其高效性和实时性，广泛应用于自然环境下的目标识别与定位任务。比如，CHEN等人[8]基于MobileNetv3优化了SSD主干网络，并引入Soft-NMS抑制冗余检测框，在自建数据集上实现了95.50%的mAP，单张图像推理时间缩短至30ms；KARTHIK等[9]则结合Swin Transformer与残差变形网络，提出双轨网络，有效提升了葡萄叶片病害分类的准确率，达到98.6%。

近年来，为进一步适应实际应用需求，研究者们纷纷致力于轻量化网络的设计。LI等[10]通过使用ShuffleNetv2重构骨干网络，并用Add操作替代Concat，显著降低了计算量、参数量及模型体积，同时保持98.8%的准确率；XIAO等[11]结合通道剪枝与注意力机制，在减少模型体积至原有的约21%（从36.5M减少到7.8M）的同时，维持了88.6%的准确率。谭厚森等[12]进一步优化了YOLOv8的部分模块，例如通过引入Partial Convolution（PConv）和改良的空间金字塔池化（SPPF），在香梨检测数据集上使平均精度均值提高了0.4–0.5个百分点，检测速度分别提升了34%与24.4%。

此外，LI等[13]还针对YOLOv7-tiny模型进行了结构优化，引入轻量特征提取模块与解耦头设计，并应用模型剪枝技术，实现了在保持良好检测性能的同时，将计算参数、模型复杂度和体积分别压缩到原网络的37.8%、34.1%和40.7%。

## 论文研究内容

### 标题3（三级标题）

#### 1）标题4

##### （1）标题5

###### a）标题6

###### b）标题6

（a）标题7

## 论文组织结构

### 标题3（三级标题）

#### 1）标题4

##### （1）标题5

###### a）标题6

###### b）标题6

（a）标题7

# 相关理论与技术

## 基于深度学习的目标检测算法

深度学习是机器学习的重要分支，利用卷积神经网络（CNN）可以自动进行病虫害的分类和定位。随着技术的不断进步，基于深度学习的目标检测算法逐渐分为一阶段和双阶段两种模式。双阶段目标检测算法将检测过程分为两个步骤：首先，通过区域提议算法生成候选框，然后通过卷积神经网络提取目标特征并进行分类。该方法的代表性算法包括R-CNN及其改进版本，如SPP-Net、Fast R-CNN、 Faster R-CNN 和 R-FCN 等都是在 R-CNN 的基础上进行了改进和优化。

相比之下，一阶段目标检测算法则简化了这一过程，直接使用卷积神经网络对图像进行一次传递，从而省略了候选框生成步骤。这种方法不仅减少了模型的复杂性，还提高了计算效率，常见的一阶段目标检测算法有 SSD 和 YOLO（You Only Look Once）模型系列。

YOLO模型通过将输入图像划分为多个网格，每个网格负责预测该区域内的目标，这使得YOLO能够在单次前向传递中同时完成目标定位、候选框生成和分类任务，从而显著提高了检测速度。YOLO系列还采用回归的方法进行边界框预测，并且通过卷积神经网络（CNN）直接从特征图中提取目标特征，避免了传统双阶段方法中需要候选框生成的复杂步骤。尽管精度和速度方面表现出色，但在处理小目标时的精度有所欠缺，尤其是在同一网格区域内存在多个目标时，可能会出现漏检现象。为了克服这些问题，后续版本对模型进行了多方面的改进，例如引入批量归一化（BN）提高训练稳定性、引入先验框机制改善目标定位、以及多尺度训练提升模型对不同尺寸目标的检测能力。同时，YOLO系列还加强了特征融合和多层次信息的提取，使得模型在各种实际应用中表现更加出色。

YOLOv8是YOLO系列目标检测模型的第8个主要版本，由Ultralytics团队开发[15]。它继承了YOLO系列模型的一阶段检测机制，根据参数量和复杂度的不同，YOLOv8共包含5个模型，分别是YOLOv8n、YOLOv8s、YOLOv8m、YOLOv8l以及YOLOv8x。YOLOv8n网络结构主要由骨干网络(Backbone)、颈部网络(Neck)、和检测头(Head)组成，如图2-1所示。

输入端(input)采用马赛克数据增强、自适应灰度填充等对输入图片进行预处理。Backbone部分由Conv，C2fckbone、和SPPF构成，负责提取图像多尺度特征。Neck部分借鉴了PAN和FPN设计思想，将骨干网络获取的特征通过上采样进行细节特征增强，并通过下采样整合深层语义信息，从而实现不同尺度特征的高效融合。Head部分采用解耦头(decoupled-head)结构，实现分类和回归的解耦处理，分别优化目标分类和定位精度，并引入Task Aligned Assigner标签分配策略，对分类分数和回归分数进行加权匹配，以提高正样本的匹配精度，同时使用Anchor-free模型，简化复杂度的同时提升了检测器的泛化性。

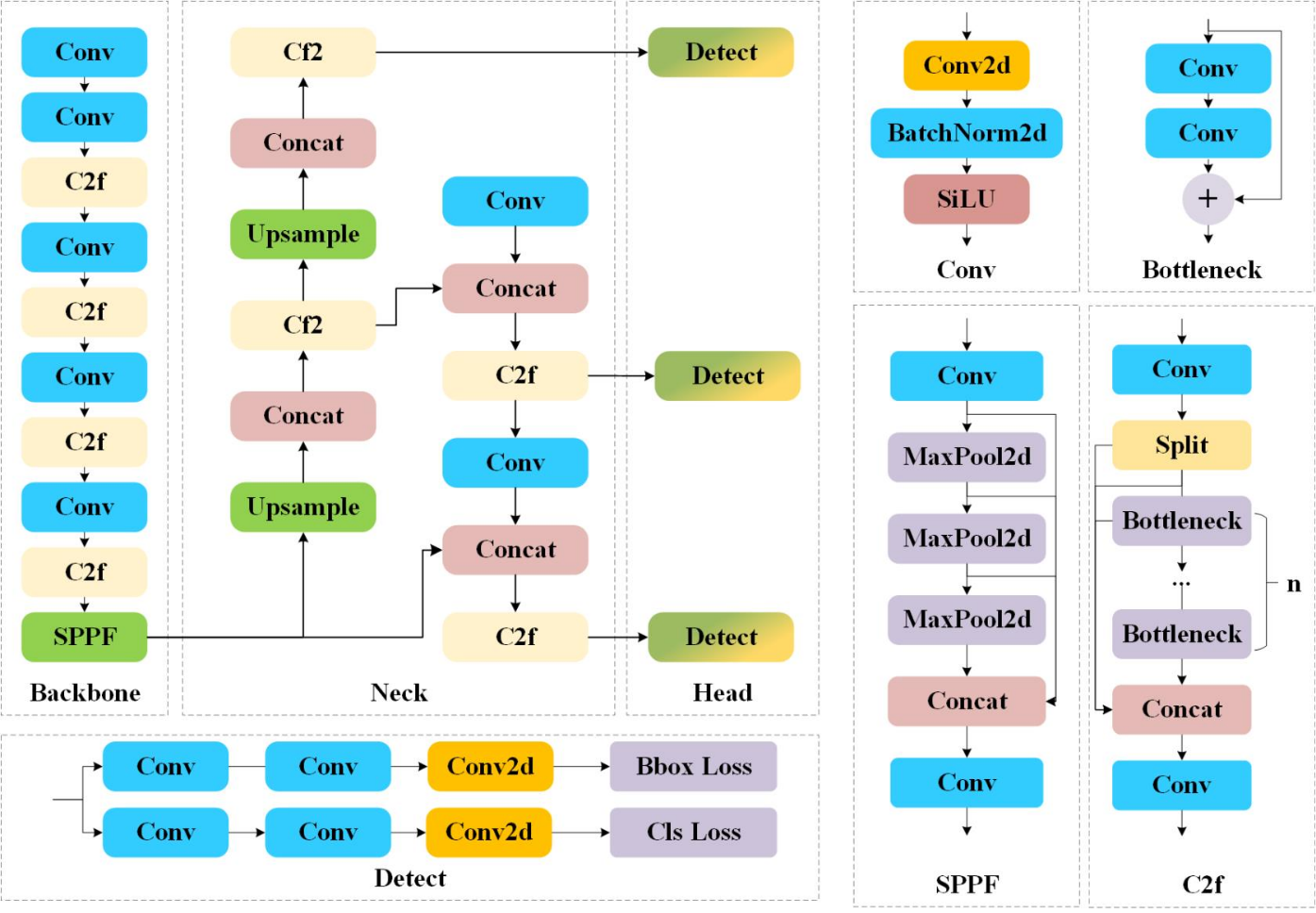
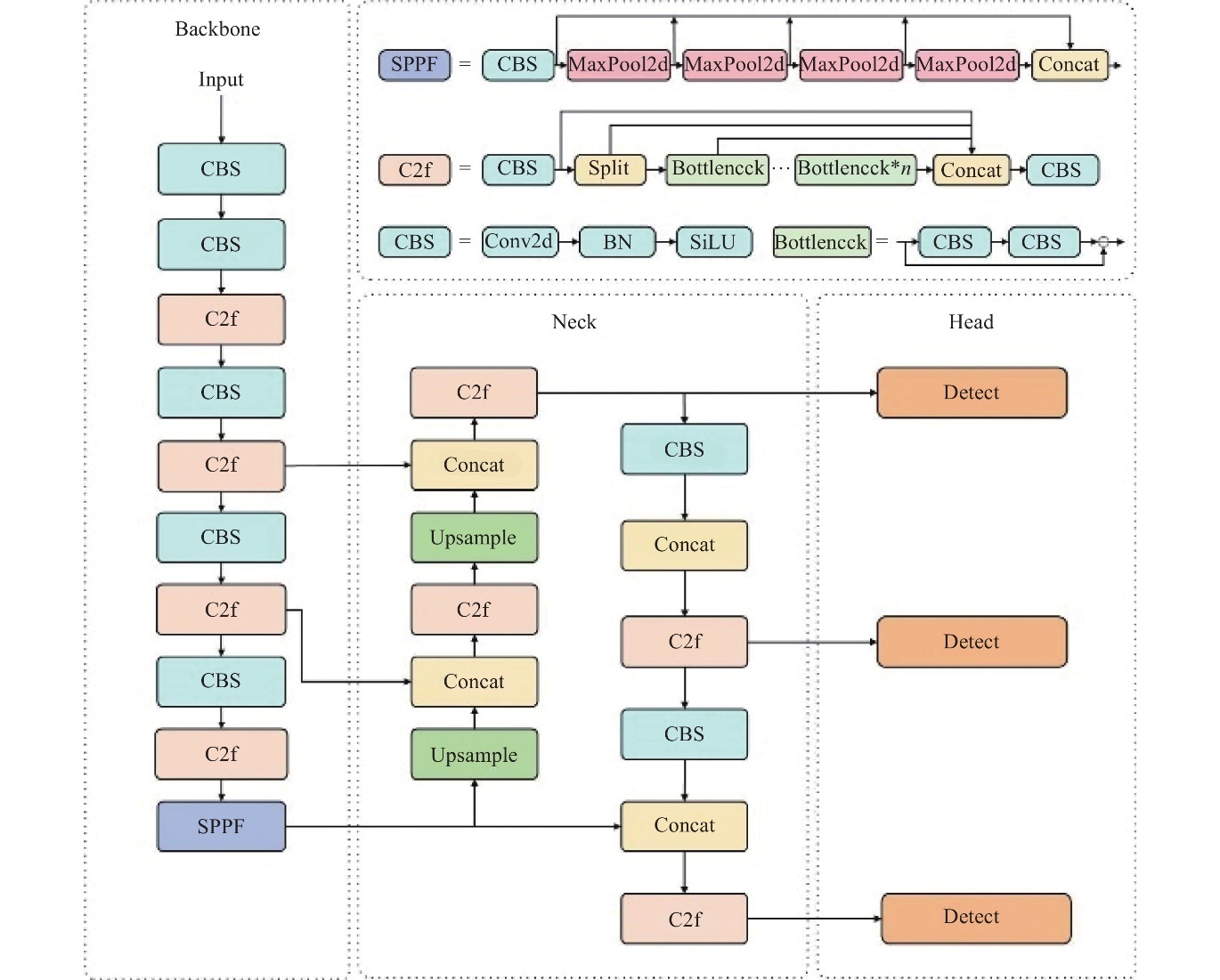


图 2‑1 YOLOv8n网络结构



## 系统开发Web技术

### 后端开发技术

#### 1）SpringBoot框架

Spring框架自2003年推出，是一个轻量级的Java开发框架。它基于Interface21框架重新设计，并通过控制反转（IOC, Inversion of Control）与依赖注入（DI, Dependency Injection）实现对对象生命周期的容器化管理，从而有效解耦代码并实现模块化，提高了资源的可配置性和易管理性。Spring框架采用面向切面编程（AOP, Aspect-Oriented Programming）进行声明式事务管理，并整合了多种持久化技术进行数据访问管理。此外，Spring框架还提供了丰富的Web框架，广泛应用于企业级Java应用的开发，涵盖Web应用、微服务架构、批处理任务等领域。

随着技术的不断发展，Spring框架在Java EE（Java Enterprise Edition）开发中的使用逐渐变得笨重，尤其是在配置和部署过程中需要大量的XML文件，这一复杂配置过程被称为“配置地狱”，大大降低了开发效率。自2013年起，Spring Boot项目开始研发，它是Spring家族中的新框架，旨在简化Spring应用程序的创建和开发。Spring Boot遵循“约定优先配置”原则，避免了XML配置，通过默认配置结合Maven等包管理工具，自动配置数据库连接、Web服务器、日志记录和安全性等。它还提供了大量的“起步依赖”（Starter Dependencies），使开发者能够通过引入相应的依赖包，快速集成所需的第三方框架，生成Spring应用程序。此外，Spring Boot集成了Tomcat等Servlet容器，简化了项目部署，只需将项目打包成jar或war文件，即可在目标环境中运行。在开发过程中，开发者可以通过JavaConfig类和注解等方式进行配置，使代码更加简洁，提升维护性。

#### 2）MyBatis-plus

MyBatis-Plus是基于MyBatis的增强框架，旨在简化开发过程并提高效率。MyBatis作为一款优秀的ORM（对象关系映射）框架，在此基础上，MyBatis-Plus提供了更多功能和特性，使得开发者在进行数据库操作时更加轻松和高效。MyBatis-Plus减少了MyBatis原有的复杂配置，开发者只需少量配置即可使用大部分功能。它还提供了代码生成器和强大的通用接口，能够帮助开发者快速完成基础功能的开发。

在Web开发中，MyBatis-Plus通常与Spring Boot结合使用。基于Spring Boot的简洁配置方式与自动化配置功能，MyBatis-Plus在数据持久层提供了强大且易用的支持，使得两者结合能够显著提升开发效率。

