基于瑞芯微 RK3588 的智能舌诊系统

及 AI 辅助诊断模型优化

摘要

本系统基于瑞芯微 RK3588 高性能处理器与 OV13855 摄像头，构建了一套嵌入式舌象智能诊断系统。系统通过摄像头实现实时采集舌象图像，将集成舌色特征增强模块的 YOLOv7 模型部署于板端对“红舌（Pink-red）”、“薄白苔（Thin-White）”、“白腻苔（White-Greasy）”、“黄腻苔（Yellow-Greasy）”、“灰黑苔（Grey-Black）” 五类舌象进行分类，再将分类结果输入经过 LORA 微调后的 DeepSeek-R1-Medical-Doctor 模型，生成针对性食疗及调理建议并返回用户界面。系统集成硬件驱动、图像预处理、模型推理及结果展示模块，实现从图像采集到健康建议的全流程本地化处理，为中医舌诊智能化提供轻量化解决方案。系统在嵌入式平台上的实时性与准确性满足基础舌象诊断需求，可应用于家庭健康监测、基层医疗辅助诊断等场景。

第一部分　作品概述

1.1 功能与特性

图像采集：OV13855 摄像头实时开启、拍照及视频录制。

舌象分析：对采集的图像/视频帧进行实时检测，利用 YOLOv7 模型完成五类舌象分类，输出分类置信度。

健康建议生成：基于舌象分类结果，通过微调后的 DeepSeek-R1-Medical-Doctor 模型生成个性化食疗方案与调理建议，如饮食推荐、生活习惯改善等。

交互展示：通过用户界面实时显示采集画面、分类结果及建议内容，支持二维码查询。

系统特性包括全流程本地化处理（无需云端依赖）、低功耗嵌入式部署、实时响应及轻量化模型架构。

1.2 应用领域

家庭健康监测：用户可在家中自主采集舌象，系统实时提供健康评估与调理

建议，辅助慢性病管理（如脾胃虚弱、湿热体质等）。

基层医疗辅助诊断：为乡镇卫生院、社区诊所等基层医疗单位提供舌诊辅助工具，弥补中医诊断资源不足的问题。

远程医疗服务：结合移动终端，支持患者与医生远程对接，通过舌象数据辅助远程问诊。

中医教学与科研：为中医学生提供标准化舌象识别案例，为科研人员提供舌象数据采集与分析平台，助力舌诊客观化研究。

1.3 主要技术特点

异构计算架构：利用 RK3588 的 ARM 处理器+NPU（神经网络处理器）实现模型加速推理，平衡计算效率与功耗。

轻量化模型部署：对 YOLOv7 与 DeepSeek 模型进行量化压缩，适配嵌入式平台内存与算力限制，同时保持整体分类准确率>80%。

全流程本地化处理：从图像采集到建议生成均在板端完成，保障数据隐私，避免网络依赖。

多模态交互设计：支持图像/视频双模式输入，结合图形界面与轻量化交互逻辑，提升用户操作便捷性。

自适应预处理算法：针对舌象采集场景优化图像增强、去噪及舌体分割算法，降低环境光干扰。

1.4 主要性能指标

指标类别	具体指标	数值	备注
模型性能	mAP@0.5	94%	整体分类性能
	mAP@0.5:0.95	75.3%	检测定位精度
推理速度	推理时间	8.6ms	640×640@batchsize=32
	NMS 时间	1.4ms	
系统性能	单帧处理时间	≤20ms	
	端到端响应时间	≤20s	采集到建议生成

1.5 主要创新点

（1）通过 YOLOv7 与 DeepSeek 的级联架构，实现了从图像特征到中医语义的跨模态转换，更贴近真实中医“望诊-辨证”的临床思维过程。

（2）提出舌色特征增强模块（Tongue Color Feature Booster, TCFB），通过强

化舌色特征提取提升 YOLOv7 的分类精度。

（3）嵌入式全流程方案：在 RK3588 平台实现舌象采集—分类—建议生成的端侧闭环，无需云端支持。

（4）模型轻量化集成：通过量化与剪枝技术，将 YOLOv7 与 DeepSeek 模型集成至嵌入式平台，保持高性能同时降低资源占用。

1.6 设计流程

系统设计遵循“需求分析—硬件选型—软件架构设计—模型训练—集成测试”流程：

（1） 基于中医舌诊需求确定五类舌象分类目标；

（2） 选型 RK3588 与 OV13855 构建硬件平台，完成结构设计；

（3） 开发图像采集驱动、图像预处理模块及用户交互界面；

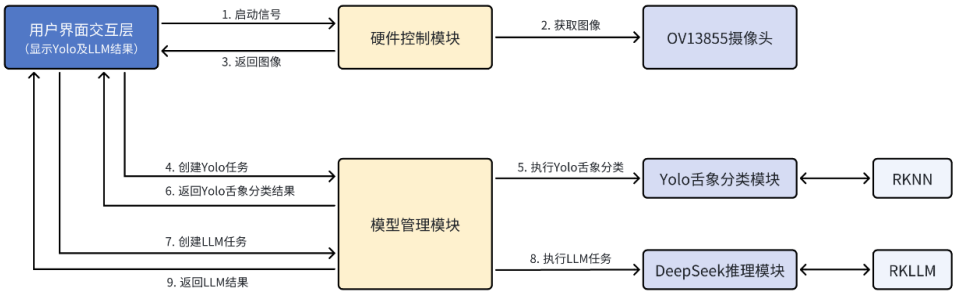
（4） 在数据集上将 TCFB 模块与 YOLOv7 模型相结合，利用其提取的舌色特征来限制 YOLOv7 的检测结果。

（5） 微调 DeepSeek 模型，完成 YOLOv7 与 DeepSeek 板端部署

（6） 全系统联调优化，测试实时性与准确率。

第二部分　系统组成及功能说明

2.1 整体介绍



（1） **用户界面交互层：**负责实时展示摄像头图像，用户可选择开始/停止检测，舌象检测结果、DeepSeek 推理报告将异步更新到用户界面；推理完成后，可生成推理结果二维码；

