# 1. 引言

## 1.1. 编写目的

本系统设计文档旨在详细描述眼底影像医学诊断系统的架构、功能模块、接口设计、数据库设计及系统安全策略，作为项目开发和后期维护的重要依据。通过本设计文档，开发人员可以清晰了解系统的整体结构和技术实现细节，运维人员可以根据部署架构和监控策略保障系统稳定运行。

本设计文档将帮助团队在开发过程中保持一致性，减少因理解偏差而导致的开发错误，确保系统具备良好的性能、可扩展性和安全性。

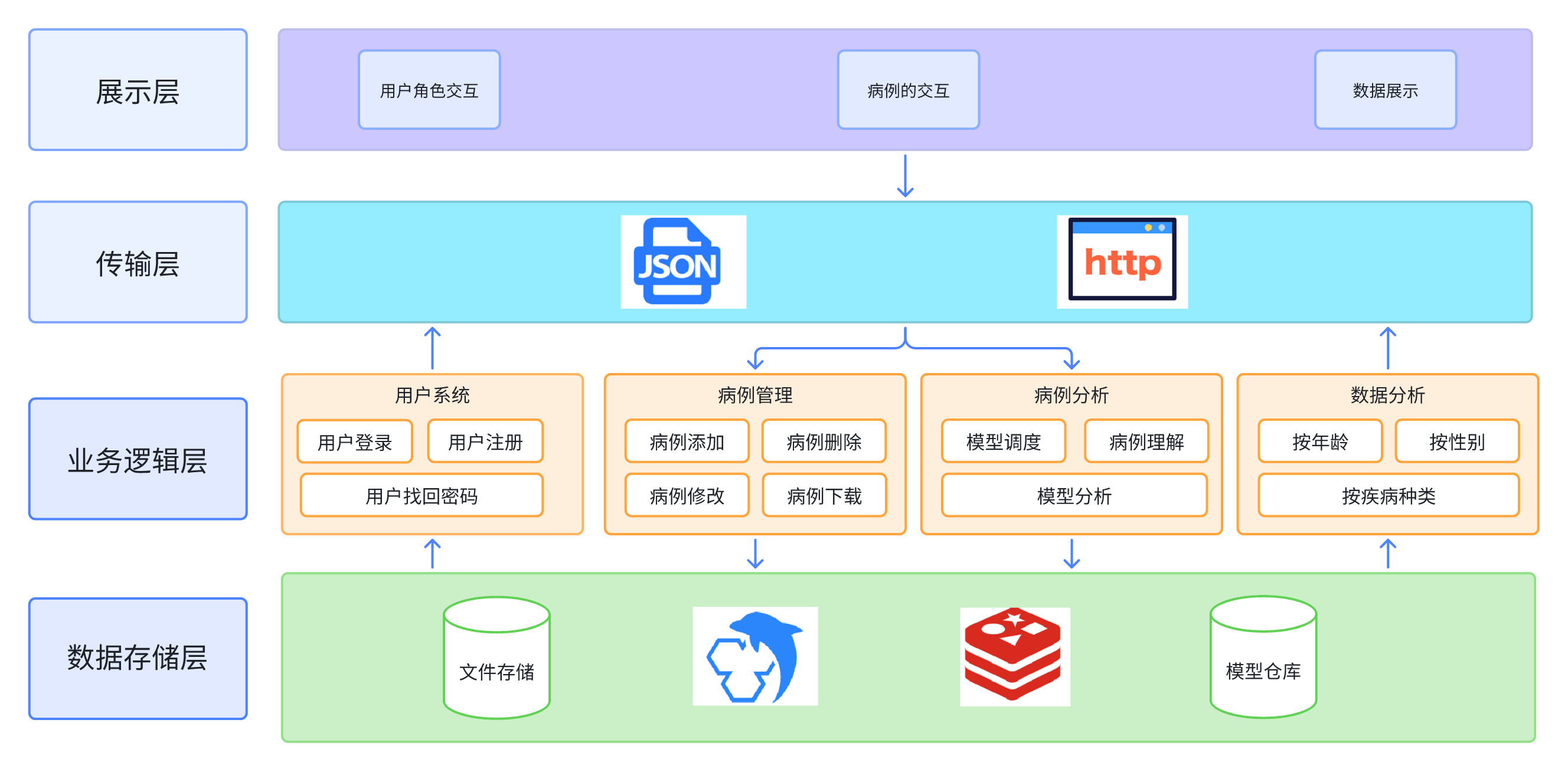
## 1.2. 定义与缩写

|  |  |
| --- | --- |
| 缩写 | 定义 |
| OSS | 阿里云对象存储（Object Storage Service） |
| JWT | JSON Web Token，用于用户身份认证 |
| DR | 糖尿病视网膜病变（Diabetic Retinopathy） |
| MyBatis | Java 持久层框架 |
| AI | 人工智能（Artificial Intelligence） |
| API | 应用程序接口（Application Programming Interface） |

表1：定义与缩写对应表

# 2. 系统总体架构设计

## 2.1. 系统架构图



该系统采用了分层架构设计，分为展示层、传输层、业务逻辑层和数据存储层。每一层的功能和作用如下：

1. **展示层**：

展示层位于系统的最上方，主要负责用户界面的呈现和用户交互。在此层，用户通过用户角色情况与系统进行交互，所有的用户请求和操作都通过此层进行处理。

1. **传输层**：

传输层负责数据的交换与传递。通过 HTTP 协议与 JSON 格式进行数据交换，确保前后端的数据传输和接口交互。该层实现了客户端请求和服务器端响应的数据通信。

1. **业务逻辑层**：

该层实现了系统的核心业务逻辑，如用户注册、登录、密码找回等。该层还包含了病历管理模块，用户可以通过该模块添加、删除、修改病历或下载病历。此外，业务逻辑层还包含病历分析模块，对病历进行进一步的模型调度、理解和分析，确保处理结果精准。

1. **数据存储层**：

数据存储层负责存储系统的所有数据，包括文件存储和数据库存储。该层使用文件存储系统保存病历的相关文件和资料，同时使用数据库（如 MySQL）进行结构化数据的存储。此外，模型库也存储在数据存储层，用于提供影像分析等模型支持。

## 2.2. 系统部署架构

### 2.2.1前端部署

采用Vue 3 + Nginx 高可用架构

**构建与静态资源分发**

通过 npm run build 生成生产环境静态文件（HTML/CSS/JS），输出至 dist 目录。

基于 nginx:1.23-alpine 定制 Docker 镜像，将静态文件部署至容器内 /usr/share/nginx/html 目录。

**配置 Nginx 反向代理规则：**

根路径（/）直接响应静态资源，启用 gzip 压缩与浏览器缓存（Cache-Control: max-age=31536000）。

/api 路径代理至后端服务集群，透传客户端真实 IP（X-Real-IP 头）以支持审计功能。

通过阿里云 SSL 证书服务为域名启用 HTTPS，强制所有流量使用 TLS 1.3 加密。

**性能优化**

使用阿里云 CDN 加速静态资源分发，边缘节点缓存命中率 ≥ 95%。

容器镜像大小优化至 ≤ 50MB（通过多阶段构建移除冗余依赖）。

### 2.2.2后端部署

选择Spring Boot 微服务集群

**容器化与负载均衡**

构建可执行 JAR 包并嵌入 Docker 镜像（基于 eclipse-temurin:17-jdk-alpine），

**通过 Docker Compose 定义服务集群：**

启动 3 个后端实例，分别监听 8080、8081、8082 端口。

配置健康检查端点（/actuator/health），容器自愈时间 ≤ 30 秒。

**Nginx 负载均衡策略：**

采用加权轮询（Weighted Round Robin）分配请求。

失败请求自动重试至备用节点（proxy\_next\_upstream 超时阈值 5 秒）。

**安全与监控**

容器间通信限制于自定义 Docker 网络（med-network），禁止外部直接访问。

集成 Prometheus + Grafana 监控 API QPS、错误率与线程池状态。

### 2.2.3数据库部署

选择MySQL 高可用集群

**容器化与数据持久化**

使用官方 mysql:8.0-debian 镜像，挂载阿里云云盘至容器内 /var/lib/mysql 目录（EXT4 文件系统）。

配置参数：

事务隔离级别：REPEATABLE READ

字符集：utf8mb4 支持 Emoji 及生僻字存储

连接池上限：500（与后端服务线程池匹配）

每日 02:00 执行全量备份（mysqldump + OSS 归档），保留周期 30 天。

灾备设计

主从同步：部署只读副本（Read Replica）实现读写分离。

故障切换：通过 Keepalived 实现 VIP 漂移，切换时间 ≤ 60 秒。

### 2.2.4模型部署

选择Python 推理服务

**独立容器化部署**

基于 tensorflow/tensorflow:2.12.0-gpu 定制镜像，集成 CUDA 11.8 与 cuDNN 8.6 驱动。

暴露 REST API 端点 /v1/predict，输入输出格式遵循 OpenAPI 3.0 规范。

**资源分配策略：**

GPU 显存限制：单容器 ≤ 8GB（通过 --gpus '"device=0"' 指定独占显卡）。

进程级隔离：限制 PyTorch/TensorFlow 线程数，避免 CPU 资源争抢。

**性能与弹性**

预热机制：容器启动时预加载模型至 GPU 显存，降低首次推理延迟。

水平扩展：根据 GPU 利用率（≥80%）自动扩容实例，最大并发处理 50 请求/秒。

### 2.2.5文件存储

选择与阿里云 OSS 深度集成

**存储架构设计**

创建私有存储桶（Bucket），地域与 ECS 实例同区域（如 oss-cn-hangzhou）。

文件分类存储：

目录 数据类型 生命周期策略

/raw\_images 原始眼底影像 永久保留

/reports 诊断报告（PDF） 1年后转低频访问存储

**临时访问安全控制**

后端生成预签名 URL，有效期 10 分钟，限制 IP 范围与 HTTP Referer。

访问日志全量采集至阿里云 SLS，支持异常下载行为分析。

## 2.3. 技术选型

|  |  |  |
| --- | --- | --- |
| 模块 | 技术 | 说明 |
| 前端 | Vue 3 + Axios + Element Plus | 实现用户界面和交互 |
| 后端 | Spring Boot + MyBatis | 实现业务逻辑和数据库交互 |
| 数据库 | MySQL | 存储结构化数据 |
| 文件存储 | 阿里云 OSS | 存储影像和报告 |
| 机器学习 | Python + PyTorch | 图像分类与分析 |
| 身份认证 | JWT | 用户登录和权限控制 |
| 部署 | Docker + Nginx | 容器化部署与负载均衡 |

