**OCR日期识别使用说明文档**

1. **实验环境搭建**
   1. **OCR项目代码下载地址：**<https://github.com/18721017183/OCR>
   2. **安装Anaconda3**

我们的环境搭建主要在anaconda3中进行。首先登陆打开anaconda3的官方网站（ <https://www.anaconda.com/products/individual>），选择适合自己的操作系统的版本下载并安装完成（参考网址：<https://repo.anaconda.com/archive/Anaconda3-2020.07-Windows-x86_64.exe>）

* 1. **需求包的安装**
* Python >= 3.6

在使用anaconda3创建环境时选择版本大于3.6即可

* Torch >= 1.5

登录Pytorch官网，根据自己电脑内的CUDA版本，选择相应的命令进行安装。（官网链接：<https://pytorch.org/>）

* Opencv-python >= 3.4

命令：pip install opencv-python

* Onnx >= 1.7

命令：python install onnx

* Onnxruntime-gpu >= 1.2

命令：python install onnxruntime-gpu

1. **制作样本**

本次使用的样本数据由三部分组成

1. 代码生成图片，约两万
2. 二值化后字母合成图片，约两万
3. 没有二值化的字母合成图片，约一万
   1. **代码生成图片**

生成所需代码下载链接<https://github.com/18721017183/text_renderer-master>

* 1. 进入text\_renderer-master所在文件夹
  2. 训练相关参数存放于parse\_args.py中，根据需要修改参数，如：生成图片数量，图片存放目录等
  3. 代码中采用msyh.ttc、Ventouse-1.ttf、TLOZ-Phantom-Hourglass-1.ttf、ProggyTinySZ-4.ttf 四种字体。在训练测试过程中，发现当采用单一字体训练时，针对个别字符效果不理想。将多种字体融合后，对所有日期字符效果均有提升。比较多种测试结果后，现采用上述4中字体来生成图片。这些字体均从“字体天下”网下载。（<http://www.fonts.net.cn/>）运行main.py。生成样本，以及对应的标注文件tmp\_labels.txt
  4. 返回OCR项目（在1.1中下载的ORC-master），运行“./ocr\_uits/样本标注\_注释.py”，将标注文件tem\_labels.txt转化为模型训练所需格式char.txt，之后自行修改文件名至char\_train.py和char\_test.py，放在./data/lv\_generate\_txt文件夹中。
  5. **使用包装袋上二值化字母生成图片**

用包装袋上的字符生成样本的原因主要有三点：1.如果仅仅采用代码生成的图片来训练模型，训练过程中很难收敛。2.因为没有用到实际在包装袋上出现的字符，效果也不太理想。3.实际能获取的包装袋的日期就那么几个，很有限。故利用包装袋上日期切分下来的字符，自己合成样本。采用二值化的原因是，过滤掉实际背景的影响。适用更多的场合。

生成过程如下：

1. 通过labelImg标注样本图片，生成xml。目的是将包装袋上的日期字符一个字、一个字切分下来；
2. 进入ocr\_utis,运行"XML数据解析2.py"，生成各个字符的图片，每类字符单独存一个文件夹，文件夹名即为字符名；
3. 如果是生成二值化的文字，则运行“图片背景颜色更换.py”；
4. 运行“OCR图片合成.py”，合成图片，并保存标签；
5. 运行“样本标注\_解析.py”，将标签转换为符合模型的要求格式；
   1. **使用包装袋上未二值化字母合成图片**

原因和二值化后字母生成图片的原因一致。没有二值化的原因是针对当前的场景，用同样的背景来训练，对日期OCR识别效果有一定提升。二值化和没有二值化两种都用到，目的就是为了适用更多的场合。

额外添加图片

针对“0”（与字母O相似），“1”（与字母I相似），“/”，“-”（/，-字符简单，容易被噪声干扰）等字符，由于相对来说比较容易识别出错，故针对这几个字符，增加相应样本，以提高其预测的准确率。

1. 通过labelImg标注样本图片，生成xml。目的是将包装袋上的日期字符一个字、一个字切分下来；
2. 进入ocr\_utis,运行"XML数据解析2.py"，生成各个字符的图片，每类字符单独存一个文件夹，文件夹名即为字符名；
3. 运行“OCR图片合成.py”，合成图片，并保存标签；
4. 运行“样本标注\_解析.py”，将标签转换为符合模型的要求格式；
   1. **如何将所有的样本混合到一个样本**

**手工复制粘贴（名称可能重合，）**

1. **模型训练**
2. 将上述生成的图片及标签文件放入“./data”文件夹下
3. 修改“config.py”中对应的参数
4. 运行“train\_win\_20200624.py”，训练结束后，生成对应onnx模型和pth模型。
5. **模型预测**
6. 进入“OCR”，（https://github.com/18721017183/OCR）
7. 进入“onnx\_单张\_多张图片预测\_main.py”,修改要预测的图片路径和生成的onnx模型路径