（2） **硬件控制模块：**负责控制摄像头开启/关闭，开启摄像头时，将图像数据实时发送到用户界面交互层；

（3） **模型管理模块：**负责模型初始化/退出，创建并处理 Yolo 舌象分类模型、DeepSeek 推理模型任务；

（4） **Yolo 舌象分类模块：**预处理上层输入的图像数据，与 RKNN 模块交互，后处理 RKNN 返回的数据；

（5） **DeepSeek 推理模块：**根据 Yolo 分类结果与用户数据，与 RKLLM 模块交互，获取并返回 DeepSeek 推理结果。

2.2 硬件系统介绍

核心处理器：采用瑞芯微 RK3588 处理器，8nm 制程工艺，8 核 64 位 ARM 架构，集成 4×Cortex-A76+4×Cortex-A55 内核。其中 A76 核主频高达 2.4GHz，A55 核主频高达 1.8GHz，内置 6TOPS 算力的神经网络处理器（NPU），可轻松驾驭多种深度学习框架，高效处理 AI 任务；还集成 ARM Mali - G610 MP4 四核 GPU，支持 OpenGL ES3.0/OpenCL2.0/Vulkan1.1，算力达 450GFLOPS，具备 2D 图形加速能力。

图像传感器：OV13855 摄像头，1300 万像素，支持 1080P 视频采集，搭配红外滤光片减少环境光干扰。

摄像头接口：通过 MIPI-CSI 接口连接 OV13855 摄像头，支持高清图像与视频实时采集。

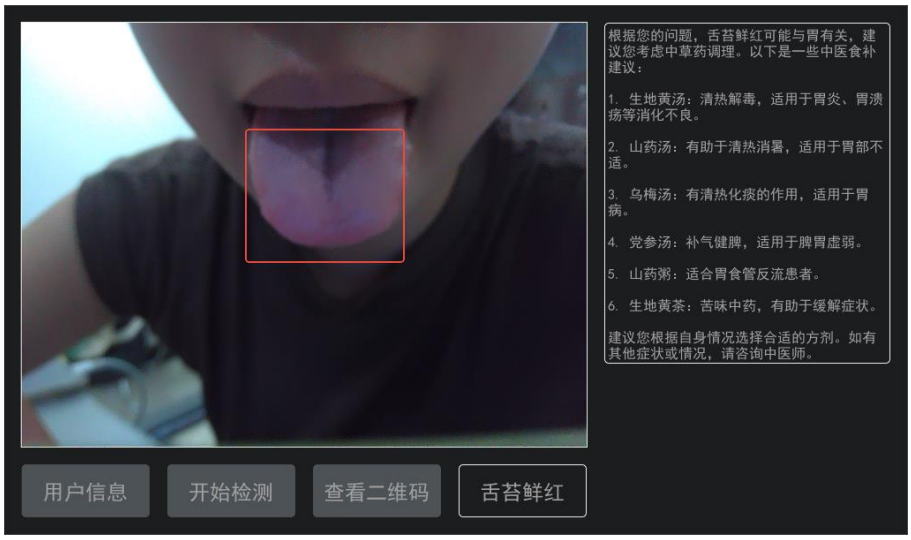
触摸屏接口：适配电容式触摸屏，支持用户通过触控操作完成摄像头控制、模式切换及结果查询等交互功能。

2.3 软件系统介绍

2.3.1 软件整体介绍；

（1） 用户信息界面：

（2） 主界面：

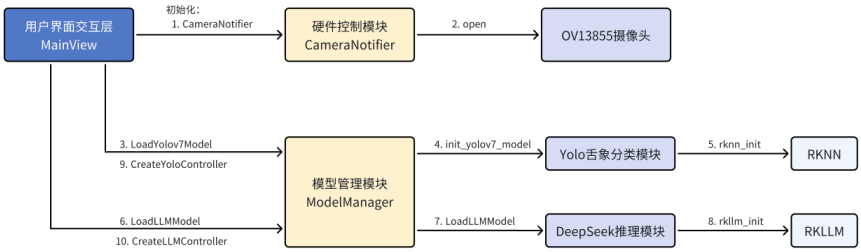


（3） 结果二维码界面：

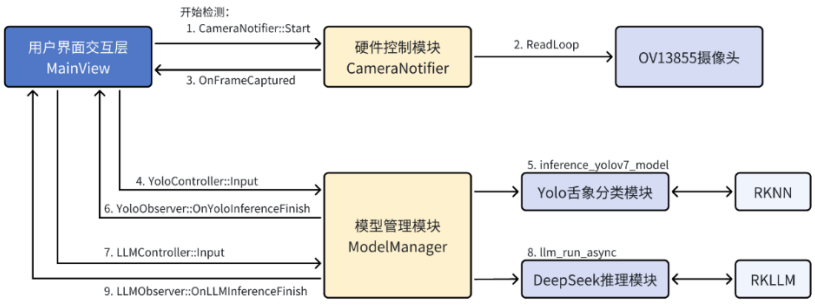
2.3.2 软件各模块介绍

各函数流程图：

（1）初始化流程：

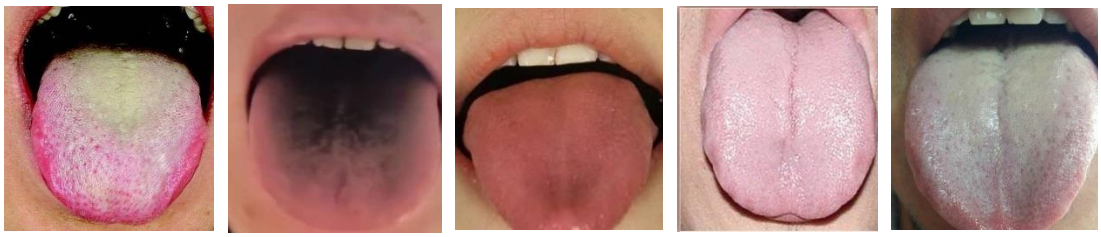


（2）检测流程：



Yolov7 舌象检测与分类各模块函数及关键输入输出变量：

基础数据集来自：飞桨 AI Studio 星河社区舌象特征数据集，该数据集包括以下五种舌象特征类别，均经过专业中医进行标记：①Mirror-Approximated；②Thin-White；③White-Greasy；④Yellow-Greasy；⑤Grey-Black。文件标记均为 VOC 格式，共 1472 张舌象采集图片，包含训练集 941 张、验证集 236 张、测试集 295 张。