表2：技术选型表

## 2.4. 接口设计

接口设计是系统设计的重要组成部分，定义了前后端之间、后端与模型之间的数据交互方式。本系统采用RESTful API 设计规范，通过 HTTP 协议进行通信，接口返回统一格式的 JSON 数据。

接口设计遵循以下原则：

RESTful 风格：使用标准的 HTTP 方法（GET、POST、PUT、DELETE）

统一返回格式：通过标准 JSON 结构返回数据，包含状态码、数据和提示信息

状态码规范：严格遵循 HTTP 状态码标准（如 200、400、401、500 等）

参数校验：对参数进行校验，防止越权访问和数据破坏

身份认证：通过 JWT 进行身份认证，保护接口安全

### ****2.4.1. 统一返回格式****

所有接口采用统一的 JSON 格式返回数据，返回格式示例如表一

|  |
| --- |
| {  "code": 200,  "msg": "成功",  "data": {  "key1": "value1",  "key2": "value2"  }  } |

表3：JSON格式实例

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **说明** |
| code | int | 状态码（200 表示成功，4xx 表示客户端错误，5xx 表示服务器错误） |
| msg | string | 提示信息 |
| data | object | 返回的具体数据内容 |

表4：字段含义示意

### 2.4.2. 状态码规范

|  |  |  |
| --- | --- | --- |
| 状态码 | 说明 | 备注 |
| 200 | 成功 | 操作成功 |
| 400 | 请求参数错误 | 请求格式或参数有误 |
| 401 | 未授权 | 用户未登录或权限不足 |
| 403 | 拒绝访问 | 用户无权限访问资源 |
| 404 | 资源不存在 | 访问的资源不存在 |
| 500 | 服务器内部错误 | 服务器异常 |

表5：状态码说明表

### 2.4.3 安全性设计

JWT 鉴权：接口访问需要携带 JWT 令牌  
角色权限控制：根据用户角色限制接口访问权限  
数据完整性校验：后端对传入参数进行完整性校验  
HTTPS 传输：通过 HTTPS 传输，防止数据泄露

## 2.5. 数据库设计

数据库设计是系统设计的重要组成部分，直接影响系统的数据存储、查询效率和扩展能力。本系统采用MySQL 8.0.34 作为关系型数据库，存储病人信息、影像记录、分析结果和诊断报告等结构化数据。同时，影像文件和报告文件采用阿里云 OSS 进行非结构化数据存储。

在数据库设计中，遵循第三范式（3NF） 设计原则，保证数据的完整性、唯一性和可扩展性。同时通过建立适当的索引和外键，优化查询效率，减少数据冗余。

### 2.5.1. 数据库概述

数据库名称：medical\_eye\_system

数据库类型：MySQL 8.0.34

存储引擎：InnoDB

字符集：utf8mb4

排序规则：utf8mb4\_general\_ci

连接池：使用 Spring Boot 自带的 HikariCP 连接池

主键生成策略：自增（AUTO\_INCREMENT）

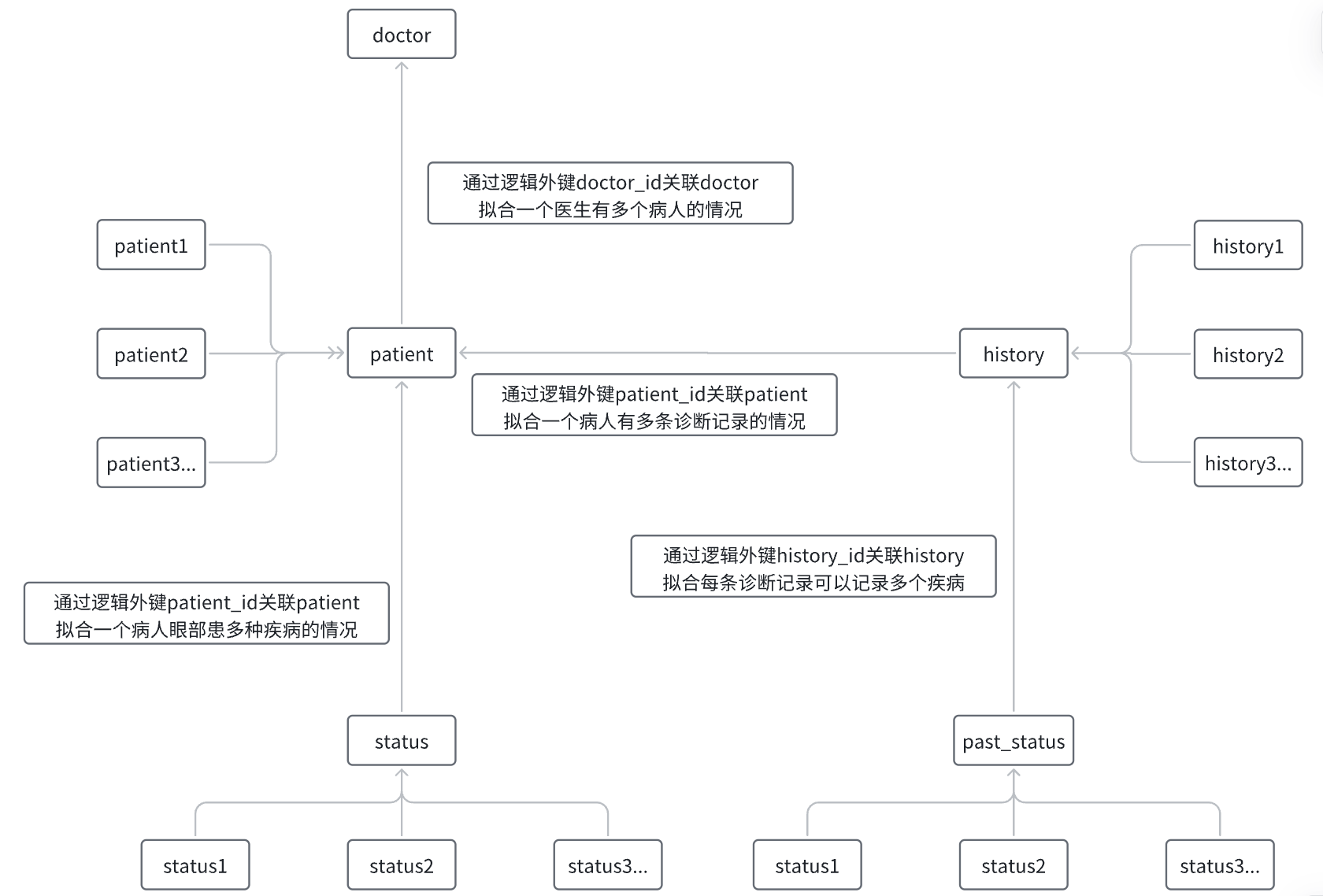
### 2.5.2. 数据库表设计

为满足业务需求，设计以下 5张核心表：

|  |  |  |
| --- | --- | --- |
| 表名 | 描述 | 说明 |
| patient | 病人基本信息表 | 存储病人基本信息 |
| satus | 病人眼部状况表 | 存储病人眼部具体患病情况，逻辑关联patient |
| hstory | 病人诊断历史记录表 | 存储病人的历史诊断记录表 |
| Past\_history | 病人历史眼部疾病状况表 | 存储病人历史眼部具体患病情况，逻辑关联history |
| dctor | 用户/医生基本信息表 | 存储医生登录信息和权限 以及医生基本信息 |

表6：数据库表设计

### 2.5.3. 表关系图



## 2.6 异常处理设计

在系统运行过程中，可能会出现网络异常、数据库异常、程序异常等情况。为保证系统的稳定性和用户体验，系统在设计中引入了完整的异常捕获和处理机制。

### 2.6.1. 异常捕获机制

* 采用 Spring Boot 的 全局异常处理机制（@ControllerAdvice + @ExceptionHandler）进行异常捕获。
* 将异常按类别分为业务异常和系统异常，分别进行不同的处理。

### 2.6.2. 异常处理返回规范

所有异常统一返回 JSON 格式，示例如下表。

|  |
| --- |
| {  "code": 500,  "message": "服务器内部错误",  "data": null  } |

表7：异常返回实例表

|  |  |  |
| --- | --- | --- |
| 状态码 | 含义 | 说明 |
| 400 | 参数错误 | 请求参数不合法 |
| 401 | 未授权 | 用户未登录或登录已过期 |
| 403 | 禁止访问 | 用户无权限访问接口 |
| 404 | 资源不存在 | 请求的接口或资源不存在 |
| 500 | 服务器错误 | 系统内部异常 |

