main.py 说明文档

1 算法概述

本算法提出了一种基于 DualFace-RCSD (Dual-ResNet Face Real-Synthetic Cascaded Detector)AI 生成人脸图像识别算法。采用两个 ResNet50 网络作为 backbone：一个处理 MTCNN 裁剪后的人脸图像，另一个处理原始图像。算法利用 ResNet50 提取深度学习特征，融合了边缘检测、色彩分布等额外特征，通过将两个 ResNet50 网络预测结果的置信度以及多维特征进行融合，最终使用决策树分类器实现对 AI 生成人脸的精确识别。同时为降低内存使用，算法采用了**临时文件存储**机制管理中间计算结果。

2 主要功能模块

2.1 MTCNN 人脸处理和图像预处理

在进行图像处理前，通过’setup_device()’ 函数检测并设置可用的计算设备，以支持 GPU 或 CPU 运算。在人脸处理阶段，利用’mtcnn_process()’ 和’crop_face()’ 函数实现基于 MTCNN 的人脸检测和区域裁剪，其中裁剪过程支持 0.3 的边界扩展，处理结果暂存于临时目录。最后通过’resize_image()’ 和’process_single_image()’ 函数进行图像预处理，将图像统一缩放至 512x512 尺寸，并支持 CUDA 加速以提高处理效率。整个过程的函数与功能直观展示在表格1中：

表 1　模型函数与功能总览

处理过程	设备设置	MTCNN 人脸处理	图像预处理
函数	setup_device()	mtcnn_process(), crop_face()	resize_image(), process_single_image()
功能	检测并设置可用的计算设备 (GPU 或 CPU)	使用 MTCNN 检测人脸 对最大人脸区域进行裁剪，支持 0.3 的边界扩展 处理结果保存至临时目录	统一缩放图像至 512x512 支持 CUDA 加速图像处理 中间结果保存至临时目录

2.2 特征提取

在特征提取阶段，算法通过边缘检测、平滑度分析、颜色分布、亮度分布和锐度图计算等多维特征提取方法，实现了对图像特征的全面捕捉。以下是所使用的函数：

- edge_detection(): 使用 Canny 算法提取边缘特征，阈值为 100-200
- smoothness_coefficient(): 使用 Laplacian 算子计算图像平滑度的方差
- color_histogram(): 计算 4x4x4 的 RGB 颜色直方图，并进行归一化
- brightness_histogram(): 计算 200bin 的灰度直方图，并进行归一化
- calculate_sharpness_map(): 计算 Laplacian 算子的归一化锐度图
- rgb_color_distribution(): 计算 RGB 三通道的均值和标准差
- hsl_color_distribution(): 计算 HSL 空间的颜色分布特征

2.3 模型架构

模型采用 CustomResNet 类实现特征提取，以配置优化的 ResNet50 网络为核心，接收 512x512 的 5 通道输入（RGB 和边缘、锐度特征图）。模型通过修改 ResNet50 的首层卷积以适应 5 通道输入，并创新性地将 ResNet 特征与手工设计的特征（平滑度、颜色和亮度直方图等）进行融合，通过 BatchNorm1d 实现特征标准化，最终输出 ResNet 特征与 265 维标量特征的组合表示。

表 2　模型架构类与功能

类	CustomResNet
ResNet50 配置	• 输入尺寸: 512x512 • 输入通道: 5 (RGB + 边缘图 + 锐度图) • 基础网络: ResNet50 预训练模型 • 第一层: 修改为 5 通道卷积 • 原始全连接层: 替换为 Identity 层 • 特征维度: 2048 • 分类输出: 7 类
特征融合	• 结合 ResNet 特征和手工特征 (平滑度、颜色直方图、亮度直方图) • 使用 BatchNorm1d 对融合特征进行标准化 • 总特征维度: ResNet 特征 + 265 维标量特征

3 算法流程

我们的算法首先进行计算设备初始化、临时目录创建和模型加载，包括预训练 ResNet50 模型和决策树分类器。对于单个的输入图像，首先通过 MTCNN 进行人脸检测和裁剪（边界扩展 0.3），并将图像统一缩放至 512x512 尺寸。随后进行初步特征提取，包括图像归一化、边缘图和锐度图提取，以及平滑度、颜色分布、亮度分布等标量

特征的计算。接着利用两个 ResNet50 网络分别处理 MTCNN 裁剪人脸图像和原图，提取深层特征并获得预测置信度。最后将所有特征进行融合和标准化，通过决策树分类器进行预测，完成分类后清理临时文件并输出结果，具体流程图如图1所示。

