****

**义眼盯真——图像篡改检测系统**

**项目计划书**

目录

[1 引言 4](#_Toc192090767)

[1.1 编写目的 4](#_Toc192090768)

[1.2 项目背景 4](#_Toc192090769)

[1.3 定义 4](#_Toc192090770)

[1.4 项目内容 4](#_Toc192090771)

[2 项目概述 5](#_Toc192090772)

[2.1 项目目标 5](#_Toc192090773)

[2.2 产品目标与范围 5](#_Toc192090774)

[2.2.1 技术性能目标 5](#_Toc192090775)

[2.2.2 用户体验目标 5](#_Toc192090776)

[2.2.3 功能范围 5](#_Toc192090777)

[2.3 应交付成果 6](#_Toc192090778)

[2.3.1 需完成的软件 6](#_Toc192090779)

[2.3.2 需提交用户的文档 6](#_Toc192090780)

[2.3.3 须提交内部的文档 6](#_Toc192090781)

[2.3.4 应当提供的服务 6](#_Toc192090782)

[2.4 项目开发环境 6](#_Toc192090783)

[3 项目团队组织 8](#_Toc192090784)

[3.1 人员分工 8](#_Toc192090785)

[4 实施计划 9](#_Toc192090786)

[4.1 风险评估及对策 9](#_Toc192090787)

[4.2 总体进度计划 9](#_Toc192090788)

[4.2.1 第一阶段：项目启动与规划（第1-2周） 9](#_Toc192090789)

[4.2.2 第二阶段：需求分析与软件设计（第3-4周） 9](#_Toc192090790)

[4.2.3 第三阶段：核心功能开发（第5-9周） 9](#_Toc192090791)

[4.2.4 第四阶段：系统集成与测试（第10-11周） 10](#_Toc192090792)

[4.2.5 第五阶段：系统优化与部署运行（第12周-第13周） 10](#_Toc192090793)

[4.3 项目里程碑计划 11](#_Toc192090794)

[5 预算 11](#_Toc192090795)

[5.1 人员成本 11](#_Toc192090796)

[5.2 设备成本 11](#_Toc192090797)

[5.3 其他经费预算 12](#_Toc192090798)

[5.4 总经费预算 12](#_Toc192090799)

[参考资料 13](#_Toc192090800)

# 引言

## 编写目的

本文档旨在明确"义眼盯真-AI图像篡改检测系统"的项目目标、实施计划及资源配置，为项目开发提供规范性指导，确保团队高效协作并按时交付成果。

## 项目背景

随着AI生成图像技术（如Deepfake）的普及，虚假图像/视频引发的诈骗、虚假新闻等问题日益严重。当前市场缺乏高效、易用的图像真实性检测工具。本项目基于深度学习技术，开发一套支持多场景、高精度的图像篡改检测系统，满足新闻媒体、司法取证、社交平台等领域的迫切需求，为构建可信数字生态提供关键技术基础设施。

## 定义

图像篡改识别模型：基于YOLO-11n-seg优化的深度学习模型，集成目标检测与实例分割功能。

多任务学习框架：同时执行图像分类（真伪判断）和区域定位（篡改区域标识）的算法架构。

数据脱敏技术：采用差分隐私（Differential Privacy）和区域模糊化处理，对用户上传图像中的人脸、车牌等敏感信息进行不可逆加密，满足GDPR与《个人信息保护法》合规要求。