表8：状态码含义表

### 2.6.3. 日志记录

记录异常日志，包含：请求路径、请求参数、异常类型、堆栈信息。

异常日志存储在ELK（Elasticsearch + Logstash + Kibana）系统中，方便检索和分析。

## 2.7 功能设计

根据前文《软件需求分析文档》的描述，团队决定将本系统实现为针对医生等医护人员的，能提供眼底影像诊断相关建议的，能提供一体化病例管理平台的，能便捷展示相关数据的综合系统。对于本系统，需要实现如下的多种功能。

1. 基础的用户登录、注册功能。为了保证数据不被窃取，在访问数据的时候需要首先进行登录，确认身份之后才允许访问。
2. 病例的相关管理功能。管理功能需要实现病例的添加、分析、修改、删除等功能，并且提供某一位病人的历史病例的查询。这些功能旨在为医生提供更易用、更实在的管理功能。
3. 病例的分析功能。在添加完病例之后，需要对病例进行分析。对病例进行分析之后，给出对应的分类结果和诊断建议，缩短医生的诊断流程，释放巨量病例带来的诊断压力。
4. 数据展示功能。本系统支持对病例的相关数据分门别类地进行展示。在宏观上能够比较好地获取核心信息，有助于整体的方向把握。

# 3 详细设计

## 3.1前端详细设计

### 3.1.1 前端架构概述

前端采用 Vue 3 作为主要框架，搭建动态界面整体架构，结合 Vite 进行构建，利用Axios 进行数据异步请求，满足程序多线程，高并发的需求。使用Element Plus 作为 UI 组件库，确保界面美观、交互流畅，并提高开发效率。利用Echarts图形库，将数据可视化为简介了然的图像，大大提高了与用户的交互能力。

### 3.1.2 主要功能模块

#### 3.1.2.1 登录与权限管理

1. 使用JWT进行身份验证。JWT 是一个开放标准（RFC 7519），用于在网络间安全传输 JSON 格式的声明（Claims）。它由三部分组成：

* Header：描述算法（如 HS256、RS256）和 Token 类型（JWT）。
* **Payload**：包含用户身份信息（如用户 ID、角色）和其他声明（如过期时间 exp）。
* **Signature**：对 Header 和 Payload 的签名，确保 Token 未被篡改。

前端发送用户凭证（用户名/密码）到后端api，后端验证成功后，返回JWT（即token）于响应体中，前端通过fetch捕获token，综合分析使用情景来决定储存方式，在敏感场景通过使用HttpOnly和Sucure标志防止XSS和中间人攻击，并设置CSRF防护，而当前端需要读取token具体内容时，则结合LocalStorage和Cookie技术，综合对token进行安全储存。并且在每次用户请求受保护的API时，通过在请求头中附加JWT来进行身份验证和权限管理。依赖后端返回401状态码而非解码token的方式来检查用户凭证的过期时间，并且引入Refresh Token机制来刷新token或跳转至登录界面，有效保障身份认证的安全性，同时保持用户体验流畅。

1. 使用Vue Router进行路由控制。路由控制模块基于 **Vue Router** 实现，主要解决以下问题：

* **用户导航权限管理**：限制未授权用户访问敏感路由。
* **动态组件加载**：根据用户角色或权限动态渲染页面组件。
* **单页应用（SPA）体验优化**：通过路由懒加载、异步导航守卫等机制提升流畅性。

路由守卫(Navigation Guards）是权限控制的核心拦截器，分为以下三类：

|  |  |  |
| --- | --- | --- |
| 守卫类型 | 触发时机 | 应用场景 |
| 全局前置守卫 | 所有路由跳转前触发 | 全局登录状态校验，权限拦截 |
| 路由独享守卫 | 进入特定路由前触发 | 精细化路由权限控制（管理员） |
| 组件内守卫 | 组件加载前/参数变化时/离开前触发 | 数据保存显示，动态参数校验 |

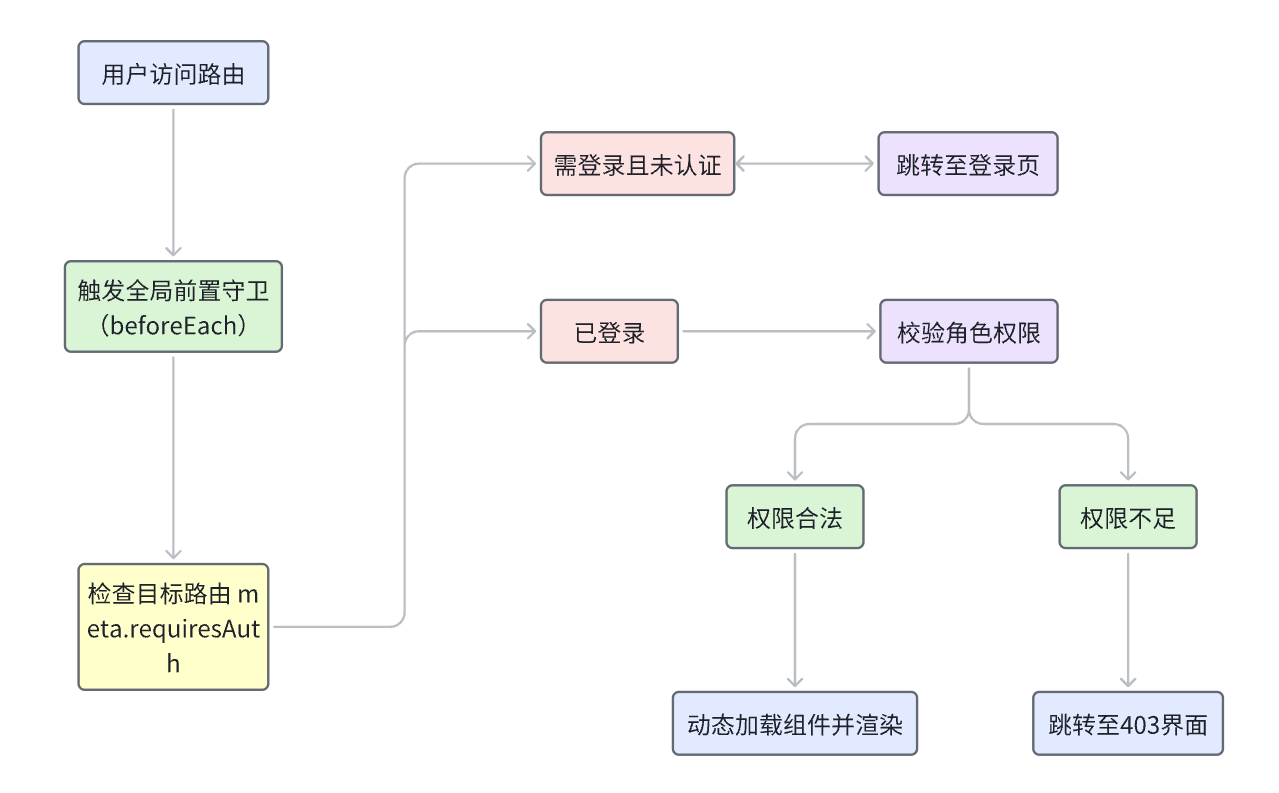
利用动态路由，扩展代码的灵活性，其主要实现逻辑如下:

* **静态路由定义：**公共路由（如登录页、404页）无需权限即可访问。
* **权限数据获取：**用户登录后，向后端请求角色或权限列表。
* **路由动态注册**：基于权限数据，通过router.addRoute()动态添加私有路由。

通过meta字段扩展路由配置，传递权限控制参数，实现路由元信息化。并且利用路由懒加载技术，优化首屏加载速度，按需加载路由组件。通过前后端权限双重校验，前端仅控制路由跳转，敏感接口仍需后端独立验证用户权限，并且设置方法越权访问，动态路由数据均实现了签名和加密算法，避免用户篡改本地权限数据，错误处理方面捕获导航异常并记录日志，定义全局404界面，三重保障确保数据与程序安全性。

导航守卫中集成了加载状态提示，如Nprogress进度条等，并且使用<keep-alive>缓存高频访问的页面组件，优化加载速度，给用户更良好的使用体验。

路由控制的流程如下图所示：



1. 采用 Vuex 和 Pinia 管理用户状态

用户状态管理模块通过集中化存储和管理用户身份、权限及会话信息，确保以下能力：

* **全局状态一致性**：跨组件共享用户登录状态、角色权限等关键数据。
* **响应式更新**：状态变化自动触发依赖组件的视图更新。
* **持久化支持**：结合浏览器存储（如 localStorage）保障页面刷新后状态不丢失。
* **安全隔离**：通过标准化流程修改敏感状态（如 Token），避免直接篡改。

状态管理库选型如下：

|  |  |  |
| --- | --- | --- |
| 方案 | 适用场景 | 核心优势 |
| **Vuex** | 复杂状态管理 | 官方维护、严格的单向数据流、完善的调试工具 |
| **Pinia** | 轻量级和TypeScript | 更简洁的 API、Composition 风格、类型友好 |

根据实际场景交叉使用多种方案确保状态持久化实现，方案对比与配置如下：

|  |  |  |
| --- | --- | --- |
| 方案 | 优点 | 缺点 |
| **localStorage** | 简单易用、长期存储 | 受 XSS 攻击风险 |
| **SessionStorage** | 页面关闭后自动清除 | 作用域仅限当前标签页 |
| **Cookie** | 支持服务端读取、自动过期 | 存储容量受限 |