#### 3）Mysql

MySQL是一款广泛使用的开源关系数据库管理系统（RDBMS），其主要功能是存储和管理数据，采用结构化查询语言（SQL）进行操作。MySQL具备高性能、灵活性和可靠性，能够处理大量并发连接和复杂查询，适用于多种应用场景。其高可用性与负载均衡能力使得系统在处理大量数据时性能出色，尤其在样本查询或插入时表现优异。MySQL还能够与多种编程语言和框架无缝集成，方便开发者快速构建和部署应用。

#### 4）Flask框架

Flask 是一个用 Python 编写的轻量级 Web 应用框架，首次发布于2010年。Flask提供了最小化的核心功能，允许开发者根据需求自由选择其他功能组件或扩展包。作为一个微框架，Flask 不强制要求使用特定的数据库、表单验证或认证机制，开发者可以根据具体项目需求灵活选择，极大地提高了开发效率和灵活性。

Flask的主要特点是易于上手、灵活、扩展性强。它内置了基本的路由、模板引擎（Jinja2）以及 WSGI 兼容的开发服务器，可以快速搭建 Web 应用原型。在开发中，Flask 支持多种扩展（如 SQLAlchemy、Flask-Login 等），使得它在处理更复杂的功能时也能保持轻便和高效。通过 Flask，开发者可以快速构建小型Web应用或 RESTful API，尤其适合用于开发原型、微服务和小型项目。此外，Flask还拥有强大的社区支持和丰富的第三方扩展包，使其能够满足各种开发需求。

### 前端开发技术

#### 1）Vue框架

Vue.js是一种轻量级、渐进式的前端框架，由尤雨溪在2014年创建并于2015年首次发布。它的目标是通过简单易用的API和响应式数据绑定系统，使构建交互式的用户界面更加容易。Vue所关注的核心是 MVVM(Model-View-ViewModel)模式中的视图层，Vue以数据驱动和组件化思想为核心进行构建，在开发时可以将一个大型的应用分解为多个独立小组件进行开发，由底向上地构建应用；每个组件都拥有自己的模板、逻辑与样式，从而实现高度复用的代码，降低代码的耦合性，简化开发过程。且Vue提供了双向数据绑定与虚拟DOM的功能，提高了性能。在vue3中，推出了组合式API，将组件逻辑分解为更小的函数，通过函数的组合实现组件的逻辑复用。

Vue Router是Vue进行路由管理的官方工具，Vue Router使用声明式的方式配置路由规则，声明路由路径与组件关联。路由管理使得应用可以在不同的页面之间传递数据，实现页面间的信息交互和跳转。Pinia是一个Vue的轻量级状态管理库，它使用响应式数据来管理状态，自动触发相关组件的更新保持界面数据同步，Pinia支持异步状态管理，可以处理异步操作、网络请求等复杂场景，保证状态管理的稳定性和可靠性。

#### 2）Element组件

Element UI 是一款基于 Vue.js 的开源 UI 组件库，由饿了么团队开发，旨在为开发者提供一套高质量、易用且美观的 UI 组件，专注于开发桌面端应用的用户界面。Element UI 提供了丰富的组件，涵盖了表单、按钮、表格、弹窗、布局等常见的 UI 需求，可以帮助开发者快速构建出符合现代设计风格的应用。组件库遵循了响应式设计原则，确保在不同屏幕尺寸下能够良好适配，提升了用户体验。

Element UI 的设计理念注重简洁、实用与美观的平衡，组件样式风格清新且具有较高的可定制性。通过简化配置和灵活的组件使用方式，Element UI 大大降低了开发难度，使得前端开发者能够专注于业务逻辑的实现，而无需过多关注界面的细节。此外，Element UI 还提供了完善的文档和示例，帮助开发者快速上手，并根据项目需求自定义组件样式，达到更好的视觉效果与功能性。

#### 3）Axios技术

Axios 是一个于在浏览器和 Node.js 中发送 HTTP 请求的HTTP 客户端。它是一个轻量级的库，专为处理 HTTP 请求和响应数据而设计，能够与Vue框架进行无缝集成。Axios技术提供简洁的 API、Promise 支持、请求拦截和自动 JSON 处理，简化了 HTTP 请求的发起和管理。

## 本章小结

本章首先介绍了基于深度学习的目标检测算法，其中目标检测算法分为一阶段和双阶段两类，其中YOLOv8作为一阶段算法，通过优化网络结构、引入新技术如解耦头、Anchor-free模型等，显著提高了检测精度和速度；随后介绍了系统开发所使用的Web技术，后端使用了SpringBoot+Flask框架，简化了Java开发流程，同时使用MySQL数据库提供数据支持，前端技术采用Vue+Element UI，提升了前端开发的效率和界面交互性。

# 基于改进YOLOv8n的番茄叶片病虫害检测模型

本文的研究目的是分析番茄叶片的发病情况，识别并返回概率最高的番茄叶片病虫害类别。本章设计了基于GhostNet网络以及基于模型剪枝的两种轻量化改进方法，并对YOLOv8n模型进行优化，以精简模型结构并提升其运行效率，从而使其满足实时检测的需求；最终将轻量化改进后的模型命名为Light-YOLOv8n。实验证明，Light-YOLOv8n模型能够在小幅度降低识别精度的情况下，显著提升其运行效率，并且对比其他目标检测模型，整体上性能更具优势，在运行效率和识别准确率之间取得了更好的平衡。

## 应用场景分析与问题描述

本系统面向农业领域中实际部署使用的病虫害检测场景，因此需要选择稳定、成熟与可维护的模型。YOLOv8作为目前目标检测领域中应用广泛的主流模型之一，具备高度工程化的实现框架、完善的文档支持以及活跃的开源社区，适合部署于需要长期运行的农业图像识别系统中。基于此，本文选取了YOLOv8系列中的轻量版本YOLOv8n（Nano）作为基础检测模型。

在构建数据集并完成初步训练后，YOLOv8n在本课题数据上取得了较高的识别精度，具备较强的特征提取与病虫害区域定位能力。经过多轮实验评估，YOLOv8n 在本任务中已基本满足识别精度的需求，但在实际部署中仍存在以下问题：

（1）在边缘设备、嵌入式系统或移动终端中部署模型时，对模型体积、计算资源消耗和推理速度有更高要求。尽管 YOLOv8n 已属轻量级模型，但其参数量和计算量（FLOPs）对于部分低功耗设备仍不够友好，存在进一步压缩空间。

（2）在农业巡检、实时识别等场景中，对模型的响应速度要求极高。为提高系统整体运行效率，减少识别延迟，有必要进一步对模型结构进行裁剪和精简，加快前端识别处理流程。

（3）YOLOv8n 虽为通用模型，在实际任务中仍存在部分模块对特定场景的适应性不强或特征冗余的情况。通过结构分析和剪枝策略，可识别并移除部分对本任务贡献较小的通道或层，达到压缩模型的目的。

## 基于 GhostNet 网络的轻量化改进

YOLOv8模型的 Backbone 网络中包含多个标准卷积模块，用于提取图像的深层特征，以提升模型对目标的感知能力。标准卷积容易产生大量冗余特征图，这些冗余特征虽然在一定程度上增强模型的表达能力，但同时也需要大量的参数与计算开销才能获得，进而导致模型参数量大、计算时间长等问题。在这些输出的特征图中，对于任意一组冗余的特征图，只需要通过一些简单变换操作，其中的一张特征图就可以由另一张特征图计算而得到。

为此，本文引入GhostNet网络的设计思想，对YOLOv8的Backbone 结构进行了轻量化改进。GhostNet 网络于 2020 年由 Han 等人提出，研究人员认为卷积神经网络强大的特征提取能力正是由于这些冗余特征，因此GhostNet网络的设计理念是采用廉价高效的操作方式来获取更多的特征。

GhostNet网络是由其核心组件 Ghost 模块（Ghost Module）堆叠而成。GhostNet的网络结构如表3.1所示。该表中，第一列为各模块的序号，第二列表示每个模块输入特征图的尺寸，第三列为模块名称，其中 “G-bneck” 表示 Ghost Bottleneck 瓶颈模块；第四列对应 Ghost Bottleneck 模块的膨胀尺寸（Expansion Size）；第五列展示各模块输出的通道数；第六列指示是否使用了SE模块；第七列则为各模块操作所采用的步长数。

表 3‑1 GhostNet 的网络结构

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | | 输入尺寸 | 模块 | 膨胀尺寸 | 输出通道数 | 是否使用SE模块 | 步长 |
| 1 | 224² × 3 | | Conv2d 3×3 | - | 16 | - | 2 |
| 2 | 112² × 16 | | G-bneck | 16 | 16 | - | 1 |
| 3 | 112² × 16 | | G-bneck | 48 | 24 | - | 2 |
| 4 | 56² × 24 | | G-bneck | 72 | 24 | - | 1 |
| 5 | 56² × 24 | | G-bneck | 72 | 40 | 1 | 2 |
| 6 | 28² × 40 | | G-bneck | 120 | 40 | 1 | 1 |
| 7 | 28² × 40 | | G-bneck | 240 | 80 | - | 2 |
| 8 | 14² × 80 | | G-bneck | 200 | 80 | - | 1 |
| 9 | 14² × 80 | | G-bneck | 184 | 80 | - | 1 |
| 10 | 14² × 80 | | G-bneck | 184 | 80 | - | 1 |
| 11 | 14² × 80 | | G-bneck | 480 | 112 | 1 | 1 |
| 12 | 14² × 112 | | G-bneck | 672 | 112 | 1 | 2 |
| 13 | 14² × 112 | | G-bneck | 672 | 160 | 1 | 2 |
| 14 | 7² × 160 | | G-bneck | 960 | 160 | 1 | 1 |
| 15 | 7² × 160 | | G-bneck | 960 | 160 | - | 1 |
| 16 | 7² × 160 | | G-bneck | 960 | 160 | 1 | 1 |
| 17 | 7² × 160 | | G-bneck | 960 | 160 | - | 1 |
| 18 | 7² × 160 | | Conv2d 1×1 | - | 960 | - | 1 |
| 19 | 7² × 960 | | AvgPool 7×7 | - | - | - | - |
| 20 | 1² × 960 | | Conv2d 1×1 | - | 1280 | - | 1 |
| 21 | 1² × 1280 | | FC | - | - | - | - |

Ghost Module 模块的最大优势在于其轻量高效的特性，其核心思想是以更少的参数生成更多的特征信息。该模块可视为一种即插即用的组件，常用于替代卷积神经网络中的传统卷积操作，以实现网络结构的优化与升级。

对输入特征图分别使用常规卷积操作与 Ghost Module 模块生成输出特征图的过程如图 3.1 和图 3.2 所示。在图 3.2 中，Ghost Module 模块首先通过常规卷积完成初步的特征提取，生成数量较少的原始特征图；随后，利用简单的线性变换对这些原始特征图进行分组卷积，以生成更多的冗余特征信息，从而有效增强模型的特征表达能力。

为了确保最终输出特征图与常规卷积操作所得结果的一致性，Ghost Module 模块通过恒等映射方式引入一条 Identity 残差边，将原始特征图与分组卷积所生成的冗余特征图进行拼接，并以此作为输出。

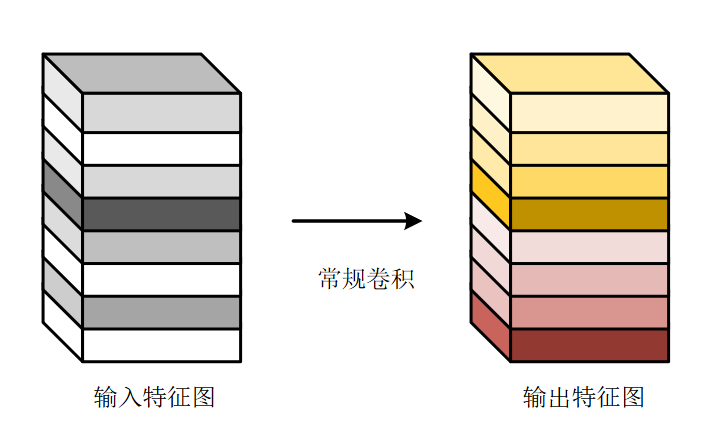


图 3‑1 常规卷积生成特征图

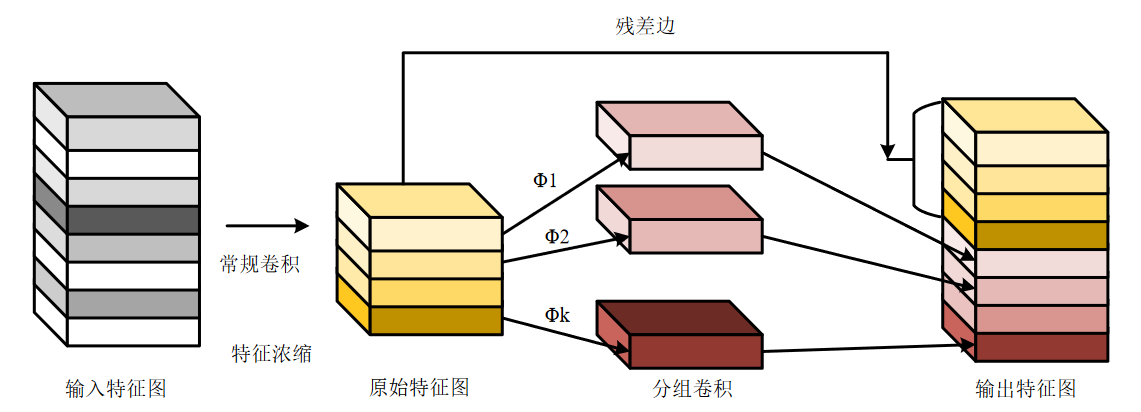


图 3‑2 Ghost Module生成特征图

假设输入特征图用表示，其中c为输入通道数、h为高度、w为宽度。卷积核表示为，输出特征图是，则使用常规卷积操作生成有 n个通道的输出特征图的过程，其计算公式如下：

（3-1）

其中，是输出通道数量；\*是卷积操作；为其尺寸大小；是偏置。整个常规卷积过程所需计算量和网络参数量的计算公式如下：

（3-2）

（3-3）

其中，是输出通道数量，也是卷积核的个数；是输入通道数量。

使用Ghost Module模块进行特征浓缩的过程，其计算公式如下：

（3-4）

其中，在特征浓缩阶段生成的初始特征图具有*m*个通道，其尺寸小于原始特征图。卷积核 的尺寸为*k*×*k*，其中*m*≤*n*。该过程未包含偏置项，其余超参数与标准卷积相同，以确保输出特征图的空间维度不变。

为实现输出特征图通道数达到*n*，Ghost Module对初始特征图实施分组卷积操作，产生*s*组冗余特征，其计算表达式如下：

（3-5）

其中，代表原始特征图*Y*的第*i*个分段，用于产生 *s* 个冗余特征。其中，表示对执行的第*j*次线性变换（不包括最后一次），生成第*j*个冗余特征，而最后的则用于恒等映射。由于恒等映射的计算开销远低于标准卷积，这一设计有效降低了模型的计算负担。最终，Ghost Module 能够输出通道数为*n*=*m*×*s*的特征图，其中原始特征图的通道数满足*m*=*n*/*s*。

