**青苗口腔人工智能自检微信系统**

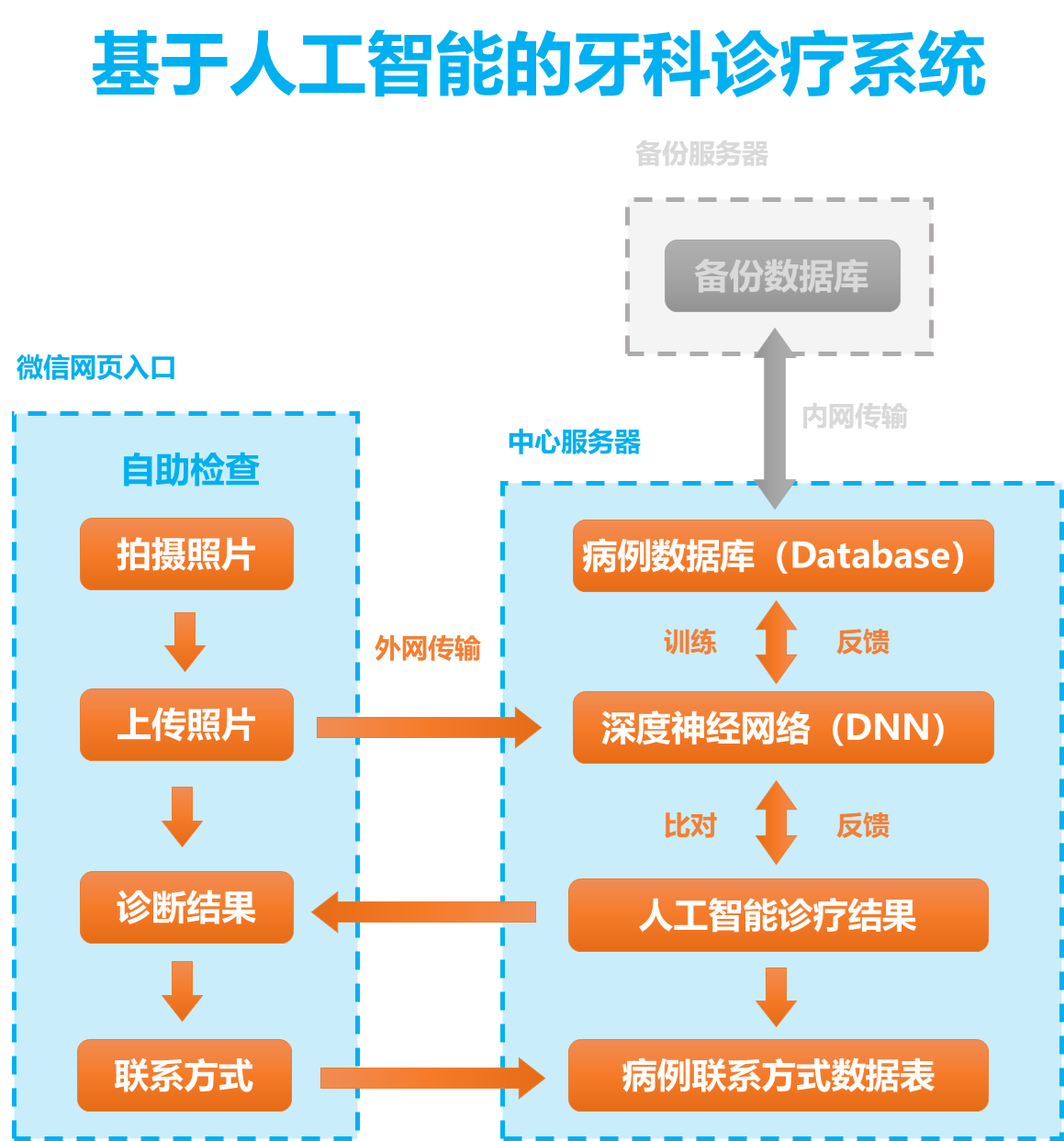
**——管理操作手册**

1. **系统功能简介：**

青苗口腔人工智能自检微信系统（以下简称AI自检系统）是一个使用Express框架，并基于node js 开发的web应用，该项目可部署在普通Linux/UNIX环境云服务器上使用，对患者上传的照片，实时进行判断是否有口腔疾病。并可不断记录患者上传过的照片，以便日后使用这类图片作为数据源继续训练模型。AI自检系统所使用的神经网络模型以graph.pb, label.txt的文件格式存放在云服务器上，可以在未来修改与训练，并可以随时替换。