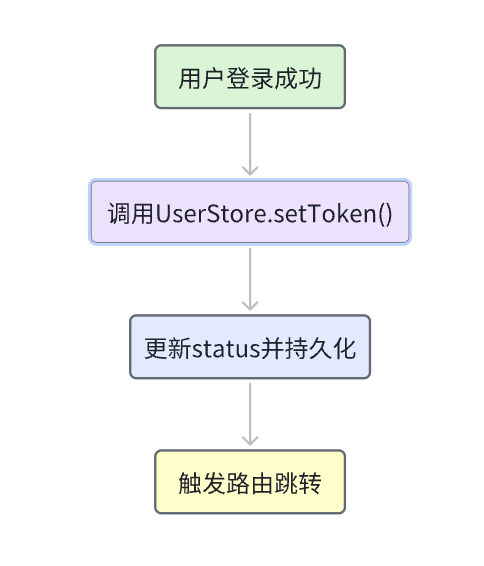
例如pinia+localstorage持久化的部分代码如下：

|  |
| --- |
| // stores/user.js  import { defineStore } from 'pinia';  export const useUserStore = defineStore('user', {  state: () => ({  token: localStorage.getItem('jwt') || null,  userInfo: null  }),  actions: {  setToken(token) {  this.token = token;  localStorage.setItem('jwt', token);  },  clear() {  this.token = null;  this.userInfo = null;  localStorage.removeItem('jwt');  }  }  }); |

引入安全防护机制如下

* **敏感操作隔离**：通过 actions 方法统一修改 Token 或用户信息，禁止直接修改 state。
* **XSS 防御**：
  + 避免在state中存储原始敏感数据（如密码）。
  + 使用 HttpOnly Cookie 存储 Token 时，需配合 CSRF Token 防护。
* **沙箱化处理**：对从持久化存储（如 localStorage）中读取的数据进行合法性校验。

确保登录状态同步的逻辑如下：



用户状态管理模块通过 **集中化存储**、**响应式更新** 和 **安全防护** 机制，保障前端应用的核心状态安全可靠。关键设计原则包括：

1. **单向数据流**：通过 actions 控制状态修改，避免不可预测的变更。
2. **分层解耦**：状态管理与业务逻辑、UI 组件分离，提升可维护性。
3. **防御式编程**：对持久化数据校验、敏感操作隔离，增强系统健壮性。

#### 眼底影像上传与管理

1. 通过 Element Plus 的 Upload 组件实现图片上传。

在前端实现眼底影像的上传与管理功能时，我们采用了 Element Plus 提供的 el-upload 组件，该组件支持 拖拽上传 和 批量上传，使用户可以更加便捷地提交医疗影像数据。在用户交互方面，该组件提供了直观的进度条显示，确保用户可以清晰地了解上传状态。此外，系统内置了文件类型、大小和完整性校验，以防止用户上传不符合要求的文件。例如，前端会限制上传的影像格式，仅允许 JPG、PNG 等高质量图片，并设定 最大文件大小 以优化系统存储和带宽占用。当用户选择影像文件后，前端会对文件进行 格式验证、大小检查，并提供 实时错误反馈。如果文件不符合要求，例如格式不支持或文件大小超过限制，系统会及时提示用户调整文件后重新上传。对于符合标准的影像，el-upload 组件会自动调用后端 API，将影像数据以 流式方式上传至阿里云 OSS。在影像上传过程中，系统会实时显示进度条，确保用户可以跟踪上传进度，提升交互体验。

为了保证影像数据的完整性和合规性，前端在上传之前会对文件进行严格校验，包括但不限于 文件格式、大小限制、文件内容完整性 等。例如，仅允许上传符合医疗影像格式的 JPG、PNG 等高质量图片文件，并限制最大上传大小，避免超大文件影响上传速度和存储效率。对于不符合要求的影像文件，系统会提示用户重新选择或压缩后再上传，以确保存储系统中的影像符合医疗影像标准。

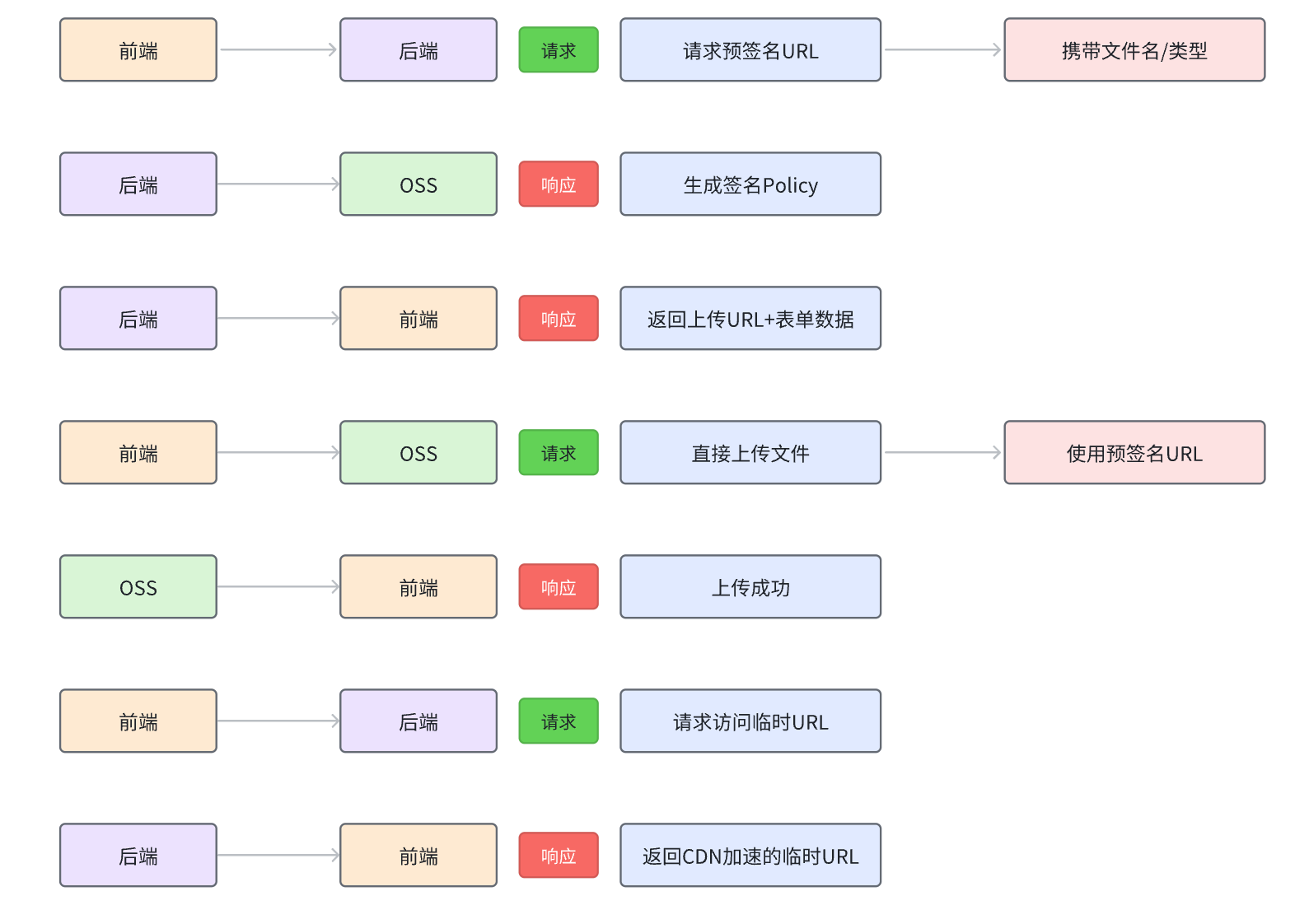
1. 影像存储至阿里云 OSS，前端通过 API 获取临时访问 URL。

上传的影像文件将直接传输至阿里云 OSS，前端通过 Axios 向后端请求临时访问 URL。采用预签名 URL 方式，使用户能够安全地访问影像数据，而无需暴露存储凭证。同时。在影像上传后，前端将调用后端 API，存储影像的元数据（如影像名称、上传时间、用户信息等）。采用 CDN 加速影像加载，提升访问速度。前端对影像进行缩略图预览，避免大文件加载带来的性能问题。

1. 文件校验与OSS直传流程

* **格式限制：**通过 accept 属性和 beforeUpload 双重校验
* **大小限制**：在 beforeUpload 中检查 file.size
* **实时反馈**：使用 ElMessage 组件提示错误

