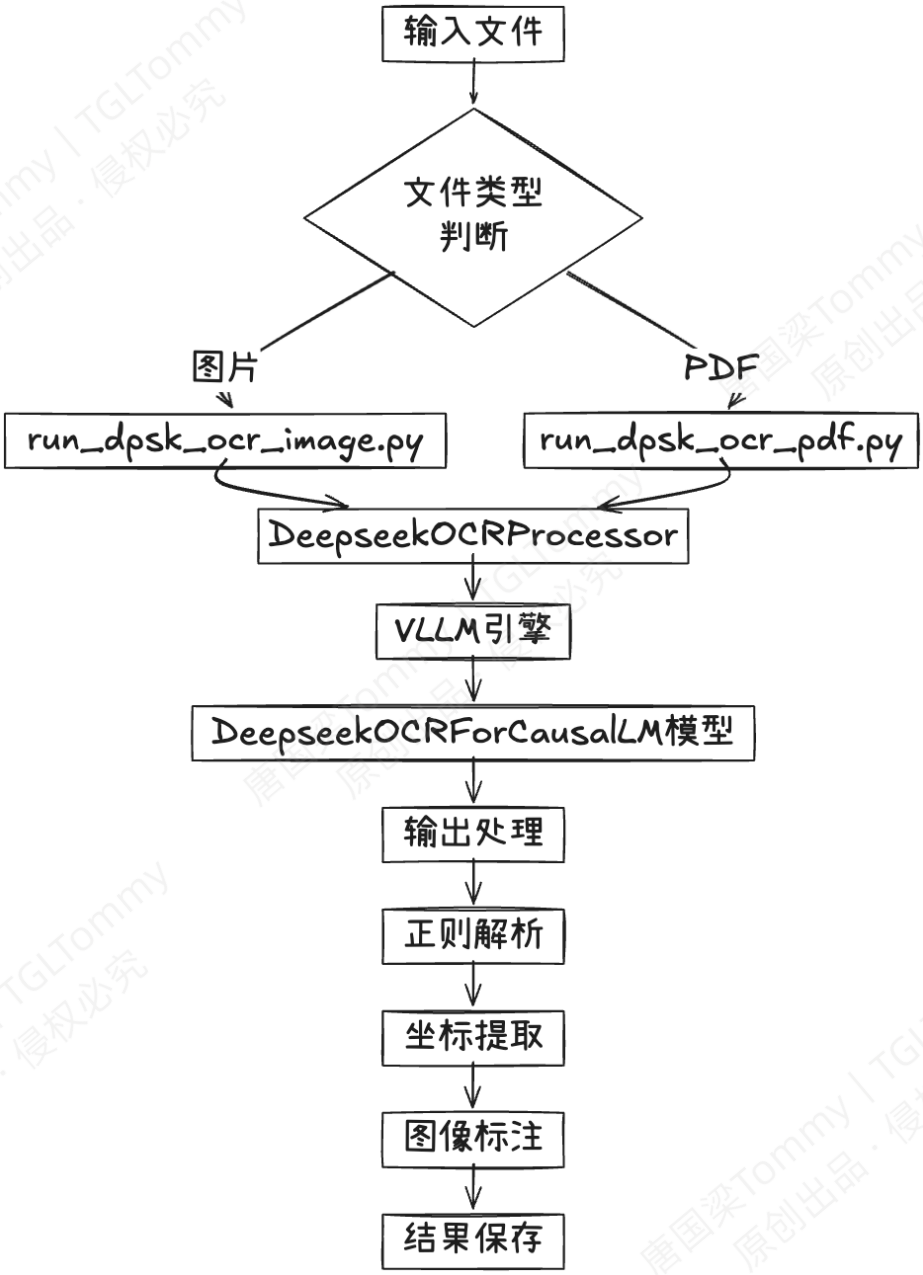
一、项目简介

DeepSeek-OCR是一个多模态大语言模型，专门用于OCR任务，采用双视觉编码器 + 语言模型的架构设计。

1.1 推理流程图



1.2 核心组件

这个架构的核心创新在于双视觉编码器的协同工作：1️⃣ SAM负责细节分割，2️⃣ CLIP负责语义理解，3️⃣ 通过投影层统一特征空间，最终与语言模型深度融合，实现了高精度的OCR识别能力。

