PaddleOCR需求分析报告

项目目标

使用RUST语言开发一个基于PaddleOCR的深度学习模型，主要目标包括：

1. 用于高效且准确地对获取的游戏画面进行图像处理，**根据圣遗物的截图以及文本坐标，从图像中提取文本信息**；
2. 通用预训练模型，不用训练即可直接使用；
3. 兼容现有的深度学习框架和工具，以便用户可以轻松地集成和扩展；
4. 确保模型的高准确性和鲁棒性，能够在面对不同的游戏图像下都能准确识别；
5. 支持多种模型格式（如ONNX），并能够在不同的操作系统（Windows、MacOS、Linux）环境下高效运行。

功能需求

1. **模型加载和初始化**：
    - 提供简洁的接口，用于加载预训练的PaddlePaddle模型。
    - 支持多种模型格式的导入，包括ONNX等。
2. **图像预处理**：
    - 提供一系列图像预处理工具，确保输入图像能够满足模型的要求，包括图像的缩放、裁剪等操作。
    - 提供图像归一化功能，支持不同模型框架（如`ort`和`tract_onnx`）。
3. **模型推理**：
    - 支持对单张图像和批量图像的推理。
    - 提供预测结果的后处理工具，以便用户可以直接使用模型输出。
4. **文本提取和保存**：
    - 提供结果可视化工具，方便用户查看模型预测结果。
    - 支持将预测结果保存为多种格式的文件。
5. **配置管理**：
    - 支持通过配置文件或代码参数设置模型的各种超参数和运行选项。
6. **多操作系统支持**：
    - 支持在不同的操作系统包括Windows、MacOS、Linux环境下高效运行。

性能需求

1. **高效推理速度**：
    - 确保模型在CPU和GPU上均能高效运行，满足实时处理的要求。
    - 优化模型的计算效率，减少推理时间，以处理大量圣遗物信息。
2. **内存和资源管理**：
    - 优化内存使用，确保在有限的硬件资源下模型能够稳定运行。
    - 提供对大批量数据处理的支持，避免内存溢出问题。
3. **扩展性**：

- 模型和代码设计应具备良好的扩展性，方便后续集成新的功能模块。

可维护性需求

1. **代码结构清晰**：
    - 采用模块化设计，确保每个功能模块独立且易于理解。
    - 提供详细的注释和文档，方便开发人员和用户理解和使用。
2. **易于调试和测试**：
    - 提供完善的测试用例，确保每个功能模块的正确性。
    - 设计良好的日志记录机制，方便追踪和定位问题。
3. **兼容性和可移植性**：
    - 确保代码能够在不同的操作系统和硬件平台上运行。
    - 提供详细的依赖和安装指南，方便用户部署和使用。
4. **持续集成和更新**：
    - 支持持续集成（CI）和持续部署（CD），确保代码的稳定性和更新的及时性。
    - 定期更新和维护项目，及时修复已知问题和漏洞。

YasOCR需求分析报告

项目目标

使用RUST语言开发针对特定场景的OCR专用模型，主要目标包括：

- 用于高效且准确地对获取的游戏画面进行图像处理，**根据圣遗物的截图以及文本坐标，从图像中提取文本信息**；
- 针对特定游戏（原神、崩坏：星穹铁道）字符进行训练的OCR模型，能够提供对于这类字符类型的高精度文本识别；
- 拥有比通用模型更快的处理能力，确保对大规模图像数据进行快速处理，满足大量扫描的实时性需求。
- 具有较高的模块化，便于与其他软件系统集成，并且可以根据需求进行扩展和优化。

功能需求

1. **针对项目应用的专用模型：**
    - 与通用的paddleOCR不同，YasOCR针对项目应用游戏进行特殊优化
    - 再识别特定游戏（原神、崩坏：星穹铁道）字符上拥有更快、精度更高的模型性能。
2. **图像预处理：**
    - 与通用模型一样，能对图像进行一系列预处理；
    - 保证预处理后的图像能够正确应用专用OCR模型；
3. **模型接口**：
    - 支持从文件加载字符映射表，可以对识别内容进行实时更新；
    - 提供多种图像输入类型（RGB图像、灰度图像等）的推理接口，将图像转换为文本。
4. **配置管理**：

    - 支持通过配置文件或代码参数设置模型的各种超参数和运行选项。
5. **多操作系统支持**：
    - 支持在不同的操作系统包括Windows、MacOS、Linux环境下高效运行。

性能需求

1. **更快的推理速度**：
    - 针对特定游戏字符拥有更快的处理速度，在普通硬件环境下，单张图像的预处理和推理总时间**不超过100毫秒**。
2. **针对特定游戏字符拥有更精准的识别能力**
    - 针对特定游戏字符拥有更高的准确率，在实际应用中超过当前主流OCR模型在特定游戏上的识别精度。

可维护性需求

1. **代码结构清晰**：
    - 采用模块化设计，确保每个功能模块独立且易于理解。
    - 提供详细的注释和文档，方便开发人员和用户理解和使用。
2. **易于调试和测试**：
    - 提供完善的测试用例，确保每个功能模块的正确性。
    - 设计良好的日志记录机制，方便追踪和定位问题。
3. **兼容性和可移植性**：
    - 确保代码能够在不同的操作系统和硬件平台上运行。
    - 提供详细的依赖和安装指南，方便用户部署和使用。
4. **持续集成和更新**：
    - 支持持续集成（CI）和持续部署（CD），确保代码的稳定性和更新的及时性。
    - 定期更新和维护项目，及时修复已知问题和漏洞。