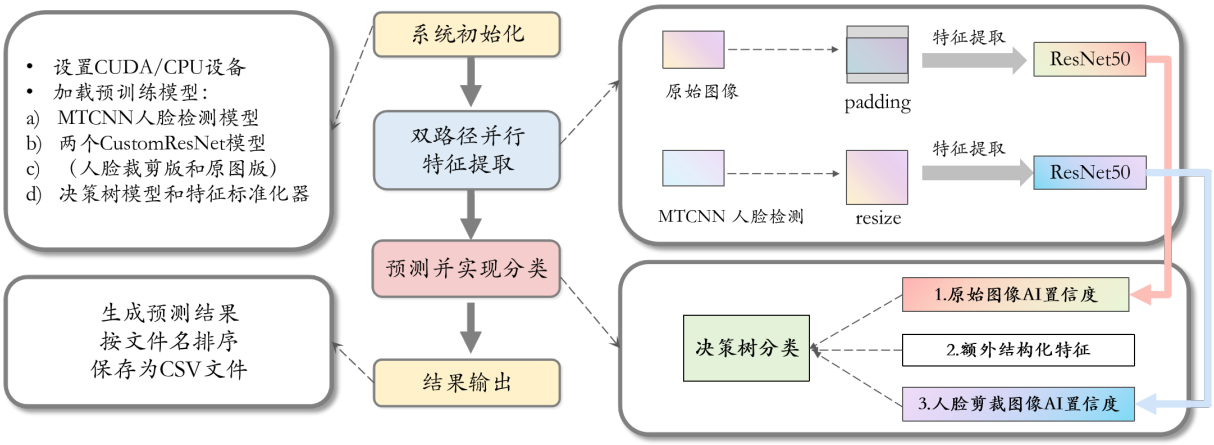

图 1　算法流程图

4 系统使用说明

接下来展示 main.py 文件的系统使用手册，详细说明了系统的输入输出要求、环境配置和操作流程。系统以双 ResNet50 模型和决策树分类器为核心，需要配套的预训练模型文件和特征标准化器，可以对指定目录下的图像进行真实与 AI 生成的图像二分类预测，最终输出 CSV 格式的预测结果。

各部分的输入输出以及所需依赖库要求展示在下表3中：

表 3　输入、输出和依赖库

输入	输出
ResNet50 模型：MTCNNresize_ResNet50_best_val_model.pth ResNet50 模型：reMTCNN 原图 padding_ResNet50_best_val_model.pth 决策树模型：best_decision_tree_model.pkl 特征标准化器：feature_scaler.pkl 测试图像目录：/testdata	CSV 文件：cla_pre.csv 格式：[图像名称 (不含扩展名), 预测结果 (0/1)]
依赖库要求	
python3.10.8　PyTorch　OpenCV-Python(CUDA 支持)　NumPy　Albumentations facenet-pytorch　scikit-learn　PIL　tqdm　joblib　pandas	

系统运行依赖 PyTorch、OpenCV-Python（支持 CUDA）等多个 Python 库，使用时需要按照指定步骤进行环境配置和模型文件准备。并且具有多项性能优化特性，支持多

种常见图像格式，并针对内存占用、处理速度等方面进行了优化，包括临时文件管理、CUDA 加速和批处理等特性。如表4所示。

表 4　使用说明和性能优化特性

使用说明	性能优化特性
1. 安装所需依赖库（pip install -r requirements.txt） 2. 读取预训练模型文件： MTCNNresize_ResNet50_best_val_model.pth reMTCNN 原图 padding_ResNet50_best_val_model.pth best_decision_tree_model.pkl feature_scaler.pkl 3. 设置图像路径和输出路径： image_folder = ”/testdata” output_file = ”./cla_pre.csv” 4. 运行脚本	• 临时文件管理降低内存占用 • CUDA 加速支持（适用于图像处理和模型推理） • tqdm 进度条可视化处理进度 • 批处理优化 • 完善的异常处理机制 • 自动资源清理

为确保系统稳定运行，手册还提供了详细的异常处理机制说明，包括错误捕获、日志记录和资源清理等内容，可帮助用户有效处理运行过程中可能遇到的各类问题，如表5所示。

表 5　注意事项和异常处理

注意事项	异常处理
• 需要充足磁盘空间存储临时文件 • 支持的图像格式：PNG, JPG, JPEG, BMP, GIF • 程序会自动创建和清理临时目录 • 预测结果说明： - 0：真实图像（真人、卡通、素描） - 1：AI 生成图像（SD 生成、StyleGAN 生成、AI 素描）	• 图像处理全流程异常捕获 • 详细的错误日志记录 • 确保临时文件的清理（即使发生错误） • 空值处理和类型检查