在分组卷积阶段，Ghost Module共执行*m*×(*s*−1)=(*n*/*s*)×(*s*−1) 次线性操作，其平均卷积核尺寸为*d*×*d*（与*k*×*k* 相近），并通过1个残差连接将生成的冗余特征与原始特征图融合。整个模块包含特征浓缩、分组卷积和特征拼接三个步骤，其计算量和参数量的计算公式如下：

（3-6）

（3-7）

这里，由于*s*≪*c*，通过上述公式可以看出，采用 Ghost Module 模块后，网络的计算量和参数量相比传统卷积操作均显著降低，降幅约为*s*倍。这一优势表明，Ghost Module 能够以更少的参数生成更丰富的特征，从而有效提升模型的运行效率。具体计算公式如下：

（3-8）

（3-9）

在表 4.1 所示的网络架构中，G-bneck（Ghost Bottleneck）模块借鉴了残差网络中的瓶颈设计思想。该模块主要由两个 Ghost Module 组成：第一个模块负责增加通道维度，以增强特征表达能力；第二个模块则减少通道数，使其能够与残差连接部分相匹配。根据步长的不同，Ghost Bottleneck 可分为两种类型：当步长为 1 时，模块直接进行特征变换；当步长为 2 时，模块会通过下采样层构建快捷路径，并在两个 Ghost Module 之间加入步长为 2 的深度可分离卷积（Depthwise Convolution）来实现特征的空间降维。图 4.3 展示了 Ghost Bottleneck 的详细结构设计。这种模块化设计既保留了特征提取能力，又显著降低了计算复杂度。

原始 Backbone 中大量采用的 C2f 模块尽管已经具备良好的轻量化能力，其设计融合了 C3 与 ELAN 的思想，使得网络能够在保持一定深度的同时增强梯度流动性，获得较好的收敛性与特征表达能力。但相比 C3 模块，C2f 结构中引入了更密集的 Bottleneck 连接，虽然梯度信息更丰富，但计算复杂度也略有增加。为在模型精度与效率之间取得更好的平衡，本文选择将 C2f 替换为 C3Ghost 模块。

图示

AI 生成的内容可能不正确。



图 3‑3 Ghost Bottleneck 瓶颈模块的结构图

如图3-4、图3-5所示，C3Ghost 模块在原有C3模块上，用 GhostConv 替代其中的标准卷积单元，使其在特征提取能力与模型参数规模之间取得更优折中。该替换方式不仅保留了残差连接与跨阶段特征融合的结构优势，也借助 GhostConv 的高效机制，进一步压缩了模型体积，提升了整体推理速度。

图示

AI 生成的内容可能不正确。图示

AI 生成的内容可能不正确。



图 3‑4 C2f模块结构 图 3‑5 C3Ghost模块结构

综上，本文最终对原YOLOv8n模型的 Backbone 网络结构进行了如下改进：将其中的标准卷积模块统一替换为 GhostConv 模块，以减少参数量；同时将所有 C2f 模块替换为轻量化的 C3Ghost 模块。改进后的 Backbone 网络对输入图像提取特征的具体流程如下：输入特征图大小为 640×640×3，首先通过一个 CBS 模块（包含 Conv、BatchNorm 和 SiLU 激活函数），输出特征图大小为 320×320×32；接着，依次经过两层使用 GhostConv 的 CBS 模块与一个 C3Ghost 模块，输出特征图大小分别为 160×160×64、80×80×128 和 80×80×128；随后，通过一个使用 GhostConv 的 CBS 模块和一个 C3Ghost 模块，得到的特征图大小分别为 40×40×256 和 40×40×256；之后，再通过一个 GhostConv 的 CBS 模块与一个 C3Ghost 模块，输出特征图大小为 20×20×512 和 20×20×512；最后，再经过一个使用 GhostConv 的 CBS 模块，输出特征图大小为 20×20×1024，并将该结果输入至原YOLOv8n模型中的 SPPCSPC 模块进行多尺度感受野的融合处理。



## 基于模型剪枝的轻量化改进

为了进一步优化网络模型的轻量化性能，本章提出了一种基于L1范数剪枝的改进方法。该方法以3.2节中轻量化改进后的YOLOv8n模型为基础，通过评估卷积核在辣椒病虫害特征提取中的贡献度，对冗余卷积核进行裁剪。具体而言，首先依据L1范数准则对每个卷积层进行分析，识别并移除那些对特征学习贡献较小的卷积核，从而实现对模型通道数或网络层数的动态调整。剪枝完成后，保留的重要参数将被重新初始化，并通过多次迭代训练来恢复因剪枝导致的精度损失。

该方法的核心优势在于能够在最大限度保留模型精度的同时，显著降低模型的复杂度。通过剪枝操作，不仅减少了计算资源的消耗，还提升了模型的推理效率。图3-6展示了该方法对卷积核进行剪枝的具体流程。

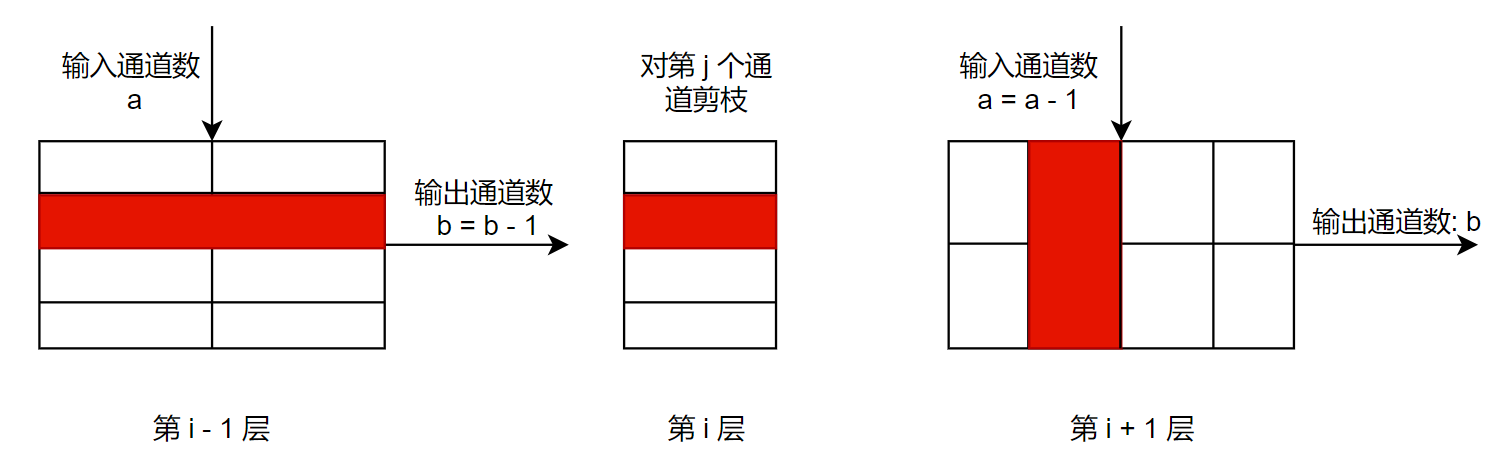




图 3‑6卷积核剪枝的过程图

其中，设  为卷积层的编号，图4.4中的红色区域代表需要被剪枝的卷积核结构。当对第  层实施剪枝时，其前后相邻卷积层的对应区域也会同步进行裁剪，以保证网络结构的连贯性。

具体而言，第层的输入特征图可表示为，其中为输入通道数，和分别为特征图的高度和宽度。经过该层卷积运算后，输出特征图变为，并作为第层的输入。该层的卷积核由个二维滤波器组成，整体可表示为三维张量 。由此，第  层到第  层的总计算量  可表示为：

（3-10）

基于L1范数准则，本章方法能够识别出对病虫害特征提取贡献较低的卷积核。假设第 层中共有个冗余卷积核，剪枝时需同时移除第层的卷积核，及其在第层对应的特征图通道。这一操作可减少的计算量由以下公式计算：

（3-11）

根据公式(4-10)和(4-11)，当剪除第层第个卷积核时，其减少的计算量占第层到第层总计算量的比例为，具体推导如下：

（3-12）

若剪除全部个冗余卷积核，则减少的计算量记为，计算公式为：

（3-13）

根据公式(3-10)和(3-13)的分析结果，剪除个卷积核可减少的计算量占总计算量的比例为。这一结果表明，本文提出的轻量化方法能有效降低模型计算复杂度，提高运行效率。

在方法设计上，本文采用L1范数准则来评估卷积核的重要性。具体而言，通过计算第个卷积层中第个卷积核的权重绝对值之和，可以量化该卷积核对特征提取的贡献程度, 它用L1范数准则可表示为。考虑到卷积层的卷积核数量与输入通道数相等，因此权重绝对值之和也可用该核的平均大小重量来表示，也就是使用绝对内核权重作为归一化评估指标，其计算公式如下：

（3-14）

基于模型剪枝的轻量化改进方法采用迭代处理的方式，逐个分析卷积层的权重矩阵作为输入，最终输出优化后的轻量化模型。该方法的具体实现步骤如下：

首先，针对每个卷积层计算各卷积核的绝对内核权重，该指标用于评估卷积核在辣椒病虫害识别任务中的重要性。通过将各卷积核的值进行排序，可以确定对模型性能影响最小的卷积核。随后，对这些重要性较低的卷积核实施剪枝操作，同时同步剪除其在后续卷积层中对应的特征图通道部分。

完成剪枝操作后，需要对模型中保留的重要参数重新初始化，并通过多次迭代训练来恢复因剪枝造成的精度损失。实验结果表明，该方法不仅能够维持较高的识别精度，还能显著提升模型的运行效率。这为后续将模型集成到实际系统中提供了便利，同时也增强了模型在各类轻量级设备上的部署能力。而且还可以保证在第五章将模型置于开发的系统内部时，能够满足实时运行效率，并且在未来还能更好地部署在其他轻量级设备上。

在具体应用方面，本研究将GhostNet网络结构与模型剪枝方法相结合，对原有的ABN-YOLOv7模型进行了轻量化改进，最终构建了ABN-Light-YOLOv7模型。如图4.5所示，该模型的检测流程中，蓝色框标注部分即为采用轻量化改进的关键环节。

## 实验设计和结果分析

### 实验数据集及实验环境

### 评价指标

（3-15）

公式（3-1）说明，…………（公式在正文中的引用）

图题注：

图 3‑3 XXXXXX

### 对比试验分析

公式按章重新编号：

 （3-1）

公式（3-1）说明，…………（公式在正文中的引用）

图题注：

图 3‑4 XXXXXX

### 消融实验分析

公式按章重新编号：

 （3-1）

公式（3-1）说明，…………（公式在正文中的引用）

图题注：

图 3‑5 XXXXXX

## 本章小结

本章围绕YOLOv8n模型的轻量化改进展开，提出并验证了两种有效的优化方法。首先，通过引入GhostConv模块替代Backbone部分原有的标准卷积，有效降低了模型参数数量和计算复杂度，同时保持了较高的识别精度。随后，设计了基于模型剪枝的轻量化策略，对卷积层中冗余或贡献较低的卷积核进行裁剪，并结合重新训练的方式恢复模型性能。在此基础上，综合应用两种方法构建了改进模型 Light-YOLOv8。

为验证所提方法的有效性，本章设计并开展了多组实验，包括与主流大参数模型、YOLO系列模型及YOLOv8n的性能对比，结果显示Light-YOLOv8模型在保持识别准确率的同时显著提升了推理速度与运行效率。此外，消融实验进一步证实两项轻量化策略均对模型性能优化有积极作用。综上，Light-YOLOv8模型在准确率与轻量化之间实现了良好平衡，为后续部署与应用提供了更优方案。

# 番茄叶片病虫害识别系统的需求与设计

番茄叶片病虫害识别系统的实现主要基于第三章设计构建的Light-YOLOv8n模型，该模型能够实现高的识别准确率和运行效率，为实时监测提供了可行性。为了更好地展示模型的检测结果，以及为用户提供更直观的交互界面，使他们能够及时得到病虫害类别并对症下药，本章对系统进行需求分析与总体设计，再对各功能模块的具体详细设计进行阐述。

## 系统需求分析

本系统旨在积极响应国家智慧农业的发展战略，助力农业信息化水平的提升与农业生产方式的转型升级。当前，辣椒作为重要的经济作物，在种植过程中常面临多种病虫害的威胁。然而，多数农户仍主要依赖个人经验和肉眼判断进行病害识别，这种方式不仅效率较低，还存在较大的主观性和误判风险。为解决上述问题，系统引入深度学习算法实现图像自动识别，结合前后端集成的Web技术，开发出一套面向辣椒病虫害的智能识别与管理平台。

该系统面向一线农业生产者，尤其是广大辣椒种植户，通过图像识别功能帮助用户快速获取病虫害类别与相应的防治方案，辅助其科学决策和及时应对，从而减少损失、提高产量。同时，系统还具备百科知识查询、识别记录管理、用户账户维护等多种实用功能，力求为农业生产实践提供准确、高效、易用的技术支持。

根据以上需求分析，可以梳理出本系统的功能模块应当包括：图像识别模块、病虫害百科模块、识别记录管理模块、数据管理模块（管理员专属）。

### 系统功能性需求分析

系统设有两类用户：一类是通过系统进行识别和信息查询的普通用户，另一类是具备系统管理权限的超级管理员。

（1）普通用户

普通用户需注册后登录系统。在登录状态下，用户可以浏览番茄叶片病虫害相关知识，了解不同病虫害的症状与防治措施；在识别功能页面，用户可上传待识别图像、待识别视频或者打开摄像头实时画面，系统将返回相应的识别类别和处理建议。对于识别结果明确且经用户确认的记录，系统将自动进行保存，便于用户后续在历史记录中查看与管理。此外，用户可查看和修改自己的账户信息。

（2）超级管理员

超级管理员为系统预设账户，具备平台管理职能。其可以对所有用户提交的识别记录进行集中管理，特别是已确认准确的图像数据，以用于系统数据库的维护和后续模型训练。与此同时，管理员还可查看和管理用户信息。