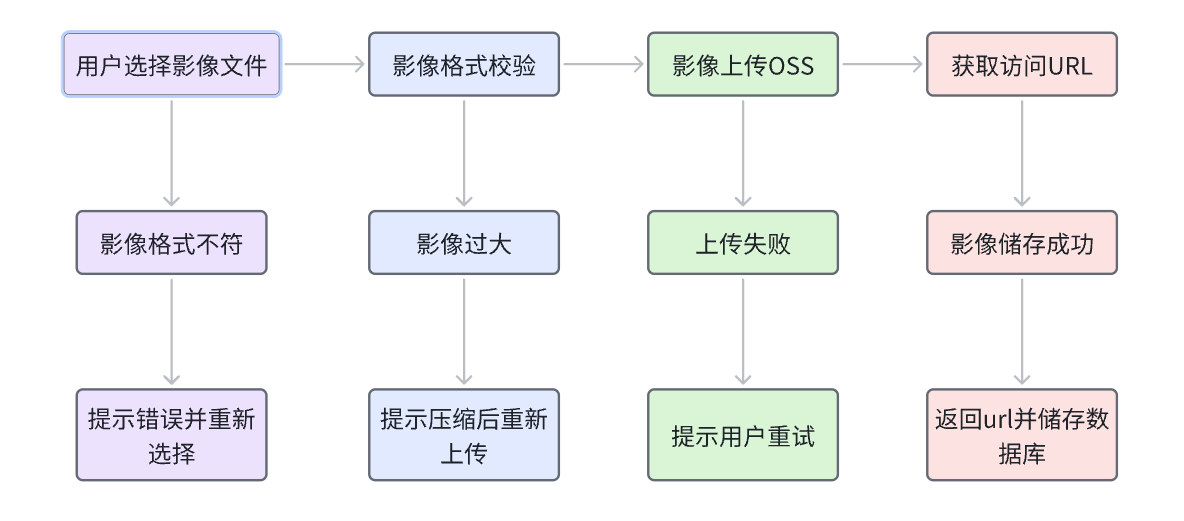
OSS直传流程图如下：



1. 影像存储表结构如下：

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| **id** | Int | |  | | --- | |  |   影像唯一id |
| **filename** | Varchar | 影像文件名 |
| **url** | Text | 影像访问 URL |
| **upload\_time** | Timestamp | 上传时间 |
| **user\_id** | int | 上传用户 ID |

影像上传流程如下：



1. 使用 Web Worker 进行前端影像压缩，减少上传流量，提高响应速度。

* **主线程职责**：负责 DOM 渲染、事件响应（点击、滚动等）、动画执行等核心任务。
* **问题场景**：直接在前端主线程执行 **大文件压缩、复杂计算（如加密/解密）** 等耗时操作时，会导致页面卡顿甚至无响应。
* **Web Worker 方案**：将这些任务交给独立的 Worker 线程处理，主线程保持流畅。

眼底影像文件通常较大（如 10MB+），直接上传耗时长、浪费带宽。使用 Web Worker 在后台线程中压缩图片（如转为 JPEG 并降低质量）。压缩后的文件体积减小，提升上传速度。用户体验提升：压缩期间用户仍可正常操作页面。在上传前校验文件完整性、生成缩略图或提取元数据。避免主线程卡顿，同时并行处理多个文件。具体代码实现分为以下两步：

第一步创建Worker文件

|  |
| --- |
| // public/compress.worker.js  self.addEventListener('message', async (e) => {  const { file, quality } = e.data;  // 压缩逻辑（使用 OffscreenCanvas）  const bitmap = await createImageBitmap(file);  const canvas = new OffscreenCanvas(bitmap.width, bitmap.height);  const ctx = canvas.getContext('2d');  ctx.drawImage(bitmap, 0, 0);  const blob = await canvas.convertToBlob({ type: 'image/jpeg', quality });  self.postMessage(blob);  }); |

第二步在组件中调用

|  |
| --- |
| <script setup>  import { ref } from 'vue';  const compressImage = (file) => {  return new Promise((resolve) => {  const worker = new Worker('/compress.worker.js');  worker.postMessage({ file, quality: 0.7 });  worker.onmessage = (e) => {  resolve(new File([e.data], file.name, { type: 'image/jpeg' }));  worker.terminate(); // 销毁 Worker 释放内存  };  });  };  </script> |

#### 眼底影像分析结果显示

1. 通过 ECharts 可视化分析结果。

影像分析结果采用 ECharts 进行可视化展示，通过病人不同时期眼部指标的折线图清晰反映病人眼部病况的近期变化情况，以柱状图显示出性别，年龄等因素对黄斑，视盘，眼底血管等不同指标的综合影响，用饼状图来综合显示不同人群患不同眼科疾病的比例，辅助医师进行患者诊断的同时也可以对人群预防提供参考。

1. 实现动态数据渲染。

通过 Vue 3 的响应式特性，实现影像分析数据的动态渲染，确保在数据变化时页面可以实时更新，提高用户体验。通过Preloader组件，用于在页面加载时显示一个加载指示器，直到所有数据都已成功加载并准备好呈现给用户。实现了显示加载动画或指示器，控制加载状态的显示与隐藏，可以在数据加载完成后通过状态或事件通知父组件隐藏。

通过Main Wrapper组件，提供应用程序的根容器，包含并布局所有其他组件。提供整个应用的布局和样式，管理内部组件的渲染，包含全局的CSS样式或变量。

1. 交互分析功能。

用户可以选择不同的时间段查看病情发展趋势，通过图例筛选特定病变区域的数据，提供放大、缩小、切换不同分析视图的交互功能。直观感受眼部各个指标的详细信息与正常指标范围的对比，从而辅助医师诊断。

#### 报告管理

1. 生成并下载 PDF 格式的病人报告
2. 实现批量管理、搜索、过滤功能

## 3.2后端详细设计

## 3.3机器学习详细设计

### 3.3.1图像数据预处理

#### 3.3.1.1数据清理与数据分析

针对诊断关键词为：图像质量差、镜头污点、视盘不可见的图像我们进行了数据删除。其对应的英文诊断关键词为："refractive media opacity"、"lens dust/normal fundus" 或 "normal fundus/lens dust"、"epiretinal membrane"。考虑到这样的数据共有(264张，占比近5%)，所以我们只删除左右两眼均出现上述诊断关键词且问题十分明显的图像（共54张，近1%），最后剩下5826张。接着我们对数据分析，做出图1。

由图1，不难看出以下数据分布的几个关键点：

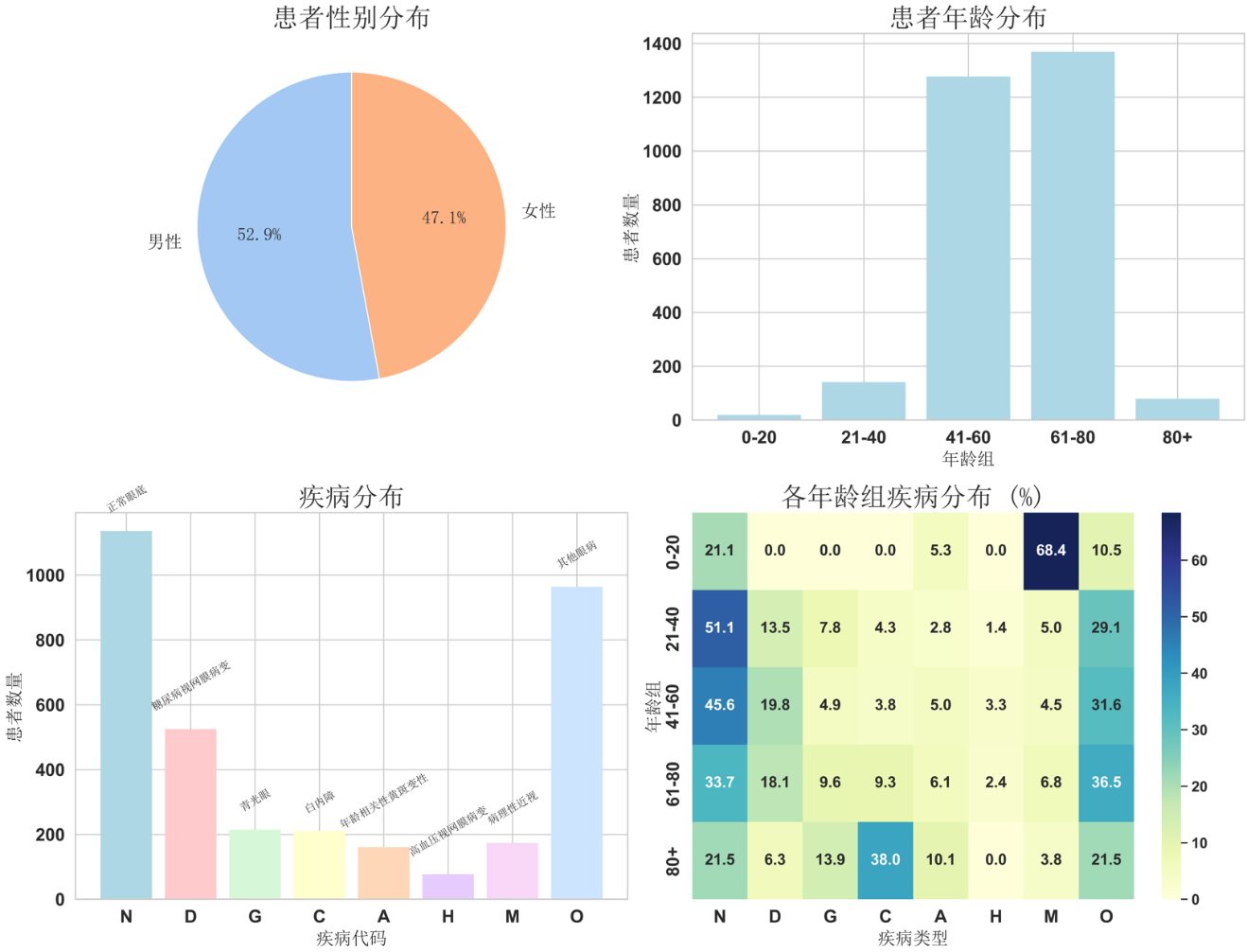
1. 患者以中、老年人为主，根据计算，年龄中位数为58岁，平均数为61岁。
2. 样本中正常样本和其他疾病样本过多，出现了样本分布不均衡的情况。
3. 部分疾病与年龄具有相关性，如病理性近视在0-20岁年龄组的患者比例就高达68.4%。
4. 样本中性别近似均衡，男性略多余女性。