（a）黄腻苔　（b）黑灰苔　(c)红舌苔　（d）薄白舌　（e）厚白舌

图 1.五种舌象

舌色特征增强模块（Tongue Color Feature Booster, TCFB），通过强化舌色特征提取提升 YOLOv7 的分类精度。TCFB 的设计使舌色信息成为分类任务的辅助决策因子，其结构如图 2 所示包含五个核心组件：图像预处理、特征提取、非线性激活、特征降维与分类输出。

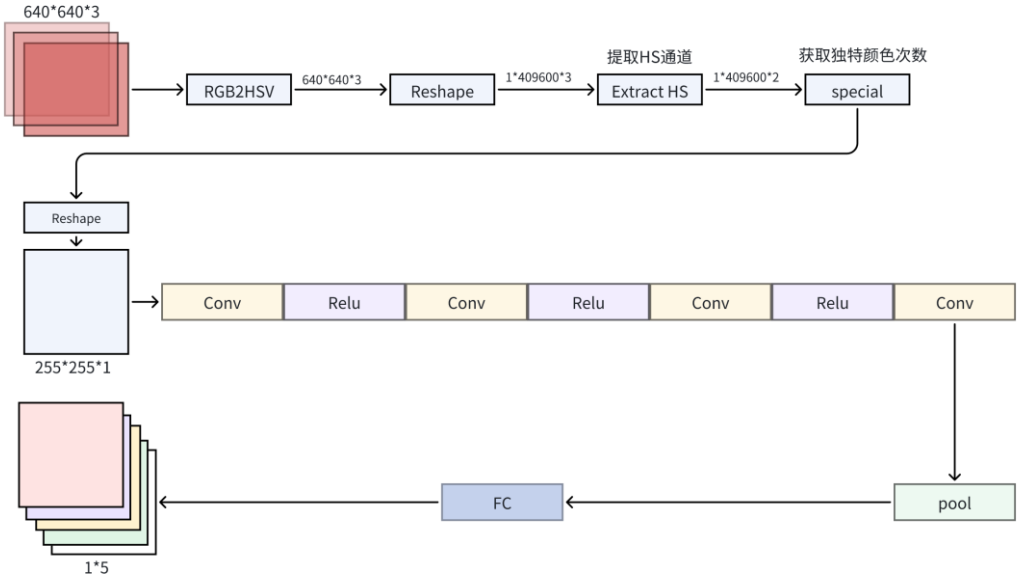


图 2.TCFB 模块

TCFB 的核心工作机制：

1. 深度特征提取：通过多层卷积操作挖掘舌象色相数据的局部模式，分层捕获舌色细微变化与分布特征。

2. 非线性建模：引入 ReLU 激活函数约束正值输出空间，增强模型对复杂舌色关系的拟合能力。

3. 特征压缩：采用全局平均池化实现降维，在降低计算量的同时提升空间鲁棒性。

4. 分类决策：全连接层将特征向量映射至舌色分类空间，输出节点数对应舌象类别数。

（1）图像预处理模块：进行颜色特性提取与转换，将图像从 RGB 颜色空间转换为 HSV 颜色空间，并统计每个颜色的出现次数；数据加载与预处理，加载预处理后的 JSON 数据，进行归一化处理，并转换为 PyTorch 张量。

函数名/核心类	描述
Calculation_List	创建一个长度为 65026 的数组，记录每种颜色的出现次数 参数： image_path：图像文件的路径。 color_counts_list：长度为 65026 的一维数组，记录每种颜色的出现次数。
write_json	读取指定类型（训练集或验证集）的图像文件，调用 Calculation_List 函数计算颜色出现次数，将结果和标签信息写入 JSON 文件。
MyDataset	自定义数据集类，用于加载训练集和验证集的数据。 参数：数据样本的索引 返回值：包含图像和标签的列表
MyModel	通过卷积层和激活函数进行特征提取；通过全局平均池化层获取固定大小的输出；通过全连接层进行分类 参数：输入的图像张量 返回值：分类结果张量

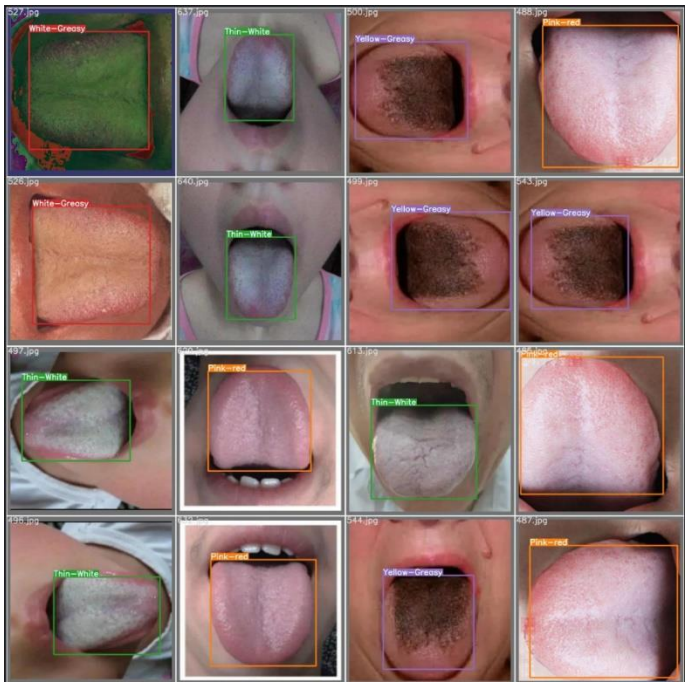
（2）模型训练模块：对目标检测模型进行训练，支持多种配置选项，如模型权重加载、数据集指定、超参数设置、训练轮数、批量大小等。脚本包含了模型初始化、数据加载、优化器设置、学习率调度、训练循环、验证评估、模型保存等完整的训练流程，同时支持分布式训练和超参数进化。