根据两种用户类型的分析，番茄叶片病虫害识别系统的用例图设计如图 4‑1所示。

图题注：

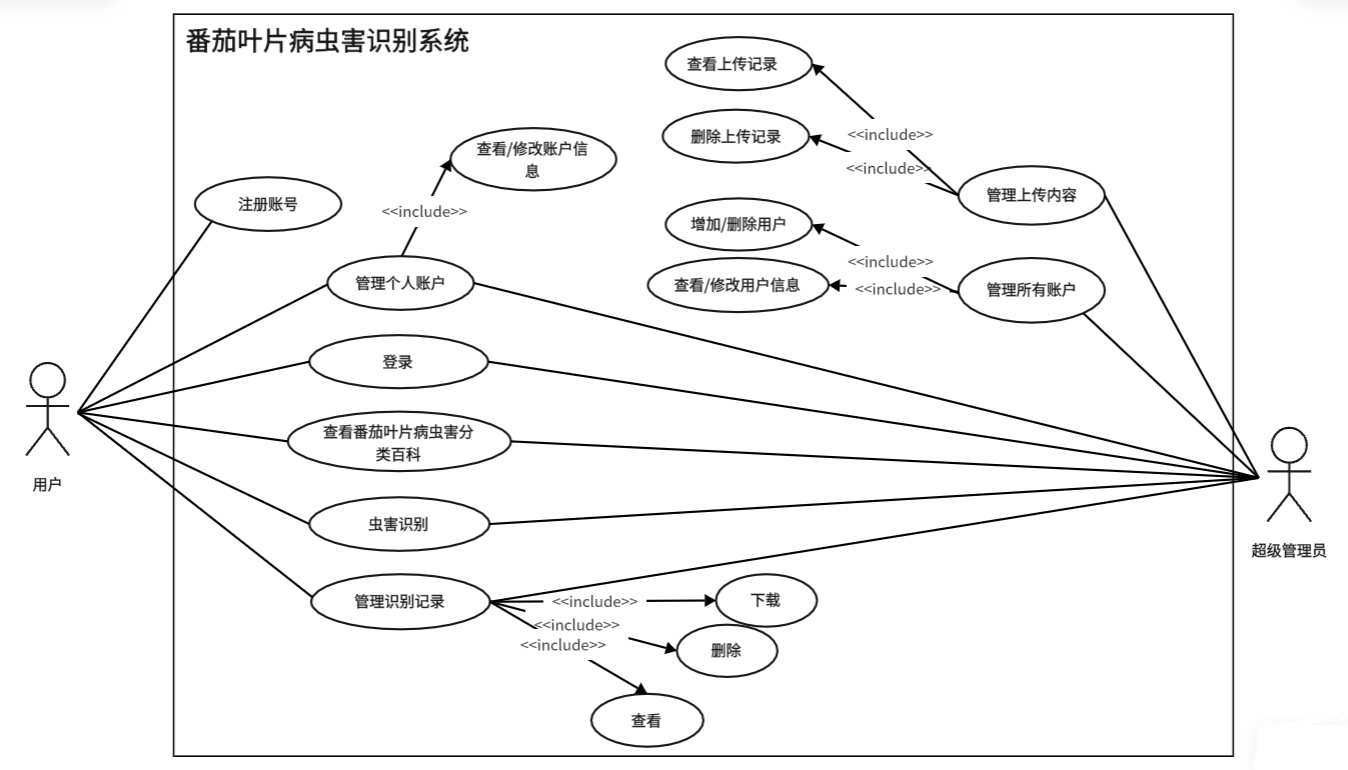


图 4‑1 系统用例图

### 系统非功能性需求分析

从系统的非功能性角度考虑，该系统需要满足以下几个需求：

（1）响应性能要求

考虑到用户多为农业从业者，日常操作习惯偏向直观高效，系统需具备良好的响应速度。在网络良好的条件下，系统对上传图片的识别响应时间应控制在2秒以内，保证用户能快速获取识别结果与防治建议，不影响现场处理进度。尤其在田间地头使用移动设备操作时，系统页面加载与功能切换的响应时间应控制在1秒以内。

（2）识别准确率要求

识别结果的准确性直接影响农户的防治决策。系统基于YOLOv8目标检测模型进行训练优化，常见病虫害（如疫病、蚜虫、病毒病等）的单类识别精度应高于90%。此外，对于模糊、光照不均等劣质图像，系统需具备基本的鲁棒性，降低误判率。

（3）并发处理能力

为满足农业合作社或农技站等集中使用场景的需求，系统应支持至少50个并发用户同时上传与识别图像，且服务器可根据访问压力自动扩展资源，保障系统在高峰期运行稳定，不出现崩溃、识别卡顿等现象。

（4）数据安全与隐私保护

农户上传的图像中可能包含种植区域、农场设备等敏感信息，系统需对上传图像与识别结果采用HTTPS加密传输并进行本地存储加密，确保用户数据在传输与存储过程中的安全性。同时，对账号权限进行精细化控制，农户仅可访问本人数据，管理员拥有数据管理权限，保障数据不被滥用。

（5）系统可维护性与容错性

考虑到使用群体以非技术人员为主，系统在发生错误时应提供简洁明了的提示信息，如“图像模糊无法识别，请重新上传”等。同时后端应配置日志记录机制，便于开发人员在出现故障时迅速定位问题并修复。建议设置每日自动备份功能，保障数据不因意外操作或系统错误而丢失。

（6）数据与模型可扩展性

随着系统运行时间的延长与用户的增多，图像识别样本将持续增长。系统应为图像数据与识别结果提供结构化存储方案，方便后期的数据检索与统计分析。

## 系统概要设计

### 系统整体体系结构

本系统基于 MVC 架构进行设计，体现了模块分层的思想，主要包括视图（View）、模型（Model）和控制器（Controller）三部分。视图层负责展示用户界面，模型层处理与数据相关的操作，控制器层用于接收浏览器请求并调用相应逻辑后返回处理结果。在此基础上，系统进一步划分为表示层、业务层和持久层，其整体架构如图 4‑2所示。

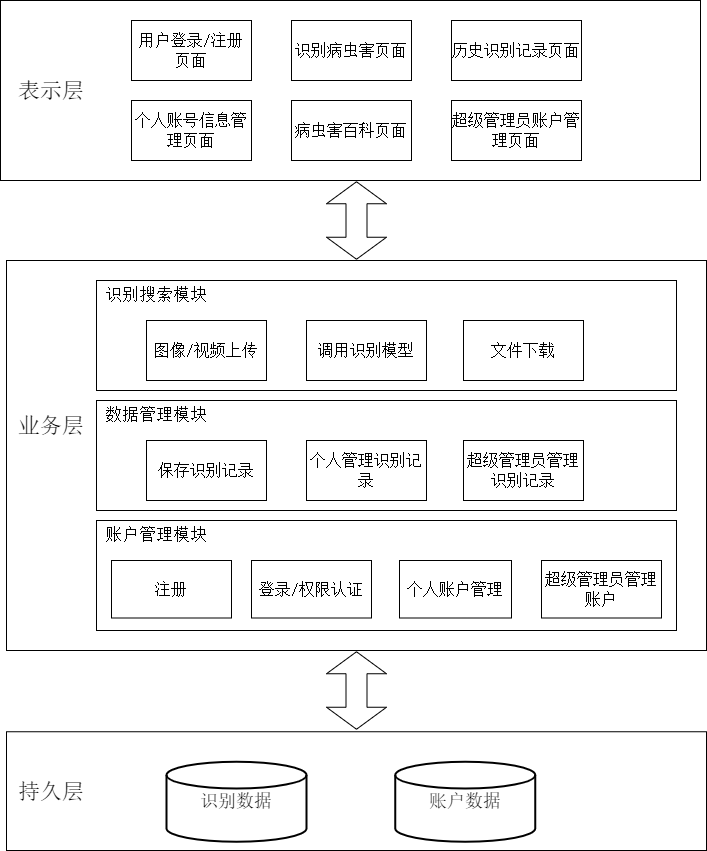


图 4‑2 系统整体架构图

表示层负责为系统的两类用户（普通用户与超级管理员）提供直观、易用的图形化交互界面。对于普通用户，系统提供了包括账户注册、登录、图像上传识别、识别结果查看与保存、历史记录管理等功能模块。用户可在“番茄叶片病虫害百科”中浏览常见病虫害类别，在识别模块中上传图片获取识别结果，结果可保存至个人历史记录中，便于随时查看、下载或删除。此外，用户还可在“账户管理”页面修改密码或更新账户信息。对于超级管理员，系统提供了账户管理和数据审核功能。管理员可以查看所有用户信息，进行账户状态的启用、停用或修改操作；同时，还可以统一管理所有用户上传的历史识别记录，包括对有效数据图像的分类、筛选与整理，为后续模型训练数据集扩充提供支持。

业务层作为系统逻辑处理的核心，负责接收来自前端的请求并执行相应操作，完成后将处理结果返回表示层进行展示。该层主要涵盖：①识别服务模块：调用部署好的辣椒病虫害识别模型，接收用户上传图像，进行图像预处理后送入模型进行推理，并返回识别结果与置信度。②账户与权限管理模块：处理用户的注册、登录验证、权限识别、信息修改等行为，确保系统操作的合法性与安全性。③数据管理模块：实现用户历史识别记录的存储、查询、删除及管理员审核等功能，同时为模型提供新数据的积累渠道。

持久层负责将系统中关键的数据内容进行结构化存储和管理，确保数据安全、稳定、可持续地支持业务运行。本系统使用 MySQL 数据库进行数据持久化，主要存储内容包括：用户信息、识别记录、病虫害百科、系统操作日志（可选）。

### 系统功能模块设计

根据以上对基于音频内容检索的网络音乐侵权判别系统的需求分析，可以把该系统划分为3大功能模块：识别搜索模块、数据管理模块、账户管理模块，如图 4‑3所示。

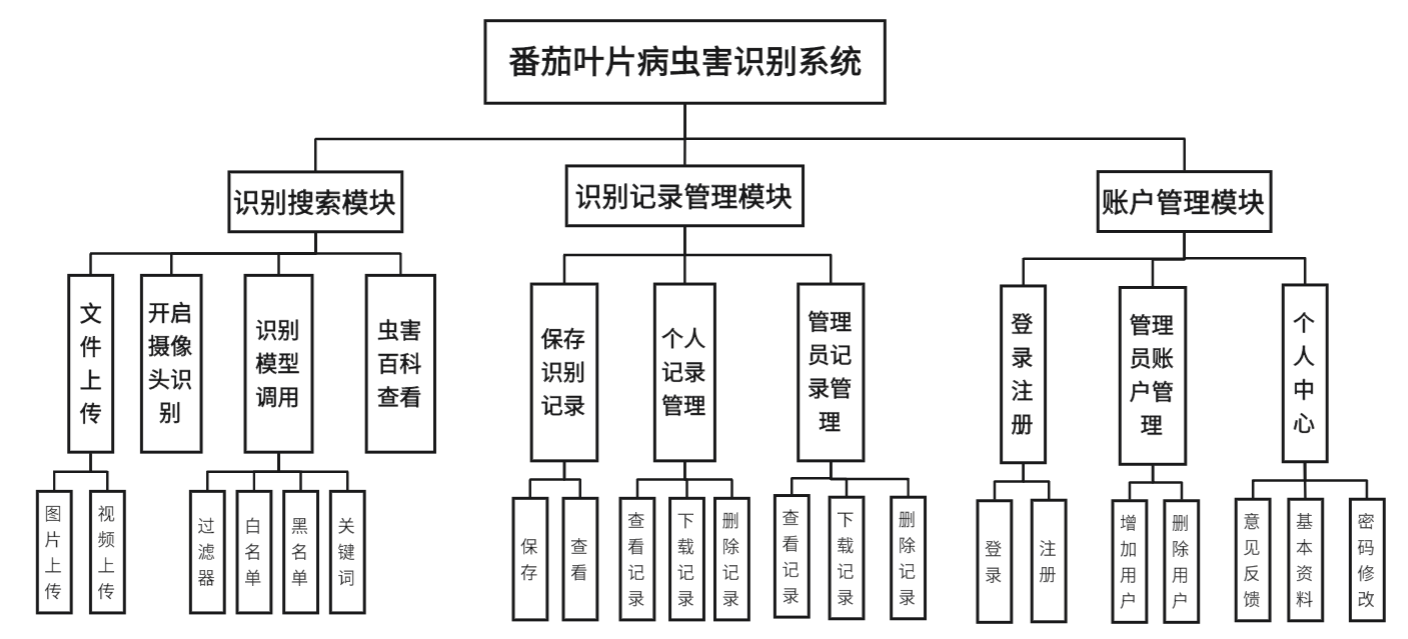


图 4‑3 系统功能模块图

识别搜索模块负责病虫害识别与交互功能，用户可通过文件上传或视频上传提交本地图片或视频片段进行分析，也可实时开启摄像头拍摄叶片图像进行识别，系统调用第3章设计的模型完成病虫害检测后提供详细的虫害百科信息供用户查阅，所有识别结果可保存至个人记录供日后查看。

用户管理模块处理系统权限与账户相关操作，普通用户通过登录注册功能访问系统，在个人中心可修改基本资料、更新密码或提交意见反馈，管理员则拥有更高权限，可进行账户管理如新增或删除用户，确保系统安全有序运行。

记录管理模块用于存储和管理历史识别数据，用户可查看、下载或删除个人识别记录，并按时间或病虫害类型进行检索，管理员能查看所有用户记录并进行批量操作，同时将有效数据归档至训练数据集以持续优化模型性能，实现系统识别能力的迭代提升。

### 识别搜索模块设计

本模块主要的核心逻辑在FileController和PredictController，FileConroller实现了upload（）接口。Flask端用于部署和调用模型。

用户上传图片获得识别结果过程的时序图如图 4‑4所示。

图示, 日程表

AI 生成的内容可能不正确。

图 4‑4 图片识别模块时序图

如图，用户(前端)发起图片上传请求, 调用FileController的upload()方法,把图片保存在Java服务端, 返回图片的url给前端; 前端点击"预测", 将图片url发送给PredictionController的predict()方法, 然后PredictionController把图片url发给flask端的predictImg()方法, flask端调用模型预测, 并把预测好的结果图片保存在flask端, 然后把结果图片上传到Java端, 获取结果图的url, 返回给PredictionController, 最后PredictionController把结果url存到MySQL的imgrecords表。对视频的识别、摄像头的识别与此类似。

### 识别记录管理模块设计

该模块主要负责管理识别记录，包括图片记录、视频记录和摄像头识别记录的查看、修改、下载。涉及对camerarecords表、imgrecords表和videorecords表的数据管理。