图1：数据分布图

#### 3.3.1.2图像增强

针对眼球在图片中位置参差不齐，可能会影响模型学习照片的空间位置的问题，我们采取可裁剪操作，对原始图像进行第一步处理。具体操作时，分析图片中黑色像素部分，保留最大的不含黑色像素的彩色图像部分，如图2 Figure 2所示。

针对图像亮度不均衡的问题，我们采用自适应直方图均衡化(CLAHE)，增强彩色眼底图像的对比度和亮度。[3]CLAHE采用局部区域自适应增强的工作方式，与全局对比度拉伸方法相比，可以更好地保留图像的局部细节和纹理信息。此外还具有限制对比度放大的优点，便于抑制噪声放大和过度增强的问题。具体操作方法是，调用OpenCV库中的createCLAHE函数，超参数clipLimit设置为2.0，tileGridSize设置为(8,8)，以平衡增强效果与噪声抑制，效果如图2 Figure 3所示

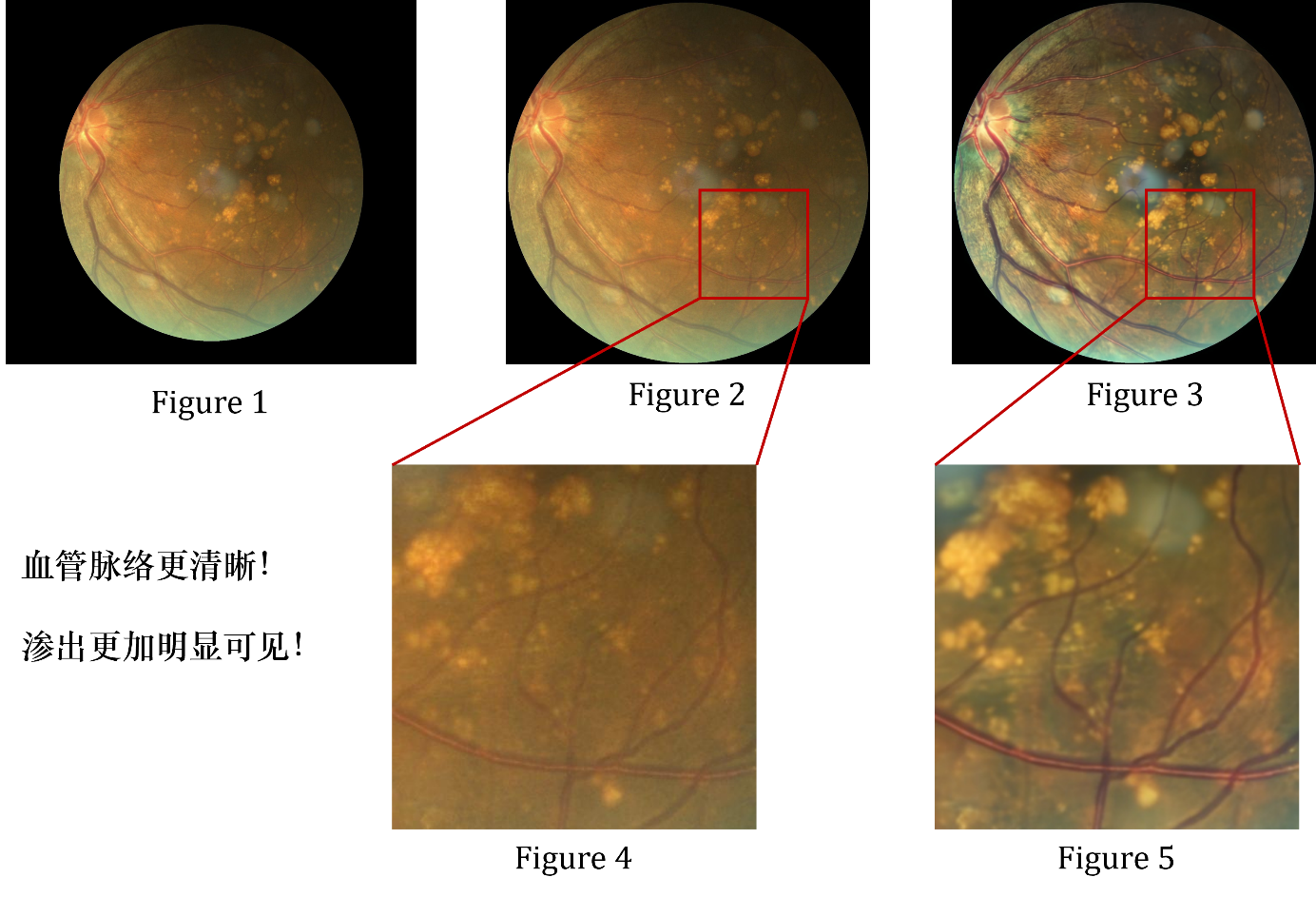
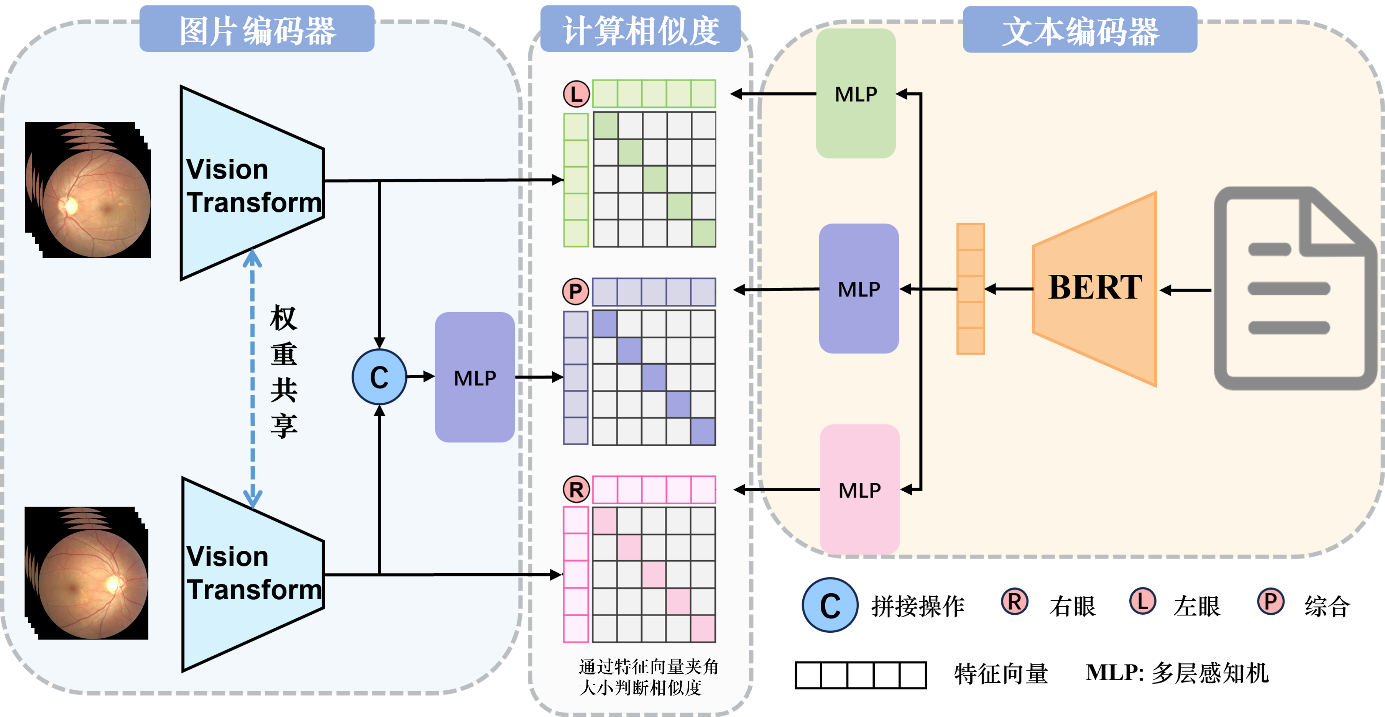
****针对图像经过均衡后，还存在噪声，所以我们采用BM3D混合去噪对图像进行操作。[4]其基本原理是利用图像中相似块的非局部自相似性，通过协同滤波和小波变换实现噪声分离，具有保留图像边缘和纹理细节的同时有效去除高斯噪声的效果。BM3D算法首先对图像进行分块，然后搜索相似块形成三维数组，接着在变换域进行滤波，最后通过聚合操作重建去噪图像，这种方法特别适合处理医学图像中的复杂结构和细节。

图2：图像增强效果图

### 3.3.2模型概述与简介

#### 3.3.2.1 RET-CLIP多模态大模型

****CLIP (Contrastive Language-Image Pre-training) 是由OpenAI研究团队于2021年初提出的一种革命性多模态预训练模型，它通过创新的对比学习方法建立了图像和文本之间的语义联系。CLIP的出现标志着视觉-语言预训练领域的重要里程碑，为多模态理解和跨模态任务提供了强大的基础框架。[1]这里用到的Ret-CLIP是CLIP在医学影像应用上的运用。[2]

**RET-CLIP采用了双编码器架构**，包含一个图像编码器和一个文本编码器。图像编码器是Vision Transformer (ViT),而文本编码器则是基于Transformer架构的BERT。这两个编码器分别将图像和文本映射到共享的高维特征空间，使得语义相关的图文对在该空间中距离较近，而不相关的图文对距离较远。