函数名/核心类	描述
train(hyp, opt, device, tb_writer=None)	设置保存目录和配置文件；加载数据集和模型；冻结模型层；设置优化器和学习率调度器；使用指数移动平均（EMA）训练循环；验证和保存模型；训练结束后处理 参数： 1. hyp：超参数字典，包含学习率、动量、损失函数权重等训练相关的超参数。 2. opt：命令行参数解析结果，包含模型路径、数据路径、训练轮数、批次大小等配置信息。 3. device：训练设备，如 GPU 或 CPU。 4. tb_writer：TensorBoard 写入器，用于记录训练过程中的指标，如损失值、准确率等。
create_dataloader	创建数据加载器，用于加载训练集或验证集的数据。它会对数据进行预处理，如数据增强、图像大小调整等，并将数据分批加载到内存中。 参数： 1. path：数据集路径。 2. imgsz：图像大小。 3. batch_size：批次大小。 4. gs：网格大小。 5. opt：命令行参数解析结果。 6. hyp：超参数字典。
check_anchors	检查并调整模型的锚框（anchor boxes）。

（3）模型测试模块：评估训练好的目标检测与关键点检测模型的性能，通过在验证集或测试集上进行推理，计算关键指标并可视化结果，为模型优化提供依据。

函数名/核心类	描述
test.test	加载训练好的目标检测模型（如 EMA 平滑后的模型），对验证集图像进行批量推理，输出检测框、置信度及类别预测。计算目标检测的核心评估指标，包括：精确率，召回率，mAP@0.5，mAP@0.5:0.95 结果可视化与保存
IKeypoint	关键点定位推理：对输入特征图进行前向计算，输出目标关键点的坐标及置信度。 坐标解码与后处理：将模型预测的归一化坐标转换为图像像素坐标，结合锚框信息调整关键点位置。 关键点评估指标：计算关键点检测的精度衡量关键点定位的准确性。

（4）检测结果



界面开发各模块函数及关键输入输出变量：

（1）硬件管理模块（CameraNotifier）：

函数名	描述
CameraNotifier(int index, CameraObserver* observer)	初始化摄像头 参数： index：摄像头节点； observer：摄像头观察者
void CameraNotifier::Start() void CameraNotifier::Stop()	启动/停止摄像头数据读取和处理，创建/等待定时器和读取线程
void CameraNotifier::ReadLoop()	在循环中持续读取摄像头帧数据，进行格式转换并控制帧率

（2）摄像头观察者（CameraObserver）：

函数名	描述
void OnFrameCaptured(std::unique_ptr<ImageBuffer> buffer)	当获取到图像数据时调用，传递图像数据 参数：图像数据 1.buffer：

（3）模型管理模块（ModelManager）：

函数名	描述
bool LoadYolov7Model(const std::string& model_path)	加载 Yolo 模型 参数： 1. model_path： 模型文件路径 返回值：是否加载成功
bool LoadLLMModel(const std::string& model_path)	加载 LLM 模型 参数： 1. model_path： 模型文件路径 返回值：是否加载成功
YoloController * CreateYoloController(YoloController::YoloObserver* observer)	创建 Yolo 控制类 参数： 1. observer：Yolo 任务观察者 返回值：Yolo 控制对象
LLMController * CreateLLMController(LLMController::LLMObserver* observer)	创建 Yolo 控制类 参数：observer：LLM 任务观察者 返回值：LLM 控制对象

（4）Yolo 控制对象（YoloController）：

函数名	描述
YoloController::YoloController(rknn_app_context_t* rknn_ctx, YoloObserver* observer)	初始化 YoloController 对象，创建工作线程 参数： 1. rknn_ctx：rknn 应用上下文指针； 2. observer：Yolo 观察者指针；
boolYoloController::Input(std::unique_ptr<ImageBuffer> buffer)	接收图像缓冲区数据并唤醒工作线程 参数： 1. buffer：图像缓冲区

	智能指针 返回值：接收是否成功
bool YoloController::GetYoloClassName(int class_id, std::string& class_name)	获取类别名称 参数： 1. class_id：类别 ID； 2. class_name：类别字符串引用 返回值：是否成功获取
void YoloController::WorkThread()	工作线程函数，进行 Yolo 模型推理，在有图像数据时调用模型推理并通知观察者

（5）Yolo 任务观察者（YoloObserver）：

函数名	描述
void OnYoloInferenceFinish(std::unique_ptr<ImageBuffer> buffer, std::unique_ptr<object_detect_result_list> od_results)	当 Yolo 模型推理完成时调用，传递原始图像数据和检测结果 参数： 1. buffer：包含原始图像数据的智能指针； 2. od_results：包含目标检测结果的智能指针
void OnYoloInferenceError()	当 Yolo 模型推理过程中发生错误时调用

（6）LLM 控制对象（LLMController）：

函数名	描述
LLMController::LLMController(LLMHandle llmHandle, RKLLMPromptCacheParam prompt_cache_params, LLMObserver* observer)	初始化 LLM 控制器，设置 LLM 句柄、提示缓存参数和观察者 参数： 1. llmHandle：LLM 引擎句柄； 2. prompt_cache_params：提示缓存参数；

	3. observer：LLM 观察者指针
void LLMController::OnLLMInferenceRunning(const std::string& text)	在 LLM 推理过程中被回调，积累输出文本并通知观察者 参数： 1. text：生成的文本片段
bool LLMController::Input(const std::string& input_text)	处理用户输入文本，构建提示模板并异步启动 LLM 推理 参数： 1.input_text 用户输入的文本 返回值：是否成功启动推理
const std::string& LLMController::GetOutputText() const	获取 LLM 生成的完整输出文本 返回值：生成的文本

（7）LLM 任务观察者（LLMObserver）：

函数名	描述
void OnLLMInferenceRunning(const std::string& text)	推理进行中时通知进度 参数： 1.text：推理增量文本
void OnLLMInferenceFinish()	LLM 推理完成时的回调函数，通知观察者推理结束
void OnLLMInferenceError()	LLM 推理出错时的回调函数，通知观察者推理失败
void WaitFinish()	阻塞等待 LLM 推理完成

LORA 微调 DeepSeek 模型各模块函数及关键输入输出变量：

系统基于 DeepSeek-R1-Distill-Qwen-1.5B 模型，使用 LoRA 技术进行参数高效微调，用于医学知识问答。系统设计包含模型加载、数据处理、训练三个核心模块。微调数据集为：FreedomIntelligence/medical-o1-reasoning-SFT 用于支持医疗问答、临床诊断推理等自然语言处理任务的监督微调。其具体的数据构成如下