跨平台兼容性：采用统一推理引擎（ONNX Runtime）实现模型格式标准化，确保Web/PC/移动端检测结果一致性误差≤1%。

## 项目内容

本项目致力于构建一套全栈式AI图像篡改检测系统，核心技术架构涵盖从数据采集、模型训练到服务部署的全流程闭环。系统以基于YOLO-11n-seg改进的多模态检测模型为核心，实现对Deepfake深度伪造、Photoshop局部篡改、图像拼接移植等主流伪造手段的精准识别，检测粒度可细化至像素级篡改轨迹分析。通过边缘检测算法，系统能够对图像中疑似篡改区域进行动态标注，支持以颜色梯度呈现篡改置信度分布，并生成包括篡改分析的交互式检测报告。针对企业级用户需求，开发支持50张/批次并行处理的批量检测模块，同时提供标准化RESTful API接口，可无缝对接社交媒体内容审核系统、电子证据管理平台等第三方应用。在终端适配方面，采用React+Electron构建跨平台桌面客户端，结合Uniapp框架实现微信小程序轻量化部署，确保在iOS/Android设备上单图检测耗时≤5秒。系统同步构建用户反馈闭环机制，通过可信结果标注功能持续收集边缘案例数据，驱动模型在线增量学习，形成检测能力动态进化体系。

# 项目概述

## 项目目标

* 在13周内完成系统开发并通过验收测试
* 控制总预算≤150,000元
* 实现团队协作效率提升（任务完成率≥95%）

## 产品目标与范围

### 技术性能目标

* 图像篡改检测准确率≥85%（覆盖Deepfake、PS等主流伪造技术）
* 单图检测响应时间≤3秒（Web端），移动端≤5秒

### 用户体验目标

* 界面操作友好度评分≥4.5/5.0
* 核心用户群体（普通用户）满意度≥90%

### 功能范围

* 单图/批量检测
* 篡改热图可视化
* 定制化检测报告
* 微信小程序轻量化服务

## 应交付成果

### 需完成的软件

* B/S架构Web端检测系统
* 微信小程序移动端
* 开放API接口

### 需提交用户的文档

* 《用户操作手册》
* 《API接口文档》

### 须提交内部的文档

* 《项目计划书》
* 《需求规格说明书》
* 《技术设计方案》
* 《开发报告》
* 《测试报告》

### 应当提供的服务

* 系统部署与运维支持
* 用户培训与技术咨询。

## 项目开发环境

项目开发基于Python 3.12编程，采用PyTorch深度学习框架及OpenCV图像处理库构建核心算法模块。开发环境通过Anaconda进行虚拟化隔离，确保跨平台一致性。代码开发与调试依托PyCharm Professional完成，其集成调试器与数据库管理工具可有效支持多模块协同开发。

团队协作采用PingCode作为项目管理平台，基于Epic-Story-Task三级任务拆解机制实现敏捷开发流程，需求任务与代码提交（Commit ID）通过系统自动化关联，确保开发进度可追溯。代码版本控制通过GitCode私有仓库管理，遵循Git Flow分支策略，主分支（main）受保护策略限制，功能开发均基于dev分支提交合并请求（Pull Request），技术文档通过GitCode Wiki以Markdown标准化归档。

测试验证环节配置Postman进行API接口自动化测试，结合PyTest框架实现单元测试覆盖率≥85%。系统部署采用Docker容器化方案，基于Alpine Linux镜像封装模型服务，确保开发、测试与生产环境的全链路一致性。环境验证标准包括：Anaconda依赖清单与设计文档完全匹配、核心检测模块在PyCharm调试环境下响应时间≤3秒。

# 项目团队组织

## 人员分工

本团队由5名组员组成，采用主次责任制，重点覆盖AI模型训练、图像算法设计和前端开发、后端开发四个核心方向。其中图像算法设计和前端开发由两名成员共同负责，以确保这些关键模块的开发质量和进度。具体分工如下：