查看图片检索结果模块的静态类图如图 4‑5所示。

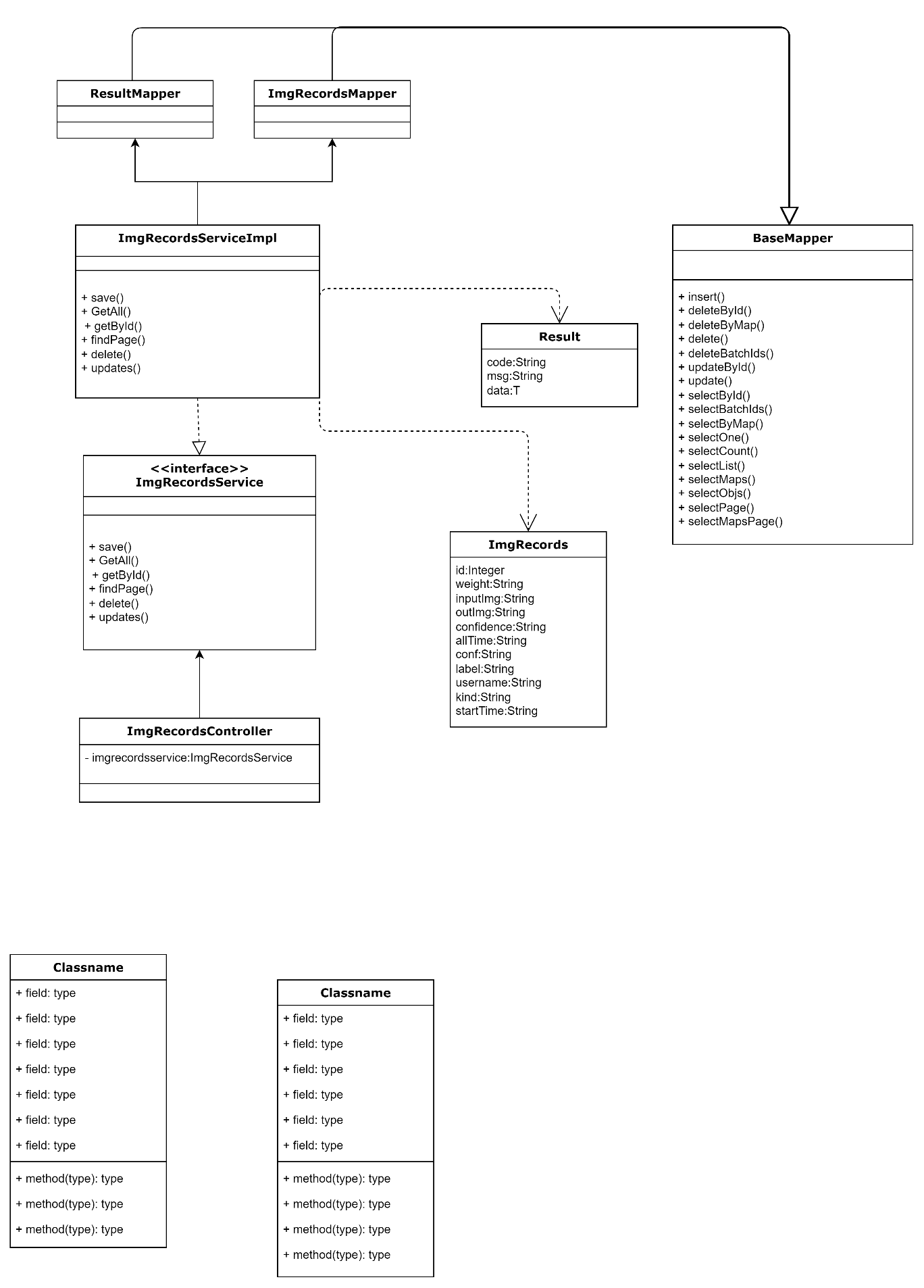


图 4‑5 查看检索结果模块类图

本模块主要的核心逻辑在ImgRecordsController类。ImgRecordsServiceImpl类实现了ImgRecordsService接口，主要提供了对识别记录的分角色查看、删除和更新功能。在查询与删改的过程中使用了继承自MyBatis-plus BaseMapper类的ImgRecordsMapper与ResultMapper，用于在持久化层查询与更新数据。对视频记录、摄像头记录的查询与此类似。

用户查看检索结果的时序图如图 4‑6所示。

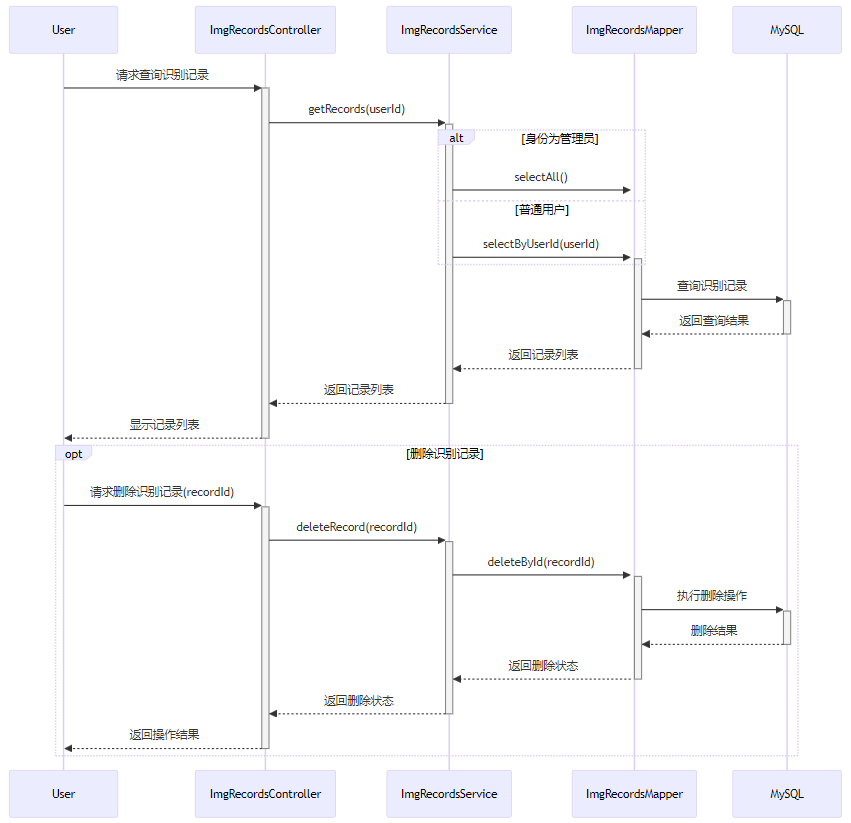


图 4‑6 查看检索结果模块时序图

如图所示，在用户进行识别记录查询时，前端首先请求ImgRecordsController的对应方法，控制器层调用ImgRecordsService的getRecords()方法处理该请求。在ImgRecordsService中，根据当前用户的身份判断其权限：若为管理员，则通过ImgRecordsMapper.selectAll()查询全部识别记录；若为普通用户，则调用selectByUserId(userId)仅查询其本人提交的记录。随后，ImgRecordsMapper从MySQL中获取对应记录，并依次返回给service层、controller层，最终以JSON格式返回至前端页面展示。

当用户发起删除识别记录的请求时，前端调用ImgRecordsController的deleteRecord()方法，控制器将该请求转发至ImgRecordsService的delete()方法。service层进一步调用ImgRecordsMapper.deleteById()执行删除操作，由mapper在MySQL中删除指定记录并将执行结果依次返回。最终，前端将该操作结果反馈给用户。

### 账户管理模块设计

用户使用系统前需要完成账号注册。该功能模块主要包含三个部分：账号注册、用户登录和账号管理。在注册环节，系统会进行以下验证：首先检查用户名是否已被占用（用户名作为登录凭证必须在系统中保持唯一性）；其次对密码进行加密处理（需确保两次输入的密码一致）；最后验证必填字段是否完整（如姓名等关键信息不能为空）。

用户登录时，系统会执行身份验证和权限检查。首先，系统会确认用户名是否存在且账户状态正常（未被禁用）。若用户名有效，则进入密码验证环节——由于密码采用加密存储，系统需先将用户输入的密码加密，再与数据库中的密文进行比对。验证通过后，系统会检查验证码的有效性（包括时效性和正确性）。全部验证通过后，系统将返回用户权限信息，并根据权限级别（普通用户或管理员）开放相应功能。用户登录的时序图如图 4‑7所示。

图形用户界面, 应用程序, Word

AI 生成的内容可能不正确。

图 4‑7 用户登录模块时序图

普通用户和超级管理员都可以管理账户信息，两者的管理逻辑基本一致，区别在于普通用户只能操作自己的账户，而超级管理员可以管理所有用户的账户。在修改账户名或密码时，如果是用户自己修改，系统会直接让当前用户下线；如果是超级管理员修改了某个正在登录的账户，系统会强制该账户下线，修改后该用户需要重新登录才能继续使用系统。对于修改账户的其他信息，只需要刷新页面，重新获取账户表单数据，就能看到最新内容。超级管理员停用某个账户时，如果该账户处于登录状态，也会将其强制下线。普通用户和超级管理员的账户管理流程分别如图 5.7 和图 5.8 所示。系统不支持直接删除账户，只能通过管理员进行启用或停用；如果确实需要删除账户，只能在数据库的账户信息表中操作，但不推荐这种方式。

## 数据库设计

系统通过识别搜索模块获取的数据，在用户确认后会被存储到数据库。普通用户可以直接管理自己上传的识别图像，而管理员则负责定期审核所有图像数据，剔除模糊、识别错误等低质量图片。系统的核心实体与实体内部的联系如图 4‑8所示。

图示

AI 生成的内容可能不正确。

图 4‑8 番茄叶片病虫害识别系统数据库ER图

#### 来源结果表

Admin表负责存储经过不同来源检索得到的歌曲结果，如表 4‑1所示。

表 4‑1 管理员Admin表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段描述 | 字段约束 |
| id | int | 管理员唯一编号 | 非空、自增、主键 |
| username | varchar(255) | 管理员登录用户名 |  |
| password | varchar(255) | 登录密码 |  |
| name | varchar(255) | 管理员姓名 |  |
| avatar | varchar(255) | 头像图片路径 |  |
| role | varchar(255) | 管理员角色类型 |  |
| phone | varchar(255) | 联系电话 |  |
| email | varchar(255) | 电子邮箱地址 |  |

#### 检索结果表

Camerarecords表负责储存音频检索后得到的结果，根据从不同来源得到的歌曲结果计算总侵权概率等，如表 4‑2所示。

表 4‑2 检索结果Query表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段描述 | 字段约束 |
| id | int | 记录唯一编号 | 非空、自增、主键 |
| weight | varchar(255) | 权重文件名称 |  |
| conf | varchar(255) | 置信度阈值 |  |
| username | varchar(255) | 操作用户名 |  |
| start\_time | varchar(255) | 操作开始时间 |  |
| out\_video | varchar(255) | 输出视频路径 |  |
| kind | varchar(255) | 检测种类 |  |

#### 会话信息表

Imgrecords表负责存储用户会话创建的详细信息，如表 4‑3所示。

表 4‑3 会话信息Session表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段描述 | 字段约束 |
| id | int | 记录唯一编号 | 非空、自增、主键 |
| input\_img | varchar(255) | 输入图像路径 |  |
| out\_img | varchar(255) | 输出图像路径 |  |
| confidence | varchar(255) | 检测置信度 |  |
| all\_time | varchar(255) | 完整处理耗时 |  |
| conf | varchar(255) | 置信度阈值 |  |
| weight | varchar(255) | 使用的模型权重文件 |  |
| username | varchar(255) | 操作用户名 |  |
| start\_time | varchar(255) | 开始处理时间 |  |
| label | varchar(255) | 检测目标标签 |  |
| kind | varchar(255) | 检测类型 |  |
| id | int | 记录唯一编号 | 非空、自增、主键 |

#### 会话信息表

User表负责储存会话信息与查询信息，如表 4‑4所示。

表 4‑4 会话与查询信息Result表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段描述 | 字段约束 |
| id | int | 用户唯一编号 | 非空、自增、主键 |
| username | varchar(255) | 用户名 |  |
| password | varchar(255) | 登录密码 |  |
| name | varchar(255) | 用户姓名 |  |
| sex | varchar(255) | 性别 |  |
| email | varchar(255) | 电子邮件地址 |  |
| tel | varchar(255) | 联系电话 |  |
| role | varchar(255) | 用户角色 |  |
| avatar | varchar(255) | 头像图片路径 |  |
| time | datetime | 注册时间 |  |

#### 音频信息表

Videorecords表负责储存版权音频库的信息，如表 4‑5所示。

表 4‑5 音频信息Audio表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段描述 | 字段约束 |
| id | int | 视频记录唯一编号 | 非空、自增、主键 |
| input\_video | varchar(255) | 输入视频文件路径 |  |
| out\_video | varchar(255) | 输出视频文件路径 |  |
| username | varchar(255) | 操作用户名 |  |
| start\_time | varchar(255) | 开始处理时间 |  |
| conf | varchar(255) | 置信度阈值 |  |
| weight | varchar(255) | 模型权重文件 |  |
| kind | varchar(255) | 检测类型 |  |

## 本章小结

本章进行了对基于深度学习的番茄叶片病虫害系统的功能需求分析，并主要分析了并发处理能力、识别准确率要求、响应性能要求三个方面的非功能性需求。基于以上需求分析结果，本章对系统进行了概要设计，根据功能将系统划分为三大功能模块：识别搜索模块、识别记录管理模块、账户管理模块。在详细设计部分使用UML类图与UML时序图对上述功能进行设计与补充，最后给出了系统的数据库表单设计。

# 番茄叶片病虫害识别系统的实现与测试

## 系统开发环境

本系统采用了B/S架构，前后端分离开发方式。算法实现语言采用了Python3.8，Web后端使用了Springboot框架架构配合Flask框架调用YOLOv8模型，使用Mysql数据库持久化数据；前端网页主要采用了vue300 +ElementUI技术。系统的开发环境配置如表 5‑1所示。

表 5‑1系统开发环境配置

|  |  |
| --- | --- |
| 组件 | 参数 |
| 服务器端CPU | Intel(R) Core(TM) i5-1035H CPU @ 1.00GHz |
| 服务器端内存 | 16GB |
| 服务器端GPU | NVIDIA GeForce MX 350 |
| 服务器端硬盘 | 250GB |
| 服务器操作系统 | Windows 10 64位 |
| 数据库及版本 | Mysql 8.0.11 |
| 前端语言及框架 | Vue3.0 / ElementUI |
| 后端语言及框架 | Java17.0.5 / SpringBoot3.0.5 / Mybatis-plus 3.5.3 |
| 开发软件版本 | IntelliJ IDEA 2022.3.2，Visual Studio Code 1.99.1 |
| 客户端浏览器配置 | chrome 125.0.6422.60及以上 |

## 系统功能模块实现

### 识别搜索模块实现

当用户在前端点击“批量上传”按钮后，选中文件并上传。系统后端接受一个MultipartFile的数组fileList，数组成员是上传的音频或视频。系统首先检测列表是否为空，若为空则给前端返回消息并让用户重新选择。若不为空则遍历列表，对每一个成员获取其文件名，将mp4文件转换为mp3文件，并保存到系统后端。代码如下所示：

用户(前端)发起图片上传请求, 调用FileController的upload()方法,把图片保存在Java服务端, 返回图片的url给前端；前端设置置信度，点击"预测"，将图片url发送给PredictionController的predict()方法, 然后PredictionController把图片url发给flask端的predictImg()方法，flask端调用模型预测，并把预测好的结果图片保存在flask端，然后把结果图片上传到Java端, 获取结果图的url, 返回给PredictionController, 最后PredictionController把结果url存到MySQL的imgrecords表。对视频的识别、摄像头的识别与此类似。图 5‑1、图 5‑2、图 5‑3展示了上传文件、点击识别显示识别结果的过程。