图 2：包含问题、医学知识推理过程、答案等 20.2k 行数据。

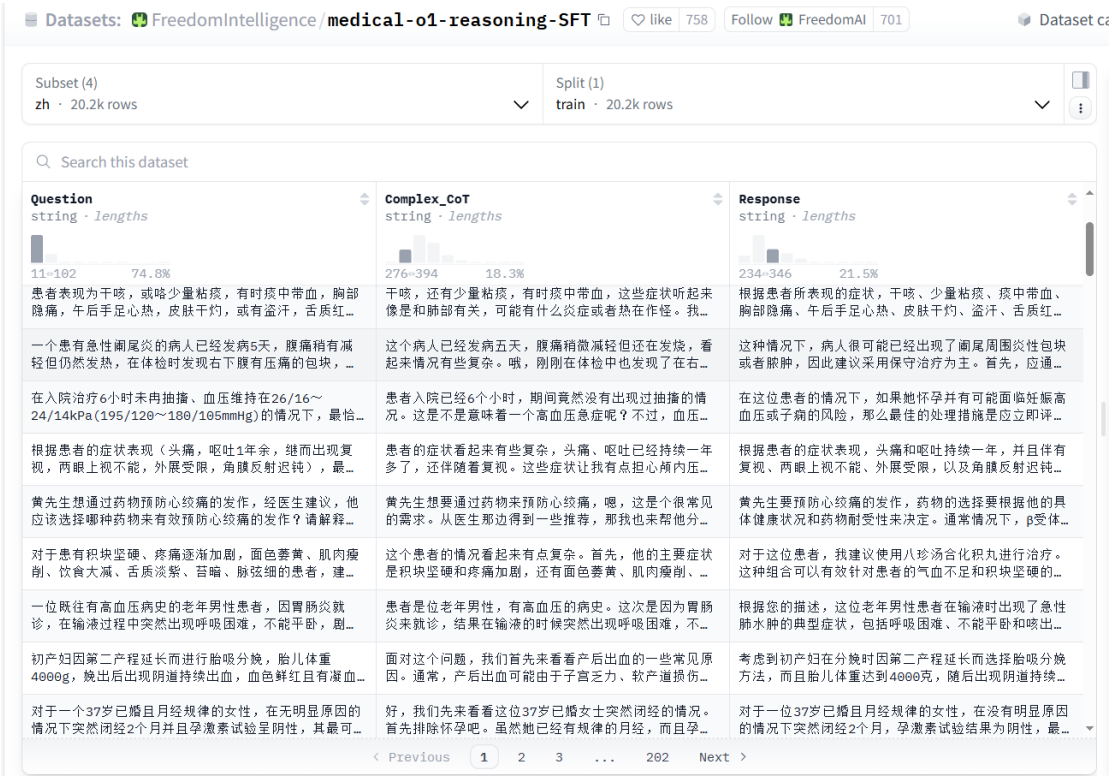


图 2.medical-o1-reasoning-SFT 数据集部分截图

（1）模型加载：

函数名/核心类	描述
def FastLanguageModel.from_pretrained(model_name: str, max_seq_length: int = 2048, dtype: torch.dtype = torch.bfloat16, load_in_4bit: bool = True, token: str = None)	加载预训练语言模型和分词器 参数： 1.model_name: 模型名称或路径 2.max_seq_length: 最大序列长度 3.dtype: 数据类型 4.load_in_4bit: 是否使用 4 位量化 5.token: Hugging Face API 令牌

FastLanguageModel.get_peft_model(model: PreTrainedModel, r: int = 16, target_modules: List[str] = None, lora_alpha: int = 16, lora_dropout: float = 0.0, bias: str = "none", use_gradient_checkpointing: Union[bool, str] = "unsloth", random_state: int = 3407, use_rslora: bool = False, loftq_config: dict = None)	为模型配置 LoRA 参数高效微调 参数： 1.model: 基础预训练模型 2.r: LoRA 秩 3.target_modules: 要应用 LoRA 的模块名称列表 4.lora_alpha: LoRA 缩放因子 5.lora_dropout: LoRA dropout 率 6.bias: 偏置参数训练策略 7.random_state: 随机种子

（2）数据处理模块

函数名/核心类	描述
formatting_prompts_func(examples): inputs = examples["Question"] cots = examples["Complex_CoT"] outputs = examples["Response"] texts = [] for input, cot, output in zip(inputs, cots, outputs): text = train_prompt_style.format(input, cot, output) + EOS_TOKEN	将原始数据格式化为模型训练所需的提示格式 参数： 1.examples: 包含原始数据的字典包括"Question", "Complex_CoT", "Response" 返回值：格式化后的文本列表

（3）训练模块

函数名/核心类	描述
def SFTTrainer(model: PreTrainedModel, tokenizer: PreTrainedTokenizer, train_dataset: Dataset, dataset_text_field: str = "text", max_seq_length: int = 2048, dataset_num_proc: int = 2,	监督微调训练器 参数： 1.model: 要训练的模型 2.tokenizer: 分词器 3.train_dataset: 训练数据集 4.max_seq_length: 最大

args: TrainingArguments, packing: bool = True）	序列长度 5.dataset_num_proc: 数据处理进程数 6.args: 训练参数