1.刘信方：项目经理，AI模型训练

主要职责：项目管理、深度学习模型训练与优化、模型评估与验证、模型部署与集成、训练数据集处理

协助工作：参与算法优化、协助后端开发、支持技术文档编写

2.张芳博：图像算法设计

主要职责：图像处理算法研发、算法性能优化

协助工作：参与AI模型优化、协助系统测试、支持文档编写

3.王劲东：后端开发

主要职责：系统整体架构设计、后端服务开发与维护、数据库设计与优化、API接口设计与实现

协助工作：参与系统测试、协助文档编写

4.袁崇昕：前端开发

主要职责：前端功能实现、用户交互设计

协助工作：参与系统测试、协助文档编写

5.李康：运维与答辩

主要职责：系统部署与运维、现场答辩

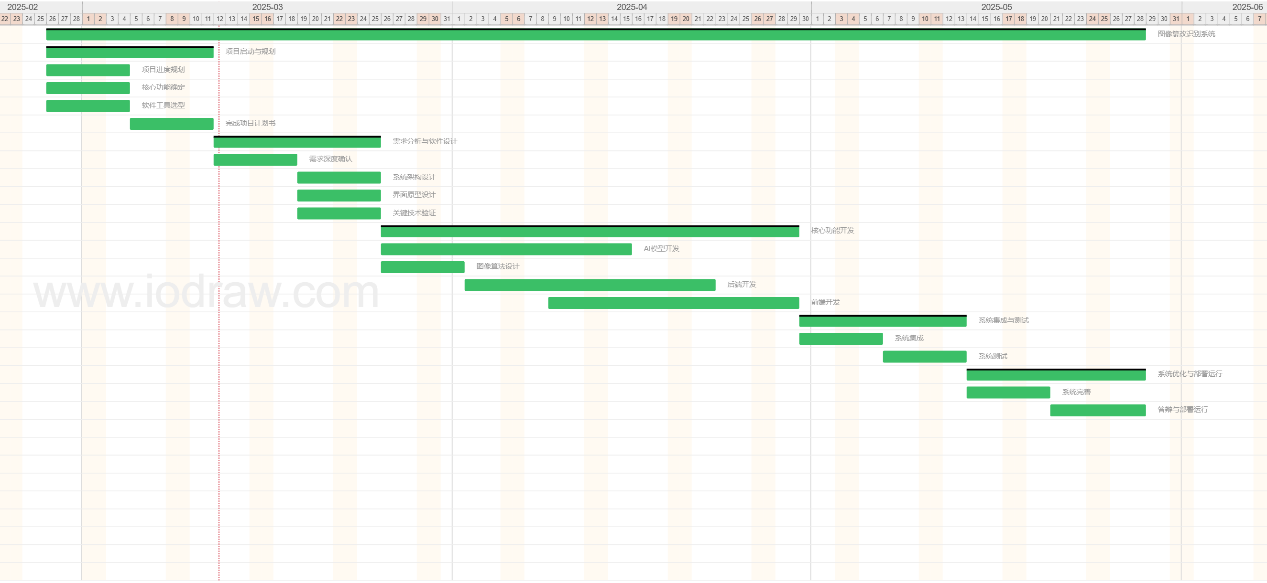
协助工作：参与系统集成、协助性能优化、支持文档编写

# 实施计划

## 风险评估及对策

|  |  |  |  |
| --- | --- | --- | --- |
| **风险类型** | **风险描述** | **影响程度** | **应对措施** |
| 技术实现风险 | 模型性能未达预期 | 高 | 预研阶段完成核心算法验证，保留备用模型方案 |
| 数据质量风险 | 训练数据不足或标注错误导致模型偏差 | 中 | 构建多源数据集，定期进行数据质量审计与清洗 |
| 人力资源风险 | 关键开发人员流动或技能不足 | 中 | 建立代码文档规范与知识共享机制，预留预算用于紧急技术外包支持 |
| 进度延误风险 | 开发任务依赖冲突导致里程碑延期 | 高 | 采用敏捷开发模式，设置总工期的15%为缓冲期 |
| 知识产权风险 | 技术方案涉及第三方专利侵权 | 低 | 核心算法采用创新设计，申请软件著作权进行保护 |