Java端的PredictionController核心代码如下：

public Result<?> predict(@RequestBody PredictRequest request) {  
 if (request == null || request.getInputImg() == null || request.getInputImg().isEmpty()) {  
 return Result.error("-1", "未提供图片链接");  
 } else if (request.getWeight() == null || request.getWeight().isEmpty()) {  
 return Result.error("-1", "未提供权重");  
 }  
  
 try {  
 // 创建请求体  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.APPLICATION\_JSON);  
 HttpEntity<PredictRequest> requestEntity = new HttpEntity<>(request, headers);  
  
 // 调用 Flask API  
 String response = restTemplate.postForObject("http://localhost:5000/predictImg", requestEntity, String.class);  
 System.out.println("java发的预测请求收到flask响应: Received response: \n" + response);  
 JSONObject responses = JSONObject.parseObject(response);  
 if(responses.get("status").equals(400)){  
 return Result.error("-1", "Error: " + responses.get("message"));  
 }else {  
// 把图片识别记录保存  
 ImgRecords imgRecords = new ImgRecords();  
 imgRecords.setWeight(request.getWeight());  
 imgRecords.setConf(request.getConf());  
 imgRecords.setKind(request.getKind());  
 imgRecords.setInputImg(request.getInputImg());  
 imgRecords.setUsername(request.getUsername());  
 imgRecords.setStartTime(request.getStartTime());  
 imgRecords.setLable(String.valueOf(responses.get("label")));  
 imgRecords.setConfidence(String.valueOf(responses.get("confidence")));  
 imgRecords.setAllTime(String.valueOf(responses.get("allTime")));  
 imgRecords.setOutImg(String.valueOf(responses.get("outImg")));  
 imgRecordsMapper.insert(imgRecords); // 插入到数据库  
 return Result.success(response);  
 }  
 } catch (Exception e) {  
 return Result.error("-1", "Error: " + e.getMessage());  
 }  
 }

Flask端调用模型并返回结果的核心代码如下：

class ImagePredictor:

    def \_\_init\_\_(self, weights\_path, img\_path, kind, save\_path="./runs/result.jpg", conf=0.5):

        """

        初始化ImagePredictor类

        :param weights\_path: 权重文件路径

        :param img\_path: 输入图像路径

        :param save\_path: 结果保存路径

        :param conf: 置信度阈值

        """

        self.model = YOLO(weights\_path)

        self.conf = conf

        self.img\_path = img\_path

        self.save\_path = save\_path

        self.kind = {

            'rice': ['Brown\_Spot（褐斑病）', 'Rice\_Blast（稻瘟病）', 'Bacterial\_Blight（细菌性叶枯病）'],

            'corn': ['blight（疫病）', 'common\_rust（普通锈病）', 'gray\_spot（灰斑病）', 'health（健康）'],

            'strawberry': ['Angular Leafspot（角斑病）', ' Anthracnose Fruit Rot（炭疽果腐病）', 'Blossom Blight（花枯病）', 'Gray Mold（灰霉病）', 'Leaf Spot（叶斑病）', 'Powdery Mildew Fruit（白粉病果）', 'Powdery Mildew Leaf（白粉病叶）'],

            'tomato': ['Early Blight（早疫病）', 'Healthy（健康）', 'Late Blight（晚疫病）', 'Leaf Miner（潜叶病）', 'Leaf Mold（叶霉病）', 'Mosaic Virus（花叶病毒）', 'Septoria（壳针孢属）', 'Spider Mites（蜘蛛螨）', 'Yellow Leaf Curl Virus（黄化卷叶病毒 ）']

        }

        self.labels = self.kind[kind]

    def predict(self):

        """

        预测图像并保存结果

        """

        start\_time = time.time()  # 开始计时

        # 执行预测

        results = self.model(source=self.img\_path, conf=self.conf, half=True, save\_conf=True)

        end\_time = time.time()  # 结束计时

        elapsed\_time = end\_time - start\_time  # 计算用时

        all\_results = {

            'labels': [],  # 存储所有标签

            'confidences': [],  # 存储所有置信度

            'allTime': f"{elapsed\_time:.3f}秒"

        }

        try:

            # 检查是否有检测结果

            if len(results) == 0:

                print("未检测到目标，请换一张图片。")

                all\_results = {

                    'labels': '预测失败',  # 存储所有标签

                    'confidences': "0.00%",  # 存储所有置信度

                    'allTime': f"{elapsed\_time:.3f}秒"

                }

                return all\_results

            for result in results:

                # 提取置信度和标签

                confidences = result.boxes.conf if hasattr(result.boxes, 'conf') else []

                labels = result.boxes.cls if hasattr(result.boxes, 'cls') else []

                # 检查 confidences 和 labels 是否为空

                if confidences.numel() == 0 or labels.numel() == 0:

                    print("未检测到目标，请换一张图片。")

                    all\_results = {

                        'labels': '预测失败',  # 存储所有标签

                        'confidences': "0.00%",  # 存储所有置信度

                        'allTime': f"{elapsed\_time:.3f}秒"

                    }

                    return all\_results

                # 获取标签名称和对应置信度

                label\_names = [self.labels[int(cls)] for cls in labels]

                predictions = list(zip(label\_names, confidences))

                # 将每个结果保存到字典中

                for label, conf in predictions:

                    all\_results['labels'].append(label)

                    all\_results['confidences'].append(f"{conf \* 100:.2f}%")

                result.save(filename=self.save\_path)  # 保存结果

            return all\_results  # 返回包含标签和置信度的字典

        except Exception as e:

            # 如果预测过程中发生异常，打印错误信息并返回空结果

            print(f"预测过程中发生异常: {e}")

            all\_results = {

                'labels': '预测失败',  # 存储所有标签

                'confidences': "0.00%",  # 存储所有置信度

                'allTime': f"{elapsed\_time:.3f}秒"

            }

            return all\_results

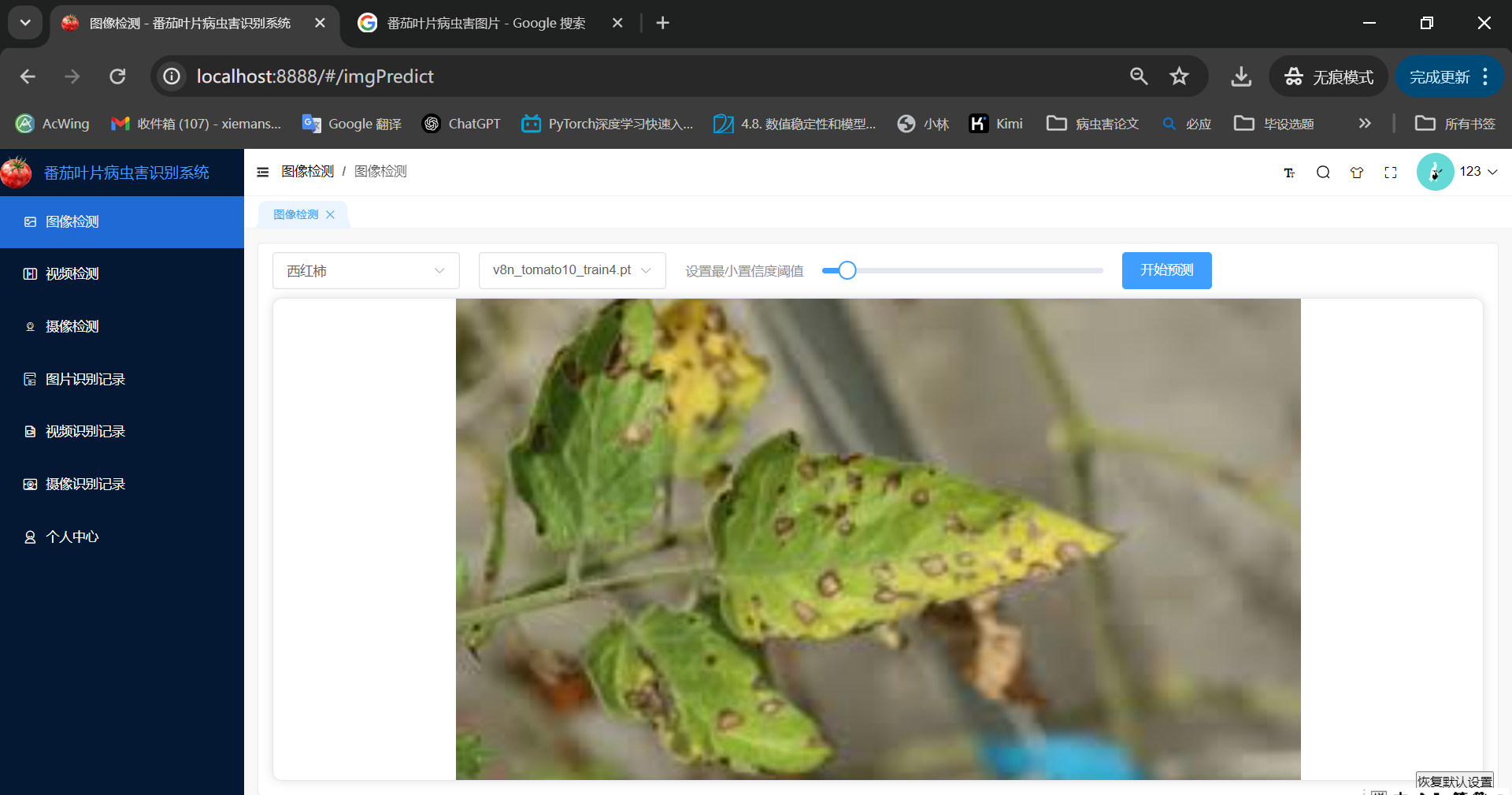


图 5‑1 上传待识别图片

图 5‑2 选择文件上传

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图 5‑3 成功上传文件

### 音频检索模块实现

用户在前端已经上传文件，并点击“批量查询”按钮后，系统在后台会自动开启原声查询、伴奏查询、旋律查询三种查询，均采用了Java与Python混合编程，经Java代码交互后调用Python脚本。三种查询的java代码处理流程类似，以原声查询代码为例，先由java代码开启ProcessBuilder调用Python脚本，在处理过程中使用BufferReader读取控制台输出，从控制台输出中实时读取处理进度并同步给HttpServletRequest.Session，在查询完成后解析json数据，将检索结果存入query与result表中，并返回前端处理结果。以原声查询为例，后端Java代码如下：

public String searchOrigin(String pythonPath, String pythonFilePath, String path, HttpServletRequest request) throws IOException, InterruptedException, JSONException {

// 用 ProcessBuilder 跑 Python 脚本

ProcessBuilder processBuilder = new ProcessBuilder(pythonPath,"-u",pythonFilePath,"--path",path);

Process process = processBuilder.start();

System.out.println(pythonPath + " -u " + pythonFilePath + "--path " + path);

// 用 BufferedReader 读取 Python 脚本的输出

BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream(),

StandardCharsets.UTF\_8));

String line;

JSONArray jsonArray = null;

while ((line = reader.readLine()) != null) {

System.out.println(line);

//同步进度

if (line.contains("originPercent:")) {

Integer originPercent = Integer.parseInt(line.substring("originPercent:".length(), line.indexOf("%")));

request.getSession().setAttribute("originPercent", originPercent);

}

//解析json数据

if(line.contains("origin\_search\_JSON:")){

jsonArray = new JSONArray(line.substring("origin\_search\_JSON:".length()));

}

}

//等待进程退出

int exitCode = process.waitFor();

System.out.println(pythonFilePath + " exited with code: " + exitCode);

if (exitCode != 0) { // 如果进程退出状态码不为0，则表示有错误

BufferedReader errorReader = new BufferedReader(new InputStreamReader(process.getErrorStream(), StandardCharsets.UTF\_8));

while ((line = errorReader.readLine()) != null) {

System.out.println(line);

}

System.out.println("调用python时发生错误");

}

process.destroy();//记得销毁进程，不然会内存泄漏

if(jsonArray != null){

//成功，存入数据库中

System.out.println("audioName:" + Paths.get(path).getFileName().toString().replace(".mp3",""));

ObjectMapper objectMapper = new ObjectMapper();

Query[] queryList = objectMapper.readValue(jsonArray.toString(),Query[].class);

for(int i =0;i<queryList.length;i++){

Result result = new Result();

result.setAudioName(Paths.get(path).getFileName().toString().replace(".mp3",""));

result.setSessionId(request.getSession().getId());

result.setSessionDatetime( new Timestamp( request.getSession().getCreationTime()));

queryMapper.insert(queryList[i]);

String queryUuid =queryList[i].getQueryUuid();

//存进result表中

result.setQueryUuid(queryUuid);

resultMapper.insert(result);

}

return jsonArray.toString();

}

return "error:数组为空";

}

在java代码中调用的Python脚本cli\_process\_origin如下：通过命令行方式调用，在参数中读入要处理的文件路径后，调用封装的Dejavu\_origin类的recognize\_djv\_from\_file()方法进行音频检索，在处理的前后以及处理的内部设定一些进度点，到进度点时实时打印进度值，调用sys.stdout.flush()来刷新控制台的缓存输出。在处理完成后，将得到的检索结果按照特定的格式打印到控制台，以供java后台解析。

import os

from dejavu\_master import Dejavu\_origin

import argparse

import sys

import io

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

def parse\_results\_to\_jsons(results):

"""将识别结果解析为JSON格式"""

print(results)

print(f"用时:{int(results['total\_time']):2}")

results\_parsed\_json = []

for index,result in enumerate(results['results']):

res\_json = {

'排名' : index + 1,

'song\_id' : result['song\_id'],

'曲名' : result['song\_name'].decode('utf-8'),

'输入置信度':result['input\_confidence'],

'指纹置信度' : result['fingerprinted\_confidence'],

'开始时间' : f"{int(result['offset\_seconds'])}",

'来源' : f"原声",

}

results\_parsed\_json.append(res\_json)

return results\_parsed\_json

def print\_json(resuls\_parsed\_json):

"""打印解析后的JSON结果"""

print("origin\_search\_JSON:",resuls\_parsed\_json)

if \_\_name\_\_ == '\_\_main\_\_':

# 设置命令行参数

parser = argparse.ArgumentParser()

parser.add\_argument('--path','-p',required=True)

args = parser.parse\_args()

file\_path = args.path

file\_name = os.path.dirname(file\_path)

#控制台同步原声查询的进度

import datetime

print(f"originPercent:{1}% End,{datetime.datetime.now()}\n");sys.stdout.flush()

#原声识别处理

print('----原声识别文件:',file\_name,'----')

#调用dejavu代码，启动Shazam处理流程

djv\_ori = Dejavu\_origin()

#开始识别文件

results = djv\_ori.recognize\_djv\_from\_file(file\_path)

#打印结果

print(f"originPercent:{100}% End,{datetime.datetime.now()}");sys.stdout.flush()

print\_json(parse\_results\_to\_jsons(results))

伴奏查询的java代码与python代码均与原声查询的实现类似，只是在处理前需使用UVR进行伴奏分离，此处不再展示。旋律查询的Python代码有所不同，展示如下：首先仍然是从命令行参数中读入要处理的文件路径与保存路径，然后调用封装的深度神经网络Melody\_Extractor类的predict\_pitch\_single\_file()方法进行旋律基频的预测，在生成预测文件后调用set\_time\_standard函数将文件的时间与格式统一以便后续检索，随后调用Lsh\_query.query()方法对基频序列文件进行检索，将检索的结果按特定格式输出，以供java后端读取解析。

if \_\_name\_\_ == '\_\_main\_\_':

import datetime

# 控制台同步旋律查询的时间和进度

print(f"melodyPercent:{1}% Start,{datetime.datetime.now()}");sys.stdout.flush()

# 设置命令行参数

parser = argparse.ArgumentParser()

parser.add\_argument('--file\_path','-p',required=True)

parser.add\_argument('--save\_path','-s',required=True)

args = parser.parse\_args()

file\_path = args.file\_path

file\_name = os.path.dirname(file\_path)

save\_path = args.save\_path

print("正在查询：" + os.path.basename(file\_path))

# 初始化CFTA\_AFF\_SA网络旋律提取器，设置gpu

extractor = Melody\_Extractor(cuda\_id=0)

print(f"melodyPercent:{5}% Predict\_start,{datetime.datetime.now()}");sys.stdout.flush()

# 调用封装的预测旋律基频方法，并保存结果

pitch\_path = extractor.predict\_pitch\_single\_file(file\_path,save\_path)

print(f"melodyPercent:{80}% End,{datetime.datetime.now()}\n");sys.stdout.flush()

# 统一旋律文件的时间和格式，以便后续检索

set\_time\_standard(pitch\_path,pitch\_path)

print("pitch\_std生成完成...")