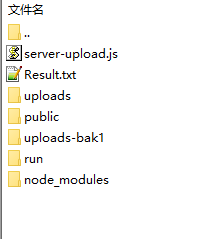
为了使改项目能够正常运行，服务器需要安装好依赖的软件环境，包括node.js，python(3.5+)，以及python内的软件包，如tensorflow等。具体步骤见第3部。

1. **系统工作示意图：**

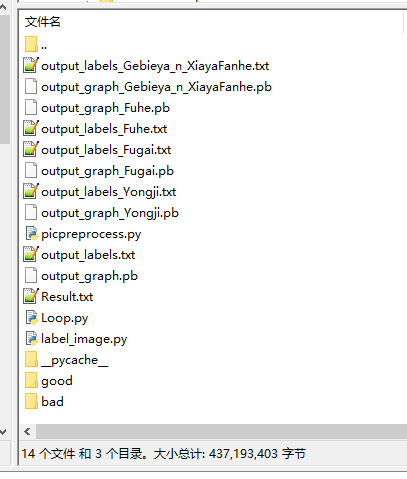


1. **操作相关步骤：**
2. **训练数据准备：xxx数据，如何分类**
3. **系统准备Installation 步骤：把附录中的一些内容摘到这里。**
4. **系统训练/再训练步骤：把附录中的一些内容摘到这里。**
5. **系统文件说明：**

文件所在地和目录：



与照片分类，牙齿病情诊断相关的神经网络模型文件在run文件夹下，进入run文件夹。



其中output\_graph\_\*\*\*.pb 与output\_labels\_\*\*\*.txt 文件一共有四对，分别对应四类病情的诊断（牙拥挤，深覆盖，深覆合，个别牙和前牙反颌）。

这几个模型在服务器运行过程中分别被label\_image.py 调用，并先使用picpreprocess.py对用户上传的三个角度的牙齿照片进行预处理，之后即进行分类，其中牙拥挤模型使用第一步照片（张嘴下牙照），个别牙和前牙反颌模型及深覆合模型使用第二步照片（正面露牙照）， 牙覆盖模型使用第三步照片（侧脸照），并都给出各自的分类结果，其结果即对应output\_labels\_\*\*\*.txt 中的种类，表示是否为健康或是有病，并最终传递给前台页面显示在检测结果里。

管理员如果现在以后继续训练模型提高准确率，则可以参照附录一里描述的方法重训练获得新的output\_graph.pb 与output\_labels.txt模型文件，即可替换现有的模型文件完成升级。

1. **系统测试大量数据步骤：加入xxx**

**附录一： 模型介绍**

目前项目中使用了inception V3 神经网路作为基本模型，该模型是由Google及相关研究人员开发的用于图片识别卷积神经网络模型，该模型是基于2012年imageNet大型视觉认知挑战的数据训练的，其试图实现的任务即是将整个训练用数据图片分类至1000个类别中，例如”斑马”，“ 达尔马提亚狗”，以及“洗碗机”。而最终该模型在校验数据中获得了3.46% 的“前5次猜测” 的错误率。

在本AI自检系统中，我们将该inception V3神经网络进行了微调以实现对于牙齿照片的分类。而所谓的“微调“，即是在保留该神经网络其他层结构与权重不变的基础上，重新训练其最终层与输出层权重的方法，该方法可以在最大保留原有训练成果的基础上，最快的将该模型移植到我们的项目应用中来。目前我们微调后得到的神经网络模型在牙齿拥挤的诊断上最高可以实现85%以上的准确率。