（4）训练结果：随着训练进行，模型在逐渐学习，预测值与真实值的差距在缩小，模型性能在提升。

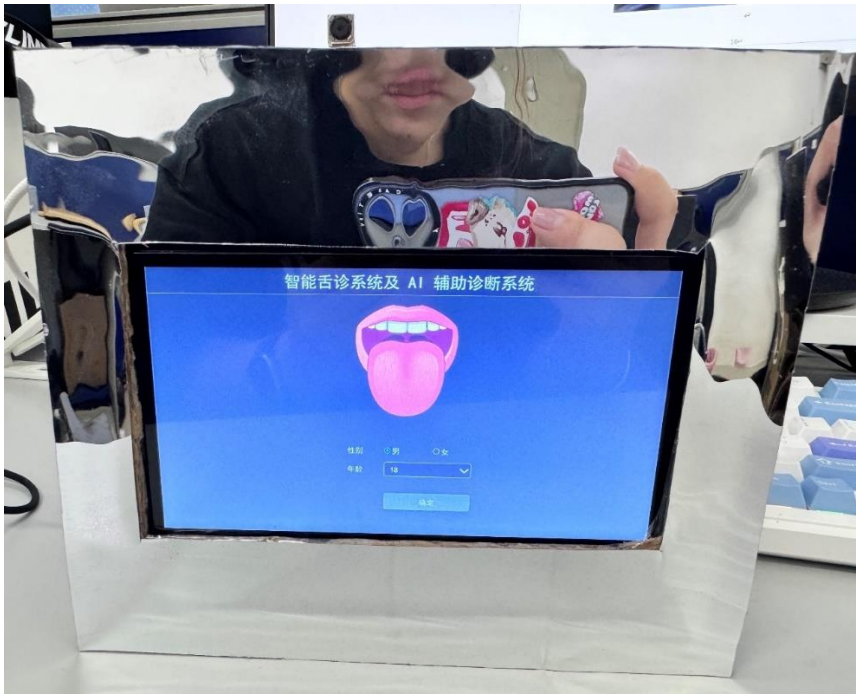
```
0^0/ \_/ \    Batch size per device = 1 | Gradient accumulation steps = 4
\        /    Data Parallel GPUs = 1 | Total batch size (1 x 4 x 1) = 4
 "-____-"     Trainable parameters = 18,464,768/1,500,000,000 (1.23% trained)
```

Step	Training Loss
10	4.534300
20	3.955200
30	3.643900
40	3.777300
50	3.395000
60	3.424900