# 初始化LSH查询对象

lsh\_query = Lsh\_query()

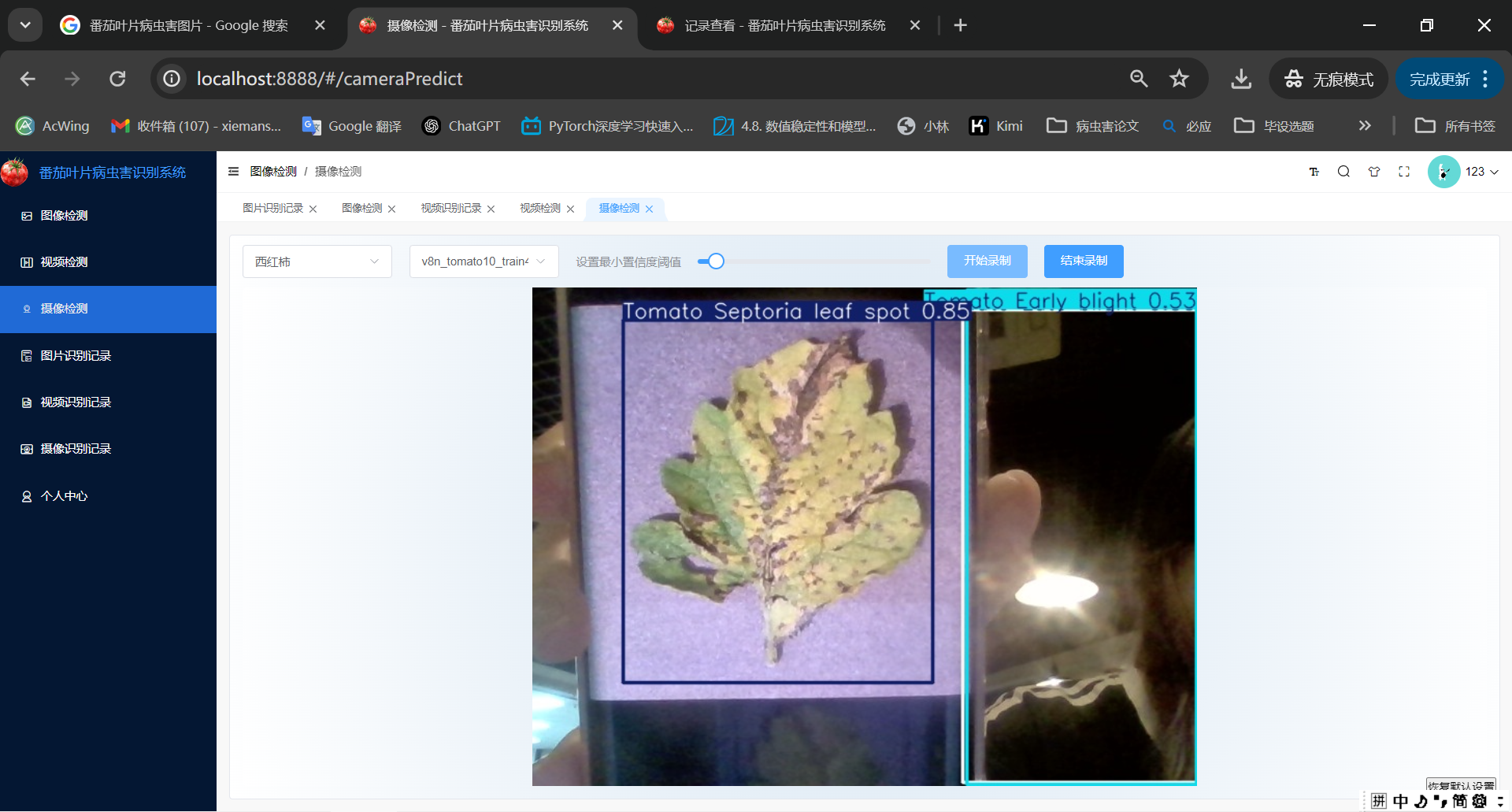
print(f"melodyPercent:{85}% End,{datetime.datetime.now()}\n");sys.stdout.flush()

#LSH查询，检索返回结果

results = lsh\_query.query(pitch\_path)

print(f"melodyPercent:{100}% End,{datetime.datetime.now()}\n");sys.stdout.flush()

print("melody\_search\_JSON:",results)



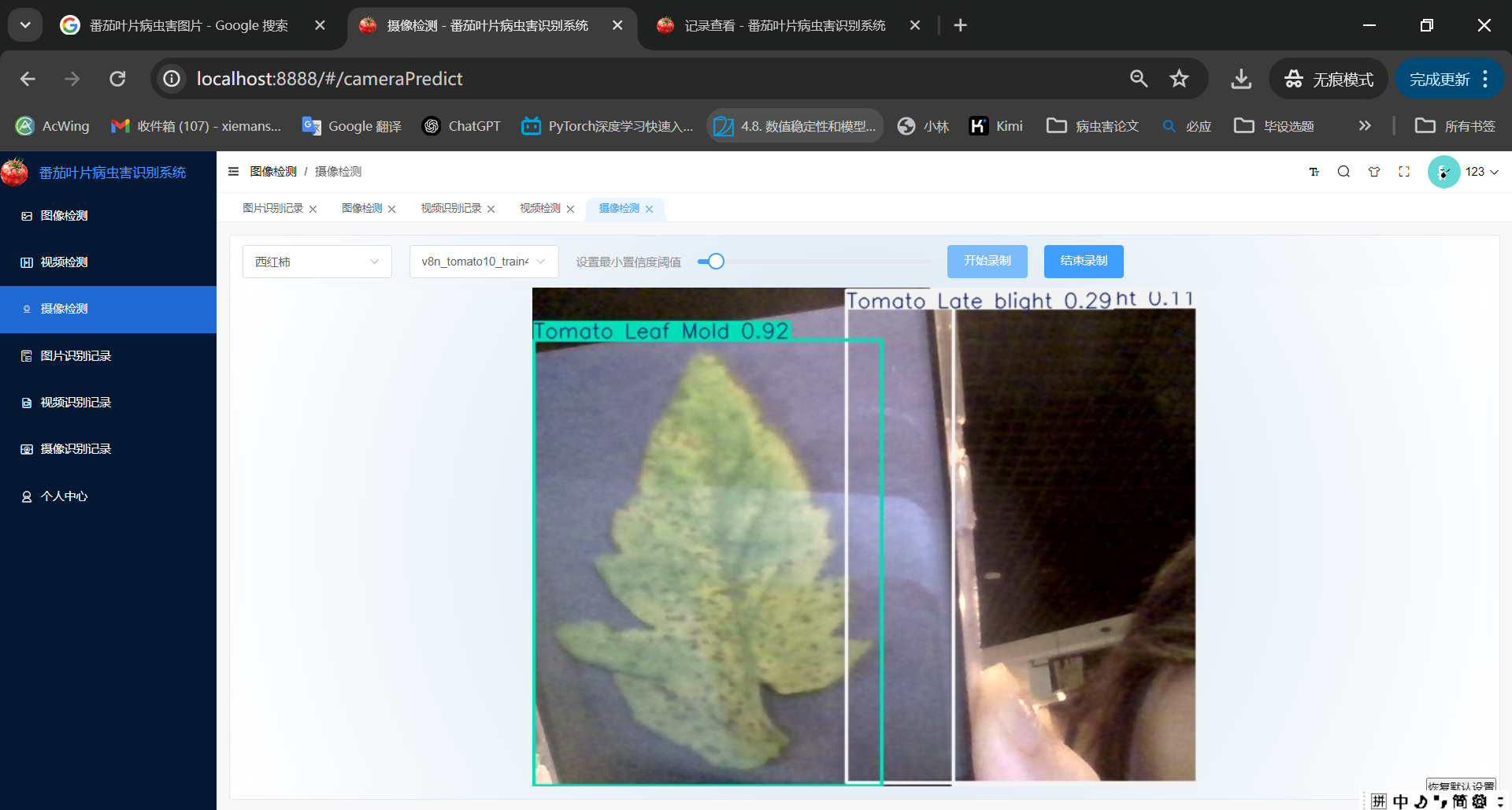


图 5‑4 批量查询并实时同步文件处理进度

电脑软件的截图

AI 生成的内容可能不正确。

图 5‑5 对单曲进行旋律查询并同步处理进度

### 查看检索结果模块实现

本小节将展示查看检索模块中的查看侵权概率总表功能的实现代码。showFullTable()方法首先接收前端传入的sessionId，首先从Query表中将等于这个sessionId的所有音频名取出作为audioNameList，然后遍历每一个查询的音频名，将检索结果全部取出。如果检索结果没有重复，则直接将检索结果的置信度query.getConfidence()作为音频-歌曲二元组之间的侵权概率。如果不同的来源之间检索得到了同样的歌曲结果，则使用基于对数和的置信度放大方法，迭代计算总侵权概率，最后将得到的json结果返回给前端的侵权概率总表进行显示。

public String showFullTable(String sessionId){

//应该能保证，每次调用的时候表里都不会是空的

//把结果转成json输出

ObjectMapper objectMapper = new ObjectMapper();

ArrayNode jsonArray = objectMapper.createArrayNode();

//把这个sessionId里面所有的音频名都抓出来

QueryWrapper<Result> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.select("DISTINCT audio\_name");

queryWrapper1.eq("session\_id",sessionId);

List<Object> audioNameList = resultMapper.selectObjs(queryWrapper1);

System.out.println("audioNameList = " + audioNameList);

//对每个音频名循环，抓出15个结果

for(int i =0;i < audioNameList.size();i++){

String audioName = (String) audioNameList.get(i);

//获取所有query\_uuid

QueryWrapper<Result> queryWrapper = new QueryWrapper<>();

queryWrapper.select("query\_uuid");

queryWrapper.eq("session\_id",sessionId);

queryWrapper.eq("audio\_name",audioName);

List<Object> queryUuidList = resultMapper.selectObjs(queryWrapper);

// 创建一个映射用于存储歌曲结果和对应的侵权概率

Map<String ,Double> probabilityMap = new HashMap<>();

// 遍历每个查询 UUID，计算侵权概率并存入映射

for(int j = 0; j<queryUuidList.size();j++){

Query query = queryMapper.selectById((String)queryUuidList.get(j));

String songResult = query.getSongResult();

//检索置信度

double infringe\_probability = query.getConfidence();

if (!probabilityMap.containsKey(songResult)){

probabilityMap.put(songResult,infringe\_probability);

}

else {

//map中有这个key，说明重复，需要使用置信度放大方法额外计算概率

System.out.println("重复，对数和放大概率，" +audioName +songResult);

infringe\_probability = 1- (1-infringe\_probability)\*(1-probabilityMap.get(songResult));

probabilityMap.put(songResult,infringe\_probability);

}

}

System.out.println("map:" +probabilityMap);

// 将计算结果转换为 JSON 对象，并添加到 JSON 数组中

for (Map.Entry<String, Double> entry : probabilityMap.entrySet()) {

ObjectNode jsonObject = objectMapper.createObjectNode();

jsonObject.put("audioName",audioName);

jsonObject.put("songResult",entry.getKey());

jsonObject.put("infringeProbability",String.format("%.1f",entry.getValue() \* 100) );

System.out.println("jsonObject:" + jsonObject);

jsonArray.add(jsonObject);

}

}

return jsonArray.toString();

}

电脑软件的截图

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图 5‑6 查看按侵权概率由高到低排序的侵权概率总表

### 历史会话模块实现

本小节将以历史会话模块中的“查看历史会话”为例展示实现过程。当用户在前端点击“历史会话页面”时，java后端就会将Result表中所有的查询结果取出，将每一个sessionId对应的所有查询过的音频名audioName都封装到一起，并按照会话的创建时间session\_datetime进行降序排序，以供用户在前端查看每个会话的具体情况。

public String showAllSessionTable() {

// 创建查询条件，选择所有不同的 session\_id 和 session\_datetime，并按 session\_datetime 降序排列

QueryWrapper<Result> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.select("DISTINCT session\_id,session\_datetime");

queryWrapper1.orderByDesc("session\_datetime");

List<Result> resultList = resultMapper.selectList(queryWrapper1);

int length = resultList.size();

SessionTableResult[] sessionTableResults = new SessionTableResult[length]; // 声明一个长度为5的对象数组

for (int i = 0; i < length ;i++){

String sessionId = resultList.get(i).getSessionId();

Timestamp sessionDatetime = resultList.get(i).getSessionDatetime();

QueryWrapper<Result> queryWrapper = new QueryWrapper<>();

// 创建查询条件，获取对应 session\_id 的所有不同的 audio\_name

queryWrapper.select("DISTINCT audio\_name");

queryWrapper.eq("session\_id", sessionId);

String[] audioNames = resultMapper.selectObjs(queryWrapper).toArray(new String[0]);

SessionTableResult sessionTableResult = new SessionTableResult();

sessionTableResult.setSessionId(sessionId);

sessionTableResult.setSessionDatetime(sessionDatetime);

sessionTableResult.setAudioNames(audioNames);

// 将对象存入数组

sessionTableResults[i] = sessionTableResult;