**图片编码器** 采用双通道设计，专门处理左右眼底图像的视觉特征提取。该模块以两个平行的Vision Transformer(ViT)作为骨干网络，这一选择充分利用了Transformer在捕获长距离依赖关系方面的优势，特别适合处理眼底图像中的复杂病理特征。值得注意的是，两个ViT编码器之间实现了权重共享机制，这一设计有三重意义：首先，减少了模型参数数量，降低了过拟合风险；其次，确保左右眼特征提取过程的一致性，增强了模型的生物学合理性；最后，通过参数共享实现了更高效的学习过程。

视觉特征提取后，系统通过拼接操作将左右眼的特征向量进行整合。这种拼接策略保留了双眼各自的特征信息，同时建立了它们之间的关联性。拼接后的特征经过专门设计的多层感知机(MLP)进行非线性变换，将高维特征映射到与文本特征兼容的语义空间，为后续的跨模态相似度计算奠定基础。[1][2]

**文本编码器** 以BERT模型为核心，负责处理与眼底图像相关的临床文本描述。BERT作为预训练语言模型的代表，能够有效捕获文本中的语义信息和上下文关系，特别适合处理医学专业术语和复杂的临床描述。文本经过BERT编码后，通过三个功能相似的多层感知机(MLP)处理分支，分别生成针对左眼、右眼和综合双眼的文本特征表示。

这种三路并行的设计体现了模型对不同临床场景的适应能力：有时诊断可能只关注单侧眼部情况，有时则需要综合考虑双眼状态。三个分支的MLP通过不同的参数优化，能够提取最适合各自任务的文本特征，增强了模型的灵活性和精确性。

**相似度计算模块** 是实现视觉-文本跨模态匹配的核心环节。图中展示了三种相似度矩阵，分别标记为"L"（左眼）、"P"（综合）和"R"（右眼）。每个矩阵表示相应视觉特征与文本特征的语义匹配程度，通过计算特征向量间的余弦相似度实现。[1]

这种多视角的相似度计算策略使模型能够从不同维度评估视觉-文本匹配关系，既可以单独分析每只眼睛的病理特征与临床描述的一致性，也可以综合考量双眼状态与整体诊断的关联性。这对于检测视网膜病变、青光眼等可能表现为双眼不对称的眼科疾病具有特殊价值。[2]

**微调数据来源** RET-CLIP模型构建了患者级别的视网膜眼底双目图像-文本三元组数据集（RET-Clinical）。该数据集包含来自中国北京同仁医院的193,865例样本，每位患者的三联数据由左右眼的彩色眼底照片（CFP）及对应的临床诊断报告组成[2]

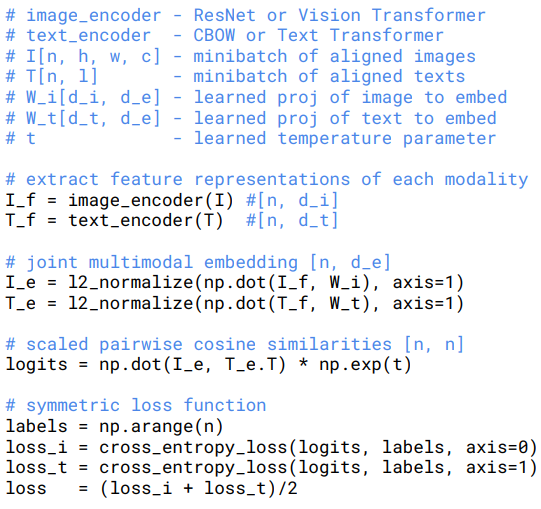
 给出具体的核心算法示例：

图3：预训练核心算法示例

#### 3.3.2.2基于RET-CLIP的微调模型

图4：微调模型框架

基于预训练好的模型\*，针对多标签分类我们进行了微调。模型框架如图4所示。

假设有N个样本的批次， 其中分别为第i个样本的左右眼图像、及元数据。给定的标签可以表示为{。

经过基于Vision Transformer的双分支融合图片编码器，将编码为。

元数据经过由通道注意力块与全连接层组成的多层感知机的编码后变为。

给定的标签经过基于BERT的文字编码器编码为

同时，我们注意到数据集中给出了患者的年龄和性别，因此在常见的模型基础上

**创新性**地增加了一个元数据处理模块

映射到同一特征空间的特征向量计算余弦相似度，若大于阈值(实验中设为0.72)则认为该患者患有该疾病。

最后根据结果与实际的多热标签

(Multi-hot label)进行比较，计算损失函数。

#### 3.3.2.3模型与训练过程具体细节

**文本编码器** 采用的RoBERTa-wwm-ext-base-chinese，它的权重初始化为经过2亿图像文本对训练的中文版CLIP。该编码器采用12层Transformer结构，隐藏层维度为768，注意力头数为12，参数量约为110M。我们保留了预训练模型的字符级tokenizer，最大序列长度设为256。为防止过拟合，我们在Transformer层中应用了0.1的Dropout率。

**图像编码器** 采用的是ViT-B-16，它也是经过2亿图像文本对训练的中文版CLIP。该编码器将输入图像分割为16×16的patch，通过12层Transformer进行编码，隐藏层维度为768，注意力头数为12，总参数量约为86M。为保持与CLIP预训练一致，输入图像尺寸统一调整为224×224像素。图像增强技术包括随机水平翻转、色彩抖动和随机擦除，以提高模型泛化能力。

**元数据编码器** 采用的是两个全连接层组成的多层感知机，Dropout率均采用为0.3。第一层将输入特征映射到72维隐藏空间，采用GELU激活函数；第二层将隐藏空间特征映射到36维表示空间。元数据包括性别年龄，这些特征经过归一化和独热编码后送入编码器。为处理缺失值，我们采用均值代替，性别默认为男。

**融合层** 采用单层全连接层辅以批量归一化层，Dropout率采用为0.5。该层接收文本、图像和元数据编码器的输出，进行多模态信息融合。

**激活函数** 采用从ReLU函数改进的GELU函数，相比ReLU函数，它的导数更加平滑，不容易出现梯度消失的情况.[5]

**模型初始化** 对于图片和文本编码器，我们初始化为RET-CLIP预训练好的模型。元数据编码器、融合器中涉及到的全连接层均采用正态分布初始化。

**损失函数** 采用Focalloss与BCEloss的混合损失函数[6]，具体为

其中的weight是根据每个类别的数量计算得到的权重，旨在减弱分类不均衡的影响

**优化器与学习率** 采用AdamW优化器，权重衰减为0.065，梯度裁剪阈值设置为1.0.仅解冻视觉编码器的最后几层（默认4层）Transformer块，以保留预训练知识的同时允许模型适应眼底图像领域。对于不同的部分采用不同的学习率：

视觉骨干网络：2.4e-5 分类头：7e-4 元数据编码器：5e-4 前75步采用线性预热策略，而后使用余弦退火调度。

批量大小设置为16，训练轮数设置为10轮，随机种子设置为42，精度采用混合精度训练(AMP),硬件环境为Nvidia GEFORCE RTX 4090,采用python 3.10 CUDA 12.6 版本。

#### 3.3.2.4模型效果

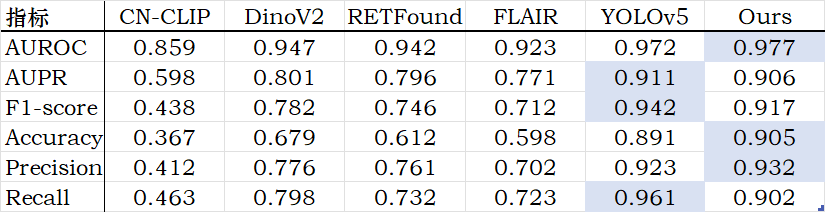
近年来眼底彩色影像分析是眼科疾病辅助诊断的重要领域，近年来基于深度学习的方法取得了显著进展。目前研究主要分为三大方向：

自监督学习基础模型方面，RETFound 通过自监督方式构建眼底图像基础模型，但仅利用视觉信息进行重建学习。[7]DINOv2 虽为通用视觉模型，但显示出在眼底图像上的迁移能力。

视觉-语言多模态融合是近期研究热点。FLAIR 尝试将文本监督引入眼底图像分析，但仅使用简单标签。PMC-CLIP利用生物医学文献图文对构建模型，但非眼科专用。

疾病特定方法针对具体眼病开发算法。Gulsha在糖尿病视网膜病变检测方面达到专科医生水平。IDRID挑战赛(2020)提供了标准化数据集。青光眼检测领域，Ahn开发了早晚期青光眼识别模型。

基于此我们选择了CN-CLIP、DinoV2、RETFound、FLAIR、YOLOv5作为我们对比的模型，值得注意的是，每一个模型的最优超参数是不尽相同的，为了公平起见，我们对每个模型都进行了不少于10+次的参数调优过程，取最佳的模型进行对比。得到以下图像：



先解释一下指标，

AUROC衡量正负例区分能力，通过宏平均（标签独立计算）或微平均（全局合并）评估，适用于类别均衡场景。AUPR关注类别不平衡下的精确率-召回率关系，对正例极少的标签更敏感。F1-Score为精确率和召回率的调和平均，平衡假正例（FP）和假负例（FN）的影响，通过宏/微平均综合评估多标签性能。Accuracy为样本级别准确率。Precision侧重减少FP适用于需高预测置信度的场景；Recall侧重减少FN用于确保正例覆盖率。