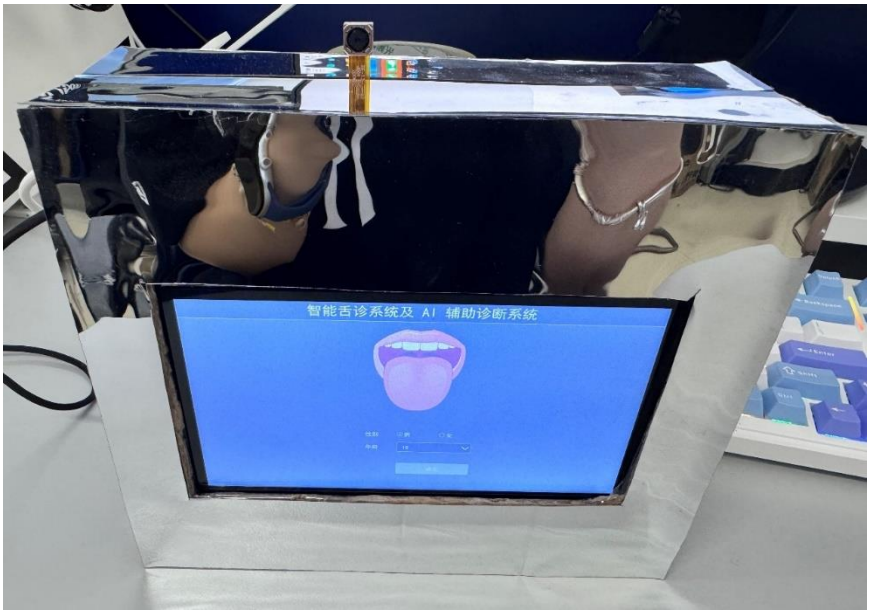
第三部分　完成情况及性能参数

3.1 整体介绍

系统实物正面：

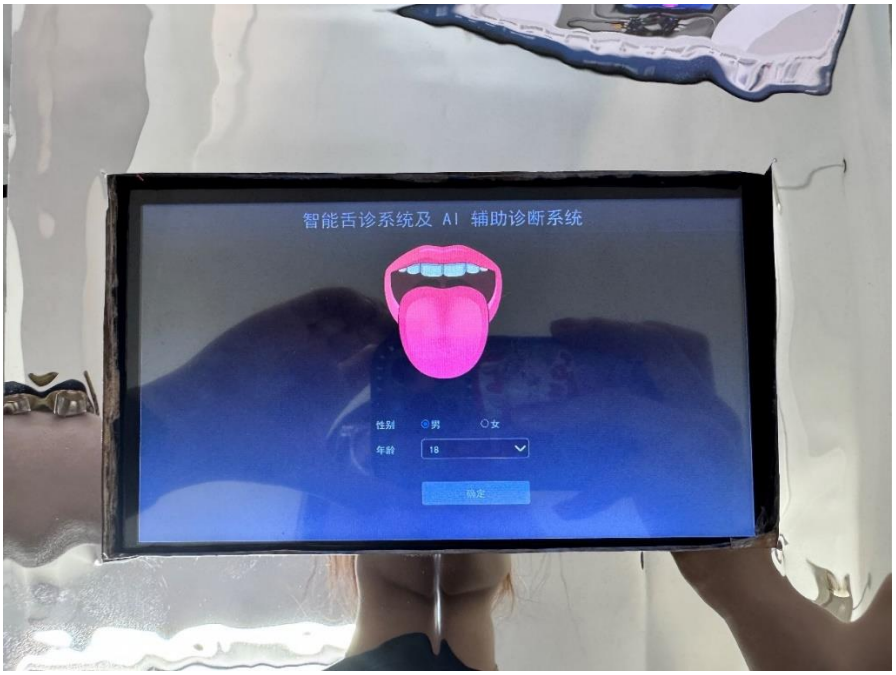


斜 45°全局性照片：

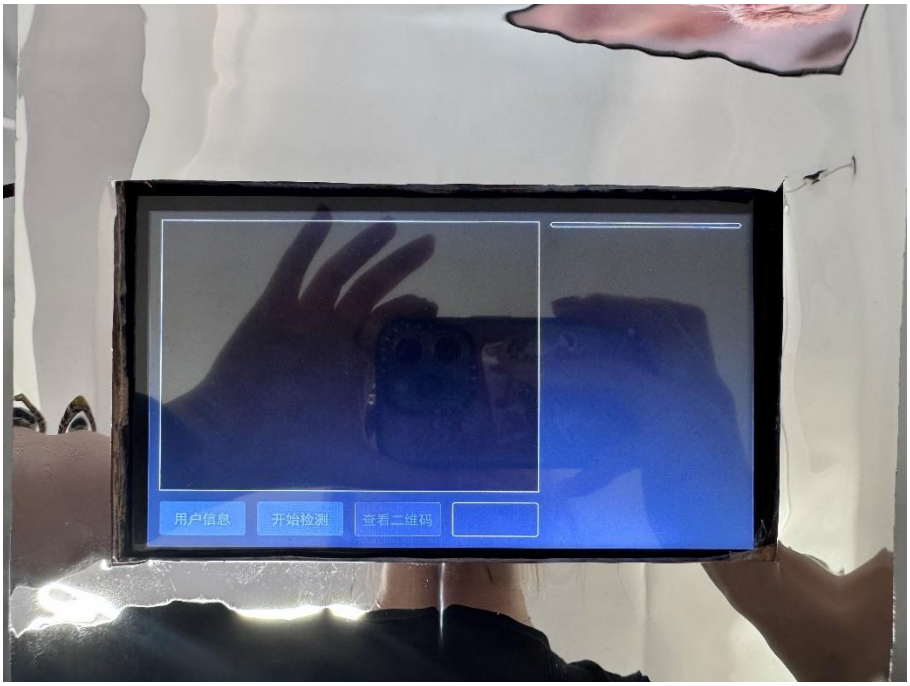


3.2 工程成果

（1）用户选择性别及年龄界面：

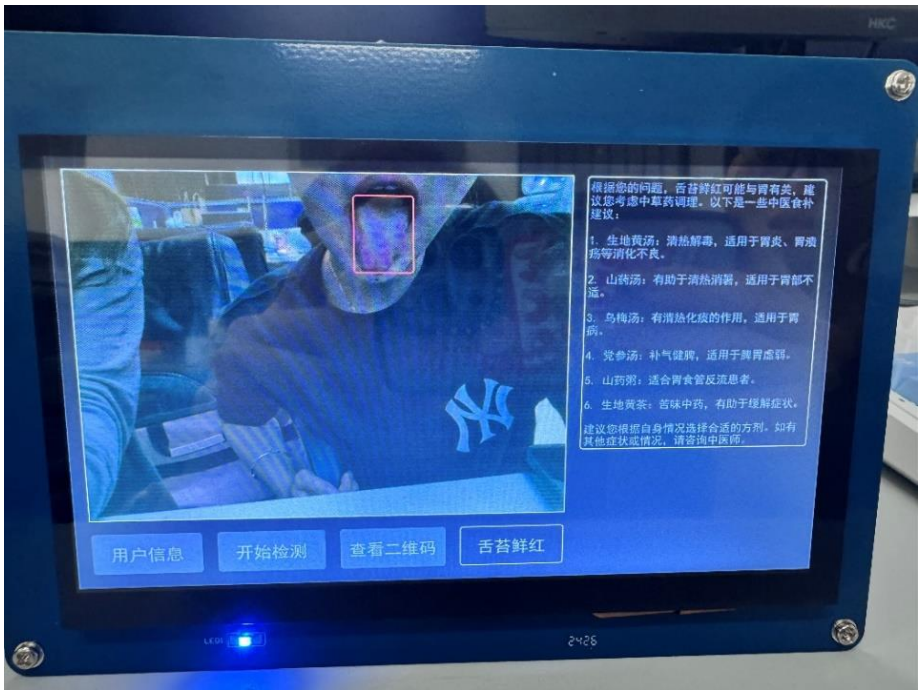


（2）检测舌象界面：



（3）检测结果及生成诊断建议界面：

年龄 22 岁性别女，舌苔鲜红：



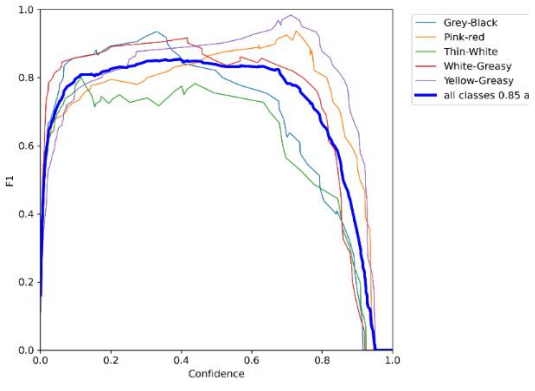
年龄 35 岁性别女，舌苔白而薄：

（4）查看二维码界面：

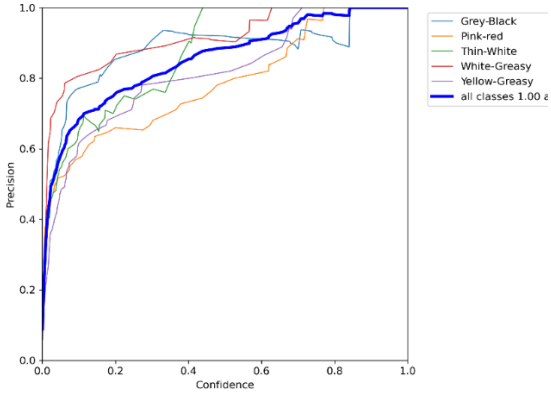
3.3 特性成果

（1）目标检测模型的评估结果：

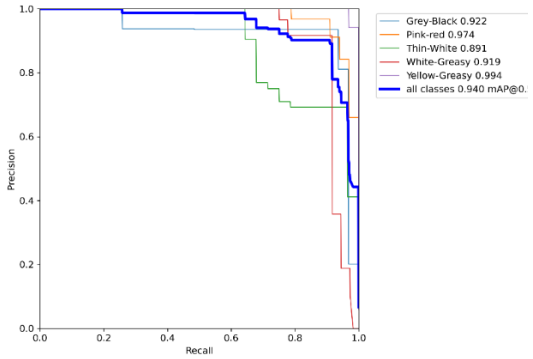
```
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100%|██████████| 5/5 [00:02<00:00,  1.68it/s]
                 all         160         160       0.852       0.874        0.94       0.753
         Grey-Black          160          31       0.926       0.807       0.922        0.75
           Pink-red          160          33       0.732        0.97       0.974       0.822
         Thin-White          160          28       0.895       0.679       0.891       0.719
       White-Greasy          160          36        0.91       0.917       0.919       0.725
      Yellow-Greasy          160          32       0.799           1       0.994       0.749
Speed: 8.6/1.6/10.2 ms inference/NMS/total per 640x640 image at batch-size 32
```