**训练过程**

在目前的模型训练过程中，我们使用了Google开发的tensorFlow软件库与一系列基于python实现的神经网络重训练代码。训练时，人工标记好的数据首先被分成了3部分，分别用于模型的训练，校验与最终的测试。其中校验与测试的数据各占10%，而训练的数据为80%。训练开始后，训练数据通过模型获得模型的输出结果，然后根据输出结果与实际结果的差异使用Stochastic gradient descent（SGD）的方法调整其权重，调整之后再次将训练数据（Train accuracy 即为模型在训练数据的表现）输入并比较结果差异（Cross entropy），如此反复训练并在每次循环中使用校验数据（Validation accuracy 即模型在校验数据的表现）来监控模型的性能变化。训练结束后，测试数据将用来测试该模型的最终表现。模型如果在校验数据（Validation accuracy）与测试数据（Final test accuracy 即模型在最终测试数据的表现）中都获得较好的表现，训练便成功而该模型既可部署至服务器端作为新采集图片的分类器。

**附录二： 系统结构**

业务逻辑层

数据与神经网络层

表现层

html页面，网页图片样式等文件 (server/public/)

js内实现的点击响应，页面跳转与数据处理，结果返回等逻辑 (server/server-upload.js)

用户照片数据存储与调用神经网络模型对照片进行诊断，并返回分类结果 (server/uploads, server/run/)

**附录三： 模型的再训练**

模型的再训练其实即是用新数据训练output\_graph\_\*\*\*.pb 与output\_labels\_\*\*\*.txt的过程。在经过一段时间积累新的照片数据后，便可将这些新数据按照各自的标签加到原有的数据集中，与原数据一起使用训练新的模型文件。使模型更加精确的有效一个方法就是使用更大量的数据。

新模型的训练首先需要安装好python 与Tenserflow，对于64位的Windows操作系统建议从官网下载并安装python-3.6-amd64.exe，安装好后可以在命令行内测试：



安装python的过程中需要勾选安装pip，以方便下一步使用pip安装tensorflow包。pip是否安装成功也可以在命令行中测试：



之后便可以使用pip安装tensorflow：

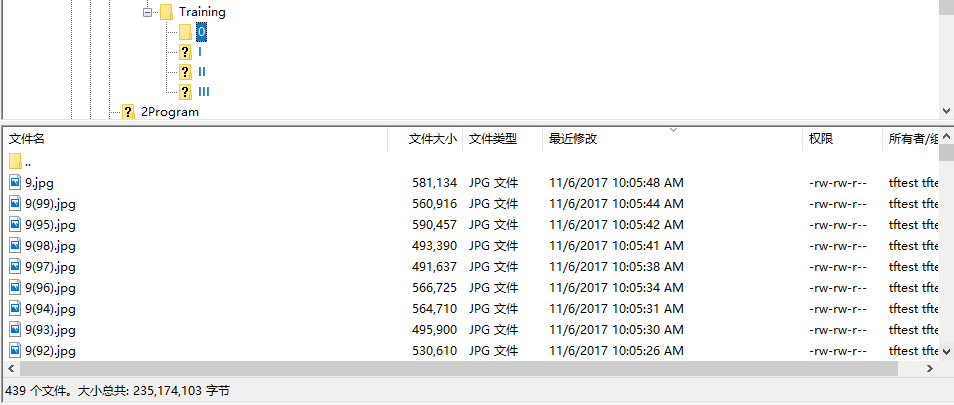


如果安装顺利通过，则完成所有环境的配置，下一步就需要整理好训练数据的目录。

训练数据的目录需要如下图布置：



其中Training文件下包含了所有照片的分类，即一个文件夹对应一个分类（分类取名可以随意），而各分类文件夹下则包含了改类的所有照片数据。



设置好数据目录后，即可运行一段python代码开始训练：

python retrain6.py --how\_many\_training\_steps=1000 --image\_dir=C:\QingMiao\6DevelopmentFiles\1Data\908\_1000Train\Front6 --summaries\_dir=C:\tmp\OralCareImage\test

这段命令中包含三个参数，其中“--how\_many\_training\_steps” 设定了整体训练的循环次数，循环的次数越久，模型训练的越充分，但当预测结果与实际结果的差异值的梯度下降至对象值或接近最小值时，就没有必要继续训练了。“--image\_dir” 参数定义了训练数据的路径，在上例中即为training文件夹的位置。“--summaries\_dir” 参数定义了输出结果的位置，其默认输出模型文件即为output\_graph.pb 与output\_labels.txt，重新训练过程中，只需要将这两个文件替换项目中所用的原pb/label.txt文件即可完成模型的替换。