}

try {

String json = objectMapper.writeValueAsString(sessionTableResults); // 将 SessionTableResults数组转换为 JSON 字符串

return json; // 返回 JSON 字符串

} catch (JsonProcessingException e) {

e.printStackTrace();

return "{\"error\": \"Failed to serialize JSON\"}"; // JSON 序列化失败时返回错误信息

}

}

电脑软件的截图

AI 生成的内容可能不正确。

图 5‑7 查看历史查询会话页面

### 音乐播放器模块实现

音乐播放器在前端采用vue3+typescript语言实现。以左侧的音乐播放器为例，代码如下。播放器可分为player-content，time, sArea三个部分，player-content部分用于显示音频名基本信息和播放器的播放/暂停按钮，点击时会触发PlayPause1()方法来控制音频的暂停和播放；time部分用于显示歌曲目前的进度与音频总时长；sArea部分用于显示歌曲的进度条信息，当鼠标悬停时会触发showHover1()方法显示鼠标位置处的音频时间，鼠标点击时会触发playClick1()方法，跳转到鼠标指定的音频位置进行播放。

<template>

<audio ref="audioPlayer1" :src="audioSource1"></audio>

<div class="player" id="player1">

<div class="player-content">

<div ref="musicName1" class="music-name"></div>

<div class="player-controls">

<div class="btn prev iconfont">&#xe603;</div>

<div ref="playPauseBtn1" :class="playPauseBtn1Class" @click="PlayPause1"></div>

<div class="btn next iconfont">&#xe602;</div>

</div>

<div ref="time" class="time">

<div ref ="tProgress" class="current-time"></div>

<div ref="totalTime" class="total-time"></div>

</div>

<!-- 进度条 -->

<div ref="sArea1" @mousemove="showHover1" @mouseleave="hideHover1" @click="playClick1" id="s-area">

<div ref='instime' id="ins-time">'--'</div>

<div ref="sHover1" id="s-hover"></div>

<div ref="seekBar" id="seek-bar"></div>

</div>

</div>

</div>

</template>

图形用户界面, 文本

AI 生成的内容可能不正确。

图 5‑8 音乐播放器悬停进度条

# 结论与展望

## 标题2

### 标题3

公式按章重新编号：

 （6-1）

公式（6-1）说明，…………（公式在正文中的引用）

图题注：

图 6‑1 XXXXXX

# XXXXXXX（标题1）

## 标题2

### 标题3

公式按章重新编号：

 （7-1）

公式（7-1）说明，…………（公式在正文中的引用）

图题注：

图 7‑1 XXXXXX

# XXXXXXXX（标题1）

## 标题2

### 标题3

公式按章重新编号：

 （8-1）

公式（8-1）说明，…………（公式在正文中的引用）

图题注：

图 8‑1 XXXXXX

# XXXXXXXXX（标题1）

## 标题2

### 标题3

公式按章重新编号：

 （9-1）

公式（9-1）说明，…………（公式在正文中的引用）

图题注：

图 9‑1 XXXXXX

# XXXXXXXXXX（标题1）

## 标题2

### 标题3

公式按章重新编号：

 （10-1）

公式（10-1）说明，…………（公式在正文中的引用）

图题注：

图 10‑1 XXXXXX

# XXXXXXXXXXX（标题1）

## 标题2

### 标题3

公式按章重新编号：

 （11-1）

公式（11-1）说明，…………（公式在正文中的引用）

图题注：

图 11‑1 XXXXXX

# 结论与展望

结论与展望：结论包括对整个研究工作进行归纳和综合而得出的总结；所得结果与已有结果的比较；联系实际结果，指出它的学术意义或应用价值和在实际中推广应用的可能性；在本课题研究中尚存在的问题，对进一步开展研究的见解与建议。结论集中反映作者的研究成果，表达作者对所研究课题的见解和主张，是全文的思想精髓，是全文的思想体现，一般应写得概括、篇幅较短。撰写时应注意下列事项：

（1）结论要简单、明确。在措辞上应严密，但又容易被人领会。

（2）结论应反映个人的研究工作，属于前人和他人已有过的结论可少提。

（3）要实事求是地介绍自己研究的结果，切忌言过其实，在无充分把握时，应留有余地。

## 标题2

### 标题3

公式按章重新编号：

 （12-1）

公式（12-1）说明，…………（公式在正文中的引用）

图题注：

图 12‑1 XXXXXX

# 致 谢

致谢是作者对论文的生成作过贡献的组织或个人予以感谢的文字记录，内容应客观、真实，语言宜诚恳、真挚、恰当。本科毕业设计（论文）致谢中一般主要感谢导师和对论文工作有直接贡献和帮助的人士和单位。

**一般致谢的内容有：**

（一）对指导或协助指导完成论文的导师；

（二）对国家科学基金、资助研究工作的奖学金基金、合同单位、资助或支持的企业、组织或个人；

（三）对协助完成研究工作和提供便利条件的组织或个人；

（四）对在研究工作中提出建议和提供帮助的人；

（五）对给予转载和引用权的资料、图片、文献、研究思想和设想的所有者；

（六）对其他应感谢的组织和个人。

致谢言语应谦虚诚恳，实事求是。字数不超过1000汉字。

**致谢格式规范：**

1. **标题部分：**居中编排“致 谢”二字（三号宋体），字间空两个半角字符。段落设置：段前2行，段后1行，行距1.2倍。
2. **致谢正文部分：**中文小四宋体，数字及英文字体均为Times New Roman，每段首行缩进2字符起排，段与段之间不空行。段落设置：行距1.2倍。

# 参考文献

（此上两空行不能删除，是为EndNote的参考文献列表所预留）

文后著录的参考文献务必实事求是。论文中引用过的文献必须著录，未引用的文献不得出现。应遵循学术道德规范，避免涉嫌抄袭、剽窃等学术不端行为。

参考文献一般应是作者亲自考察过的对学位论文有参考价值的文献，除特殊情况外，一般不应间接引用。

参考文献应有权威性，应注意所引文献的时效性。

参考文献的数量一般**不少于30篇，且均以近5年的文献为主**。

**参考文献格式规范：**

1. **标题部分：**居中编排“参考文献”四字（三号宋体）。段落设置：段前2行，段后1行，行距1.2倍。
2. **字体字号：**首行居中编排“参考文献”四字（三号宋体）。段落设置：段前2行，段后1行，行距1.2倍。参考文献内容采用五号字（中文宋体，数字及英文字体均为Times New Roman）。
3. **参考文献的著录格式：**应符合国家标准GB/T 7714-2015《文后参考文献著录规则》。参考文献中每条项目应齐全。
4. **参考文献中的标点符号：**英文文献用半角，中文文献用全角。
5. **参考文献的组织方式：**可采用顺序编码制组织，也可以采用著者-出版年制组织，但一篇论文只能选择其中一种引用方式。
6. **顺序编码制**

顺序编码制是指各篇文献按照正文部分标注的顺序连续编码，标注时将序号置于方括号中。

文献中的作者不超过三位时全部列出，超过三位时，一般只列前三位，中文姓名后面加 “，等”，英文姓名后面加 “, et al”，作者姓名之间用逗号分开。

外国人名一般采用姓在前，名在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，姓和名之间空1格，如：“Metcalf S W”。也可采用名在前，姓在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，名和姓之间空1格，如：“SW Metcalf”。

中文人名的英文表达方式：

简写时，采用姓在前，名在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，姓和名之间空1格，如，“钱学森”，简写为“Qian XS”。

全拼时，采用名在前，姓在后的著录法，名的第一个字母大写，名连写，名后空1格写姓，姓的第一个字母大写。如，“钱学森”，写为“Xuesen Qian”。

**具体要求如下：**

A 专著（包括普通图书［M］、论文集和会议录［C］、科技报告［R］、学位论文［D］、标准［S］）

主要责任者．文献题名［文献类型标志］．其他责任者．版本项（第1版不标注）．出版地：出版者，出版年：引文页码．获取和访问路径．

B 专著中的析出文献

析出文献主要责任者．析出文献题名[文献类型标志]．析出文献其他责任者//专著主要责任者．专著题名：其他题名信息. 版本项（第1版不标注）．出版地：出版者，出版年：析出文献的起止页码．获取和访问路径．

C连续出版物

主要责任者．题名：其他题名信息［文献类型标志］．年，卷（期）－年，卷（期）.出版地：出版者，出版年．获取和访问路径．

D连续出版物中的析出文献（包括期刊中析出的文献[J]、报纸中析出的文献[N].）

析出文献主要责任者．析出文献题名［文献类型标志］．连续出版物题名：其他题名信息，年，卷（期）：页码．获取和访问路径．

E专利文献

专利发明者/专利申请者或所有者．专利题名: 专利国别,专利号［文献类型标志］.公告日期或公开日期. 获取和访问路径．

F电子文献（包括专著或连续出版物中析出的电子文献）

主要责任者．题名：其他题名信息[文献类型标志/载体类型标志]．出版地：出版者，出版年（更新或修改日期）．获取和访问路径．

表1 文献类型和标志代码

|  |  |  |  |
| --- | --- | --- | --- |
| 文献类型 | 标志代码 | 文献类型 | 标志代码 |
| 普通图书 | M | 会议录 | C |
| 汇编 | G | 报纸 | N |
| 期刊 | J | 学位论文 | D |
| 报告 | R | 标准 | S |
| 专利 | P | 数据库 | DB |
| 计算机程序 | CP | 电子公告 | EB |

表2 电子文献载体和标志代码

|  |  |  |  |
| --- | --- | --- | --- |
| 载体类型 | 标志代码 | 载体类型 | 标志代码 |
| 磁带（magnetic tape） | MT | 磁盘（disk） | DK |
| 光盘（CD-ROM） | CD | 联机网络（online） | OL |

**样例：**

1. 刘国钧，郑如斯．中国书的故事［M］．北京：中国青年出版社，1979：110-115．
2. 昂温 G．外国出版史［M］．陈生铮译．北京：中国书籍出版社，1988．
3. 辛希孟．信息技术与信息服务国际研讨会论文集：A集［C］．北京：中国社会科学出版社，1979．
4. 冯西桥．核反应堆压力容器的LBB分析［R］．北京：核能技术设计研究院，1997．
5. 张和生．地质力学系统理论［D］．太原：太原理工大学，1998．
6. 全国文献工作标准化技术委员会第七分委员会．GB/T 5795-1986．中国标准书号［S］．北京：中国标准出版社，1986．
7. 罗云．安全科学理论体系的发展及趋势探讨［M］//白春华，何学秋，吴宗之．21世纪安全科学与技术的发展趋势．北京：科学出版社，2000：1-5．
8. 钟文发．非线性规划在可燃毒物配置中的应用［C］//赵玮．运筹学的理论与应用：中国运筹学会第五届大会论文集．西安：西安电子科技大学出版社，1996：468－471．
9. 高义民，张凤华，邢建东等．颗粒增强不锈钢基复合材料冲蚀磨损性能研究［J］．西安交通大学学报，2001，35(7)：727-730．
10. Papworth A, Fox P, Zeng GT, et al. Ability of aluminum alloy to wet alumina fibres by addition of bismuth[J]. Mater Sci & Technol, 1999, 15(4): 419-428.
11. 丁文祥．数字革命与竞争国际化［N］．中国青年报，2000－11－20(15)．
12. 姜锡洲．一种温热外敷药制备方案：中国，881056078［P］．1989-07-26．
13. Koseki A, Momose H, Kawahito M, et al. Complier: US, 828402［P/OL］2002-05-25 [2002-05-28]. http://FF&p.
14. Online Computer Library Center, Inc. History of OCLC[EB/OL]. [2000-01-08]. http://www.clc.org/ about/history/default.htm.
15. 江向东．互联网环境下的信息处理与图书管理系统解决方案［J/OL］．情报学报，1999，18(2): 4[2000-01-18].http://www.chinainfo.gov.cn/periodical/qbxb．
16. Scitor C. Project scheduler[CP/DK]. Sunnyvale, Calif. : Scitor Corp, 1983.
17. Metcalf SW. The Tort Hall air emission study[C/OL]//The International Congress on Hazardous Waste, MarquisHotel, Atlanta, Georgia, June 5-8, 1995: impact on human and ecological health[1998-09-22]. <http://atsdrl>.atsdr.cdc.gov:8080/cong95.html.

**2. 著者-出版年制**

著者-出版年制是指各篇文献首先按照文种集中，然后按照著者字顺和出版年排列，标注时将著者姓氏与出版年置于圆括号内。

1. 正文引用的文献采用著者-出版年制时，各篇文献的标注内容由著者姓氏与出版年构成，并置于“（ ）”内，倘若只标注著者姓氏无法识别该人名时，可标著者姓名，例如中国人、韩国人、日本人用汉字书写姓名。集体著者著述的文献可标注机关团体名称。倘若正文中已提及著者姓名，则在其后的“（ ）”内只著录出版年。
2. 正文中引用多著者文献时，对欧美著者只需标注第一个著者的姓，其后附“et al.”“等”之间留适当空隙。

C. 在参考文献表中著录同一著者在同一年出版的多篇文献时，出版年后应用小写字母a, b, c ...区别。

D. 多次引用同一著者的同一文献，在正文中标注著者与出版年，并在“（ ）”外以角标的形式著录引文页码。

**样例：**

BAKER S K, JACKSON M E. 1995. The future of resource sharing [M].

New York: The Haworth Press.

尼葛洛庞帝．1996. 数字化生存［M］．胡永，范海燕，译. 海口：海南出版社．

杨宗英．1996. 电子图书馆的现实模型［M］．中国图书馆学报（2）：24-29．

刘斌．2014. 力学［M］．合肥：中国科学技术大学出版社．

# 附 录

**用于抽检的论文，附录内容全部隐去。**

附录编号依次编为附录A，附录B……。附录标题各占一行，按一级标题编排。每一个附录一般应另起一页编排，如果有多个较短的附录，也可接排。附录中的图表公式另行编排序号，与正文分开，编号前加“附录A-”字样，（序号如：附录A-图1……附录B-图1；附录A-表1……附表B-表1；（附录A-1）……（附录B-1））。

附录包括：A外文原文、B外文翻译、C有关图纸、D计算机源程序……，此部分内容页眉同上文（奇数页为一级标题内容，偶数页为“西安交通大学本科毕业设计（论文）”）

附录后依次编排《**任务书》**（双面打印）、**《考核评议书》**（双面打印，背面是《评审意见书》）和**《答辩结果》。此部分内容不再额外添加“附录X”一级标题，页眉和排版严格参照模板文件**。《考核评议书》、《评审意见书》和《答辩结果》必须分别由指导教师、评阅人和答辩组据实填写。

以上全部内容须与论文**装订在一起**，且页码必须接着参考文献的页码连续编写。

**装订顺序如下：**

1）外文原文

2）外文翻译

3）有关图纸

4）计算机源程序等

5）任务书（奇数页起排）

6）考核评议书（奇数页起排，背面是《评审意见书》）

7）答辩结果（奇数页起排）

1. 脚注是对文中有关内容的解释、说明或补充，使用上角标（序号①、②…）标注，脚注可用小号字（一般小五号宋体）列在相应正文同一页最下部并与正文部分用细线（版面宽度的1/4长）隔开。（删除脚注的方法：直接删除正文中的脚注编号即可） [↑](#footnote-ref-1)