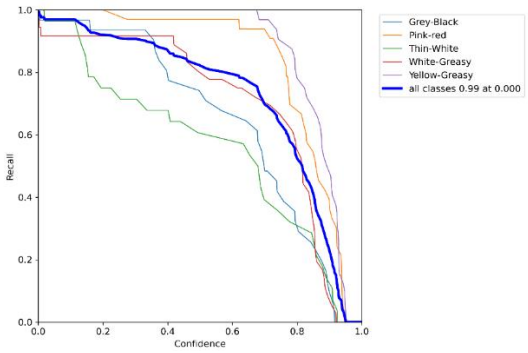
（a）F1



（b）P_curve



（c）PR_curve



（d）R_curve

（2）大模型微调后指标：

```
0^0/ \_/ \    Batch size per device = 1 | Gradient accumulation steps = 4
\         /   Data Parallel GPUs = 1 | Total batch size (1 x 4 x 1) = 4
 "-____-"    Trainable parameters = 18,464,768/1,500,000,000 (1.23% trained)
```

[60/60 01:55, Epoch 1/2]

Step	Training Loss
10	4.534300
20	3.955200
30	3.643900
40	3.777300
50	3.395000
60	3.424900

第四部分 总结

4.1 可扩展之处

1.增加更多中医诊断维度（脉象等），可增加“裂纹舌”“齿痕舌”等更多舌象分类，提升诊断维度。

2.接入更多中医知识图谱结合脉象、面诊等数据，构建更全面的中医智能诊断系统。

3.引入迁移学习进一步提升小样本舌象的分类准确率。

4.增加数据云端同步功能（需加密处理），支持远程阅片与历史数据追踪。

4.2 心得体会

在开发基于瑞芯微 RK3588 的智能舌诊系统过程中，我们深刻体会到从 GPU 训练环境到嵌入式设备部署的转化充满挑战。特别是模型部署阶段遇到的兼容性问题，让我们对异构计算环境下的模型优化有了更深入的理解。

RK3588 的 ARM+NPU 异构架构，是项目的核心支撑。最初，我们尝试纯 CPU 推理，发现面对 YOLOv7 和 DeepSeek 模型，计算耗时过长，无法满足实时性需求。引入 NPU 加速后，通过合理分配任务——ARM 处理器负责图像预处理、流程调度等通用计算，NPU 专注模型推理的并行计算，实现了效率与功耗的平衡。在舌象分类任务中，NPU 加速让推理时间从秒级压缩到百毫秒级，同时功耗控制在合理范围。异构计算架构对边缘设备具有一定的重要性，并且硬件协同设计是发挥系统性能的基础。

后续我们对模型进行优化，以适应嵌入式平台的内存和算力的限制。针对 YOLOv7 和 DeepSeek，我们采用量化压缩、结构剪枝等策略。量化过程中，平衡精度与速度是关键，经过多轮测试，在分类准确率大于 80%的前提下，将模型体积大幅压缩，来适配 RK3588 的内存资源。这里的轻量化不是简单的“瘦身”，而是诊断准确率与硬件能力间找到平衡点，多次进行参数调整以及结构优化，以满足模型和嵌入式平台的适配。

研发过程中从 OV13855 摄像头采集图像，到 Yolo 舌象分类、Deepseek 推理生成建议，所有环节在板端完成。这一设计最初是为满足医疗场景的数据隐私要求，但实践中发现，它同时解决了网络依赖问题，即使在弱网或无网环境，系统也能稳定运行，实现了“全流程本地化”这一重要理念。开发中，我们需优化数据流转效率，确保图像预处理、模型推理、结果输出的流程顺畅，功能设计要兼顾合规性与用户体验，技术方案需从场景本质出发。

多模态交互和自适应预处理算法，是提升用户体验的关键。舌诊场景中，环境光、舌体姿态差异大，自适应增强、去噪算法有效降低了这些干扰；轻量化交互逻辑让非专业用户也能便捷操作。开发时，需反复模拟实际使用场景，比如测试不同光照下的舌象采集效果、优化界面响应速度。如果这门技术最终服务于人，贴近用户需求、适配实际场景，那么这个系统才是一个真正“可用”“好用”的系统。

模型从GPU训练到RK3588部署，曾遭遇兼容性难题。最初用RKNN-Toolkit2转换模型时，算子不兼容问题频发。我们尝试直接将 GPU 训练好的模型通过ONNX 转换到 RKNN 格式，却忽略了 NPU 对算子的支持限制。通过研究硬件手册、调整模型转换策略，最终实现稳定部署。这一过程让我们学到，模型工程化是跨领域的技术融合，既要懂算法训练，也要熟悉硬件特性、工具链原理，每解决一个部署问题，都是对技术栈的拓宽。

回顾项目，从异构架构利用到模型轻量化，从本地化设计到用户体验优化，每个环节都交织着挑战与成长。智能系统开发不是单一技术的堆叠，而是硬件、算法、软件、场景的深度融合。未来，期待在边缘 AI 医疗应用中继续探索，让技术真正赋能临床、服务用户，把“智能舌诊”的创新从实验室推向更广阔的实际场景。

第五部分 参考文献

[1] 孙岩,周立新,孙连君,等.知识图谱和大语言模型在航天测控问答系统中的融合应用研究[J].上海航天(中英文),2024,41(05):178-184.DOI:10.19328/j.cnki.2096-8655.2024.05.020.

[2] 王玲,林依凡,李璐.智能诊疗在舌象研究中的应用进展[J].中华中医药杂志,2021,36(01):342-346.

[3] 陆翔,冯跃,贾旭东,等.基于多尺度特征融合与混合注意力机制的舌象齿痕识别方法研究[J].五邑大学学报(自然科学版),2025,39(02):63-70.

[4] 邢家旺.基于嵌入式 NPU 的目标检测系统研究[D].石家庄铁道大学,2024.DOI:10.27334/d.cnki.gstdy.2024.000233.

[5] 高子召.RK3588 物体检测方法研究[D].长江大学,2024.DOI:10.26981/d.cnki.gjhsc.2024.000125.

[6] 胡馨之.基于深度学习的无人机航拍图像车辆目标检测算法研究[D].杭州电子科技大学,2023.DOI:10.27075/d.cnki.ghzdc.2023.001598.

[7] 杨诗卓,周燕玲,解翔杰.融合 RAG 检索增强与 LoRA 微调的医疗问答系统优化研究[J/OL].江西科学,1-8[2025-07-07].http://kns.cnki.net/kcms/detail/36.1093.N.20250507.1655.002.html.