组件	文件位置	作用	技术特点
主模型	`deepseek_ocr.py`	多模态推理处理器	• 支持图文输入 • vLLM兼容 • 动态语言模型选择
SAM编码器	`deepencoder/sam_vary_sdpa.py`	分割感知编码	• ViT-B架构 • 相对位置编码 • 多尺度特征提取
CLIP编码器	`deepencoder/clip_sdpa.py`	视觉理解编码	• 24层Transformer • 1024维隐藏层 • Flash Attention优化
投影层	`deepencoder/build_linear.py`	特征维度统一	• 线性投影 • MLP投影 • 特征融合

二、环境配置与模型下载

2.1 系统要求

```
Ubuntu 22.04
GPU 4090 24GB
```

2.2 环境配置

```
# 创建虚拟环境
conda create -n deepseek-ocr python=3.12.9 -y
conda init bash && source /root/.bashrc
conda activate deepseek-ocr

# 安装 PyTorch
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url
https://download.pytorch.org/whl/cu118

# 安装 tokenizers, transformers
pip install tokenizers==0.21.1 transformers==4.51.1

# 安装 vLLM
pip install vllm=0.8.5

# 安装项目依赖
pip install -r requirements.txt

# 安装 Flash Attention
pip install flash-attn==2.7.3 --no-build-isolation
```

2.3 模型下载

```
# 从 modelscope 下载模型
# https://www.modelscope.cn/models/deepseek-ai/DeepSeek-OCR

pip install modelscope

modelscope download --model deepseek-ai/DeepSeek-OCR --local_dir /root/autodl-tmp/DS-OCR
```

三、实战案例

3.1 案例-1：单张图像OCR实战

步骤1: 配置参数

```
# 修改config.py文件
MODEL_PATH = "/path/to/your/model"  # 模型路径
INPUT_PATH = "/path/to/your/image.png"  # 输入图片
OUTPUT_PATH = "/path/to/output"  # 输出目录
PROMPT = '<image>\n<|grounding|>Convert the document to markdown.'  # 提示词
```

- **配置模式**

模式	参数设置	特点	适用场景
Tiny	base_size=512, image_size=512, crop_mode=False	64 vision tokens	快速推理
Small	base_size=640, image_size=640, crop_mode=False	100 vision tokens	平衡性能
Base	base_size=1024, image_size=1024, crop_mode=False	256 vision tokens	标准精度
Large	base_size=1280, image_size=1280, crop_mode=False	400 vision tokens	高精度
Gundam	base_size=1024, image_size=640, crop_mode=True	动态分辨率	复杂文档

步骤2: 运行单图像OCR

```
cd DeepSeek-OCR-master/DeepSeek-OCR-vllm
python run_dpsk_ocr_image.py
```

步骤3: 查看结果

运行完成后，输出目录将包含：

- `result_ori.mmd`：原始模型输出
- `result.mmd`：处理后的markdown格式
- `result_with_boxes.jpg`：带标注框的图像
- `images/`：裁剪的图像块

3.2 案例-2：PDF批量OCR实战

步骤1: 准备PDF文件

步骤2: 修改配置

```
# 修改config.py

INPUT_PATH = '/path/to/your_document.pdf'
OUTPUT_PATH = '/path/to/output'

# 配置参数调优
MAX_CONCURRENCY = 10  # 最大并发数
NUM_WORKERS = 6       # 预处理线程数
```

步骤3: 运行PDF OCR

```
python run_dpsk_ocr_pdf.py
```

步骤4: 查看结果

- `document_det.mmd`: 详细检测结果
- `document.mmd`: 清理后的markdown
- `document_layouts.pdf`: 带标注框的PDF
- `images/`: 裁剪的图像块

3.3 案例-3：模型部署为API服务

步骤-1: 修改配置

编辑 config.py 文件：

```
# 模型权重路径
MODEL_PATH = "/root/autodl-tmp/DS-OCR"  # 修改为实际模型路径

# 性能配置（根据GPU显存调整）
BASE_SIZE = 1024              # 基础尺寸
IMAGE_SIZE = 640              # 输入图片尺寸
CROP_MODE = True              # 启用裁剪模式
MAX_CONCURRENCY = 10          # 最大并发数（显存不足时降低）
NUM_WORKERS = 6               # 预处理线程数
```

步骤-2: 服务部署

```
python start_api.py --host 0.0.0.0 --port 6006 --workers 1

# 参数说明：
# --host：服务主机地址
# --port：服务端口
# --workers：工作进程数（GPU服务建议为1）
```

步骤-3: API测试

```
# 测试图片
python test_api.py --url http://localhost:6006 --image /root/autodl-tmp/images/1.png

# 测试PDF
python test_api.py --url http://localhost:6006 --pdf /root/autodl-tmp/pdfs/1.pdf
```

