赛道一：电表读数识别

1. 整体设计思路

针对电表读数识别这一复合型视觉任务，我们采用了 **“先检测，后识别”** 的两阶段级联方案。该方案将问题分解为两个独立的子任务，每个子任务都可以选用最优的模型来解决，从而提升整体的鲁棒性和准确性。

1. **阶段一 读数区域检测：**在输入的完整电表图片中，精确定位包含所有数字滚轮的矩形区域。
2. **阶段二 数字序列识别：**对阶段一裁剪出的读数区域图像，进行文字识别，将其转换为数字字符串。
3. **阶段三 后处理：**根据电表读数的业务规则，对识别出的数字字符串进行格式化处理。

2. 模型选型与实现细节

2.1. 阶段一：读数区域检测 (YOLOv8)

- **模型选型**：我们选用 **YOLOv8** 作为基础目标检测模型。我们具体使用了 `yolov8s.pt` 版本。
- **训练策略**：采用迁移学习的方式进行微调 。我们利用 `yolov8s.pt` 模型作为起点，然后在我们自己标注的电表数据集上进行训练。这使得模型能够快速学习到电表读数区域的特定特征。
- **实现细节**：模型会对输入图片进行预测。为应对可能出现的多个候选框，我们选择置信度最高的检测结果作为最终的读数区域。该阶段的输出是读数区域的边界框坐标 `(x1, y1, x2, y2)`。

2.2. 阶段二：数字序列识别 (PP-OCRv5)

- **模型选型**：我们选用 **PP-OCRv5_server_rec**作为基础识别模型。
- **训练策略**：同样采用迁移学习策略。使用从阶段一裁剪出的电表读数图片及其对应的真实数字标签进行微调。使模型能够适应电表特有的数字字体、字符间距、滚轮排列以及光影等特征。
- **数据集与关键配置**：训练的图片均为已裁剪好的读数区域，并统一缩放至 `[3, 48, 320]` 的尺寸。同时我们提供了一个自定义字典，其中仅包含字符 '0' 到 '9'。这极大地缩小了模型的预测空间，有效避免了将数字误识别为字母或其他符号（如 'O' 识别为 '0'，'l' 识别为 '1'），显著提升了识别准确率。并且我们使用了 `SVTR_HGNet` 架构，并配置了 `MultiHead` (CTCHead + NRTRHead) 和 `MultiLoss` (CTCLoss + NRTRLoss) 策略，能够进一步提高识别的鲁棒性。
- **实现细节**：输入是阶段一输出的裁剪图片。输出是识别到的原始数字字符串，例如 "084306"。

2.3. 阶段三：后处理

识别出的原始字符串 "084306" 并不能直接作为最终结果。根据电表读数的通用规则（最后一位为小数），进行简单的字符串处理。如果识别结果以 '0' 开头 (且不为单个 '0')，则移除。例如 "084306" -> "84306"。并且在字符串的倒数第二位和最后一位之间插入一个小数点。

3. 实验流程

1. **数据准备**：生成YOLO格式的标签文件以及具体读数的标签文件。按比例划分训练集和验证集。
2. **模型训练**：分别执行YOLO和Paddle的训练脚本，对两个模型进行微调，并保存效果最佳的模型 。
3. **集成与推理**：执行最终的推理脚本 (`inference.py`)。该脚本会加载上述两个训练好的模型。对于每张待测图片，脚本会执行完整的“检测->裁剪->识别->后处理”流水线。