## 总体进度计划



### 第一阶段：项目启动与规划（第1-2周）

* 项目进度规划
* 核心功能确定
* 软件工具选型
* 完成项目计划书

### 第二阶段：需求分析与软件设计（第3-4周）

* 需求深度确认
* 系统架构设计
* 界面原型设计
* 关键技术验证

### 第三阶段：核心功能开发（第5-9周）

1.前端开发

* 用户界面实现
* 图像上传与预览
* 结果展示模块

2.后端开发

* 数据处理服务
* 用户认证系统
* 存储系统实现

3.AI模型开发

* 训练数据集处理
* 深度学习模型训练
* 模型集成与优化

4.图像算法设计

* 图像检测算法实现
* 性能测试与调优

### 第四阶段：系统集成与测试（第10-11周）

1.系统集成（第10周）

* 前后端联调
* 模型服务集成
* 功能测试
* 性能优化

2.系统测试（第11周）

* 系统测试与Bug修复
* 文档编写
* 演示视频制作

### 第五阶段：系统优化与部署运行（第12周-第13周）

1.系统完善（第12周）

* 功能优化
* 性能提升
* 用户体验改进

2.答辩与部署运行（第13周）

* 项目答辩
* 交付使用

## 项目里程碑计划

|  |  |  |
| --- | --- | --- |
| **里程碑** | **交付成果** | **完成时间** |
| 项目规划完成 | 项目计划书 | 第2周 |
| 需求确认 | 需求规格说明书 | 第4周 |
| 模型训练完成 | 准确率≥85%的检测模型 | 第7周 |
| 系统初版上线 | 可运行的检测系统 | 第10周 |
| 项目验收 | 通过测试并交付用户 | 第13周 |

# 预算

## 人员成本

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **角色** | **人数** | **工作量** | **单价(元/人月)** | **小计(元)** | **说明** |
| 项目经理 | 1 | 3个月 | 8,000 | 24,000 | 统筹协调、风险管理 |
| 前端开发工程师 | 1 | 3个月 | 7,000 | 21,000 | Web端/小程序开发 |
| 后端开发工程师 | 1 | 3个月 | 9,000 | 27,000 | API开发与数据库管理 |
| 算法工程师 | 1 | 3个月 | 10,000 | 30,000 | 模型训练与优化 |
| 运维工程师 | 1 | 3个月 | 7,000 | 21,000 | 系统部署与维护 |
| 合计 | 5 | 15人月 |  | 123,000 |  |

## 设备成本

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目** | **规格** | **单价（元）** | **数量** | **小计(元)** | **说明** |
| 云计算资源（GPU） | 阿里云GN6v（8核32G/1\*T4） | 2,500/月 | 3个月 | 7,500 | 模型训练与推理服务 |
| 云存储 | 阿里云OSS（1TB标准存储） | 200/月 | 3个月 | 600 | 图像数据与检测结果存储 |
| 本地测试设备 | NVIDIA RTX 3060工作站 | 8,000 | 1台 | 8,000 | 开发环境搭建与本地调试 |
| 合计 |  | | | 16,100 |  |

## 其他经费预算

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目** | **内容** | **单价（元）** | **数量** | **小计（元）** | **说明** |
| 数据集采购 | 专业伪造图像数据集（DFFD） | 3,000 | 1套 | 3,000 | 补充Deepfake检测数据 |
| 数据标注外包 | 人工标注服务（5,000张图像） | 0.8/张 | 5,000张 | 4,000 | 关键样本精细化标注 |
| 第三方测试工具 | Postman Pro年度订阅 | 1,200 | 1 | 1,200 | API接口自动化测试 |
| 法律咨询 | 数据合规审查服务 | 2,000 | 1次 | 2,000 | 《个人信息保护法》合规性保障 |
| 合计 |  | | | 10,200 |  |

## 总经费预算

总预算：123,000（人员） + 16,100（设备） + 10,200（其他） = 149,300元

# 参考资料

[1]篡改检测技术前沿探索——从基于检测分割到大模型

[EB/OL]．（2024-12-24）［2024-12-26］．<https://blog.csdn.net/air__Heaven/article/details/144678696>

[2]Synthetic Image Verification in the Era of Generative Artificial Intelligence What Works and What Isn’t There yet. IEEE S&P, 2024.

[3] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[4] Tan M, Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.

[5] Wang X, Girshick R, Gupta A, et al. Non-local neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7794-7803.

[6] Woo S, Park J, Lee J Y, et al. CBAM: Convolutional block attention module[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 3-19.

[7] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[8] Kendall A, Gal Y, Cipolla R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7482-7491.

[9] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.

[10] Cubuk E D, Zoph B, Mane D, et al. Autoaugment: Learning augmentation strategies from data[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 113-123.

[11] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.