从图表中不难看出YOLOv5和我们构建的模型在性能上一骑绝尘。值得一提的是YOLOv5是我们项目前期探索阶段采取的一个模型，他在青光眼、白内障、病理性近视方面表现卓越，准确度能达到97%以上。但针对正常类型的样本准确度只有75%上下，显然不是应用的最佳选择。反观我们在多模态大模型上微调得到的模型，其针对特定样本的准确率不及YOLOv5，最高的也只有95%的准确度，但在各个类别上表现均衡，最差的其他类型疾病也有82%。其次他相比YOLOv5具有更加广阔的应用空间，详见第4部分。

### 3.3.3超参数调优过程

#### 3.3.3.1确定参数范围

共有如下超参数（按优先级排序）：

1. 微调模型解冻层数
2. 学习率以及预热步数和学习率调度器
3. 多标签分类中视为患病的阈值
4. 损失函数的选择以及组合比例系数
5. 权重衰减系数和Dropout率、批量大小

我们运用贝叶斯优化前需要确定每一个超参数的大致范围，因此在根据经验，首先确定了一个超参数的baseline:

解冻层数；4 学习率：分类头 1e-4 骨干网络 1e-5 融合层 5e-4 预热步数采用为50

Focal loss和 BCE loss比例为1:1衰减设为0.05 Dropout设为0.3 批量大小为8

综合考虑到，学习率的范围较大，待调超参数过多，所以在我们人工调参的过程中选择暂时跳过。进而需要我们手工调参确定范围的有：1.微调模型解冻层数 2.分类中的阈值 3.损失函数的组合比例。

考虑到对某一超参数的最佳效果所对应的其他超参数不尽相同，所以对于这一超参数在进行3次实验取最佳效果。

因此基于现有结果，我们确定了超参数的范围

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 解冻层数 (unfreeze\_layers) | 预热步数 (warmup\_steps) | 训练轮数 (epochs) | 批量大小 (batch\_size) | 权重衰减 (weight\_decay) |
| 2 - 5 | 50 - 80 | 4 - 7 | 8,16，32 | 0.04 - 0.08 |

至于学习率的超参数这根据经验设置范围

#### 3.3.3.2利用贝叶斯优化

**贝叶斯优化的理论基础**

贝叶斯优化是一种基于概率建模的智能超参数搜索方法，其核心思想是通过概率模型动态学习和预测参数空间。与传统搜索方法盲目遍历不同，贝叶斯优化能够根据已有试验结果，智能地缩小搜索范围，快速收敛到最优超参数组合。我们选择了Tree-structured Parzen Estimator（TPE）作为采样器，这是一种高级的概率建模算法，能够有效平衡探索（exploration）和利用（exploitation）两个目标。[8]

**TPE具体原理**

基于树结构的Parzen估计器（Tree-structured Parzen Estimator, TPE）是一种基于贝叶斯优化的序列化超参数优化算法，其核心思想是通过构建非参数概率模型指导超参数空间的探索与利用，从而高效搜索最优参数组合。

TPE的实现分为两个阶段：初始化阶段和迭代优化阶段。在初始化阶段，算法通过随机采样生成初始参数集合，并评估其目标函数值。随后进入迭代阶段，每次迭代将历史观测数据划分为表现最优的“好样本”（如前10%分位数）和剩余“差样本”两类。TPE分别对这两类样本的分布进行核密度估计（Kernel Density Estimation,KDE）构建两个非参数概率模型：一个描述高性能参数的概率分布，另一个描述低性能参数的概率分布。通过最大化预期提升量（Expected Improvement, EI）准则，算法选择使提升概率比最大的参数作为下一轮评估点。

该算法的优势体现在其树状结构建模能力，即允许对条件依赖的参数空间进行分层建模，克服了传统高斯过程在复杂参数空间建模的局限性。此外，TPE采用异步更新策略，支持并行化评估，有效平衡了探索（全局搜索）与利用（局部优化）的矛盾。

**优化策略的创新点**

本项目采用的贝叶斯优化方法体现了多个关键创新：

1. **智能参数空间探索**

使用TPE采样器动态构建参数空间的概率模型

根据历史试验结果，智能引导后续参数采样

显著减少盲目搜索的计算开销

1. **计算资源动态管理**

根据GPU可用内存自动调整批量大小

动态限制模型复杂度，避免内存溢出

实现了计算资源的精细化利用

1. **多维度协同优化**

同时优化多个超参数，捕捉它们之间复杂的交互关系

超越传统单一参数调优的局限性

提供更全面、更精准的模型性能优化方案

1. **高效剪枝机制**

引入中位数剪枝（Median Pruner）策略

及时终止表现不佳的试验

显著提高计算效率，缩短整体优化时间

**优化流程与评估**

我们设置了30次试验作为参数搜索的初始规模。每次试验包括：模型训练、性能评估、超参数记录，并以验证集上的曲线下面积（AUROC）作为关键优化指标。

最终通过optuna-dashboard得到如下图像：

最大点值为0.977，具体超参数为（对于浮点数，四舍五入）：

权重衰减为0.065，分类阈值设置为0.72，仅解冻视觉编码器4层 视觉网络：2e-5 分类头：7e-4 元数据编码器：5e-4，批量大小设置为16，训练轮数设置为7轮

### 3.3.4局限与前景

#### 3.3.4.1 局限

本研究在取得显著成果的同时，也存在一些局限性：

1. **数据不均衡问题**：如图1所示，样本中正常样本和其他疾病样本过多，造成了数据分布不均衡。尽管采用了Focal Loss和权重调整等策略，但对于样本量极少的疾病类别，模型识别准确率仍有提升空间。
2. **元数据利用有限**：虽然创新性地加入了元数据处理模块，但目前仅使用了年龄和性别信息。患者的其他临床信息（如既往病史、血糖水平、血压等）未被充分利用，这些信息可能对某些眼科疾病的诊断具有重要价值。
3. **计算资源需求较高**：基于多模态大模型的微调虽然效果优异，但模型参数量大，推理时间长，对硬件要求高，这可能限制其在资源受限环境下的应用。
4. **可解释性挑战**：深度学习模型尤其是Transformer架构的"黑盒"特性，使得模型决策过程难以解释，这在医疗领域可能导致临床应用阻力。[9]

#### 3.3.4.2 前景

尽管存在上述局限，本研究的模型及方法展现了广阔的应用前景：

1. **临床辅助诊断系统**：模型可以集成到眼科门诊的工作流程中，为医生提供初步筛查结果，减轻工作负担，提高诊断效率。特别是在基层医疗机构和眼科医师稀缺地区，可作为重要的辅助诊断工具。
2. **社区筛查应用**：模型可以部署在社区健康中心，用于大规模人群眼底疾病的初筛，及早发现高危人群，尤其对于常见的视网膜病变、青光眼和白内障等疾病具有重要意义。
3. **多模态集成扩展**：未来可以将模型与其他眼科检查结果（如OCT、视野检查等）整合，构建更全面的多模态诊断系统，提高复杂疾病的诊断准确率。
4. **医学教育与培训**：模型可用作医学生和初级眼科医师的教学工具，帮助他们快速学习识别各类眼底病变特征，缩短学习曲线。
5. **远程医疗应用**：结合移动设备和便携式眼底相机，模型可支持偏远地区的远程眼科诊断，缓解医疗资源分布不均的问题。[10]
6. **个性化随访管理**：通过进一步整合时序数据，模型可发展为患者疾病进展监测工具，为个性化治疗方案提供数据支持。
7. **药物研发支持**：在眼科新药临床试验中，模型可作为客观评估工具，减少人工评估的偏差，提高试验数据的可靠性。

基于RET-CLIP的**多模态眼底图像诊断模型**不仅在技术上具有创新性，还具有广泛的临床应用前景。未来的研究方向将聚焦于解决上述局限，进一步提升模型性能，同时拓展其应用场景，为眼科临床实践带来实质性变革。

### 3.3.5参考文献

[1] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PMLR, 2021.

[2] Wang, M., Liu, S., Yang, X., et al. "RET-CLIP: A Vision-Language Foundation Model for Ophthalmic Applications." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 23621-23631.

[3] Pizer, Stephen M., et al. "Adaptive histogram equalization and its variations." *Computer vision, graphics, and image processing*. 39(3), 1987, pp. 355-368.

[4] Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. "Image denoising by sparse 3-D transform-domain collaborative filtering." *IEEE Transactions on image processing*. 16(8), 2007, pp. 2080-2095.

[5] Hendrycks, D., & Gimpel, K. "Gaussian error linear units (GELUs)." *arXiv preprint arXiv:1606.08415*. 2016.

[6] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980-2988.

[7] Li, X., Guo, Y., Zhao, Y., et al. "RETFound: A foundation model for retinal disease diagnosis from ultra-widefield images via multi-center datasets." *Nature Medicine*. 29(12), 2023, pp. 3115-3123.

[8] Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. "Algorithms for hyper-parameter optimization." *Advances in neural information processing systems*. 24, 2011.

[9] Amann, J., Blasimme, A., Vayena, E., Frey, D., & Madai, V. I. "Explainability for artificial intelligence in healthcare: a multidisciplinary perspective." *BMC Medical Informatics and Decision Making*. 20(1), 2020, pp. 1-9.

[10] Rathi, S., Tsui, E., Mehta, N., Zahid, S., & Schuman, J. S. "The current state of teleophthalmology in the United States." *Ophthalmology*. 128(11), 2021, pp. 1489-1499.