四、核心脚本

- **主要脚本文件**

脚本文件	路径	作用	主要功能
`run_dpsk_ocr_image.py`	`/DeepSeek-OCR-vllm/`	单张图片OCR处理	• 图片加载和EXIF校正 • 流式推理输出 • 自动绘制检测框 • 支持几何结构图生成 • 输出markdown格式
`run_dpsk_ocr_pdf.py`	`/DeepSeek-OCR-vllm/`	PDF文档批量处理	• PDF转高质量图片(144 DPI) • 多线程并行预处理 • 批量推理所有页面 • 生成带标注框的PDF • 支持跳过重复页面
`run_dpsk_ocr_eval_batch.py`	`/DeepSeek-OCR-vllm/`	批量评估脚本	• 批量处理多张图片 • 公式清洗和文本整理 • 生成评估结果 • 基准测试支持
`run_dpsk_ocr.py`	`/DeepSeek-OCR-hf/`	HuggingFace推理	• 使用Transformers库推理 • 直接调用HuggingFace模型 • 简化推理流程

- **API服务相关**

脚本文件	作用	主要功能	接口端点

`api_server.py`	FastAPI服务器	• 图片OCR接口
		• PDF OCR接口
		• 异步/同步推理引擎
			`/api/ocr/image`
			`/api/ocr/pdf`
			`/health`
			`/api/clear-memory`

`start_api.py`	API服务启动脚本	• 配置主机和端口 • 设置工作进程数	命令行参数配置
`test_api.py`	API测试客户端	• 发送图片/PDF请求 • 验证API响应	测试工具

- **配置和工具**

文件	作用	配置内容
`config.py`	统一配置文件	• 模型路径设置 • 输入输出路径 • 推理参数(尺寸、并发数) • 提示词模板 • 5种模式配置

- **处理模块**

模块	文件	作用	主要功能
图像处理	`process/image_process.py`	图像预处理和特征提取	• 图片分割和裁剪 • 多尺度处理 • 图像编码和token化
文本处理	`process/ngram_norepeat.py`	防止重复生成的文本处理器	• N-gram去重 • 提高生成质量 • 白名单token支持

- **编码器模块 (deepencoder/)**

文件	作用	功能
`sam_vary_sdpa.py`	SAM视觉编码器	实现SAM (Segment Anything Model) 视觉编码
`clip_sdpa.py`	CLIP视觉编码器	实现CLIP视觉编码功能
`build_linear.py`	线性投影层	构建多模态投影层

- **常用提示词**

提示词	用途	示例
`<image>\n<|grounding|>Convert the document to markdown.`	文档结构化	学术论文、报告
`<image>\n<|grounding|>OCR this image.`	普通图片OCR	简单文字识别
`<image>\nFree OCR.`	无结构自由OCR	自由格式识别
`<image>\nParse the figure.`	解析图表	图表、公式识别
`<image>\nDescribe this image in detail.`	图像描述	详细描述生成

AI进阶学习

📺 **B站课堂**：https://space.bilibili.com/3546748815411393/pugv

🌐 **官方网站**： https://www.tgltommy.com/ (国内访问需科学上网)

如果你正在关注大模型 Agent、强化学习后训练、RLHF、DPO、GRPO、RLVR 等前沿方向，欢迎关注我最新上线的精品课程：
《大模型 Agent 强化学习实战：从后训练、奖励设计到工业级对齐系统》

这门课程围绕当前大模型 Agent 强化学习的核心技术路线展开，不只是介绍零散算法和论文概念，而是希望帮助学习者建立一套完整的技术地图：从基础原理、论文脉络、关键算法，到开源项目源码解析与工业级系统实践，逐步理解 Agent RL 为什么重要、如何演进，以及在真实应用中如何落地。

课程配套专属飞书知识库 《Agent RL 学习宝典》，包含学习路线、论文资料、项目说明、源码阅读辅助资料和后续持续更新内容，方便长期查阅和跟进前沿技术。

Agent RL 变化很快，但越是变化快，越需要一套系统的学习框架。如果你希望系统进入 Agent RL 方向，而不是停留在碎片化了解阶段，这门课会是一个很好的起点。

课程官网：https://www.tgltommy.com/p/agent-rl（国内访问需科学上网）
B站课堂：https://www.bilibili.com/cheese/play/ss842375604

更多课程详情，扫描以下二维码咨询课程助理：

B站/公众号/YouTube：唐国梁Tommy

关注微信公众号
获取更多学习资源

官方网站：TGLTommy.com(需科学上网)