# 性能异常检测与模型训练基础流程

## **运行性能异常检测APP进行异常检测**

异常检测APP运行在Android 12版本上，将APP项目UpdateMonitor导入Android Studio中，在Android Studio Terminal打开项目根目录，运行gradlew assembleDebug命令在UpdateMonitor\app\build\outputs\apk\debug目录下生成Debug版本的APK，然后通过ADB命令将APK发送到Oneplus9Pro手机上安装，或者连接USB后直接在Android Studio中点击Run实现编译安装。

Python脚本实现APK编译并安装到手机上：

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  s = input("输入UpdateMonitor项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\UpdateMonitor：")  os.system("cd " + s + "&& gradlew assembleDebug && adb install " + s + "\\app\\build\\outputs\\apk\\debug\\AnomalyDetection.apk") |

打开APP界面如下，程序运行主界面如下图1，打开主界面首先显示当前IP,使用USB连接手机与电脑，在cmd窗口执行adb root以及adb tcpip 5555命令，然后点击确定进入主界面，点击启用监控，会弹出提示框选择预测方式，使用本地性能检测或者服务器性能异常检测或者不使用检测（只进行监控），点击确定后即可进入性能检测界面，进行性能检测。点击关闭监控会结束当前监控以及异常检测。

图形用户界面

低可信度描述已自动生成图形用户界面, 应用程序

描述已自动生成图形用户界面, 应用程序, Teams

描述已自动生成图形用户界面, 应用程序

描述已自动生成

(a) (b) (c) (d)

图1 程序运行主界面

当使用云端检测时，需要修改UpdateMonitor源代码中的服务器IP地址，两者需要在同一局域网下，修改服务器IP地址在UpdateMonitor项目ADBLIB.java中的private class RunnableTask中，位于文件第903行，然后重新编译安装APP项目，在服务器端运行Server/Server.py程序，启用APP的云端检测即可进行远程服务器检测。

## 异常检测模型的训练

整体模型训练流程遵循：**正常数据采集——异常数据采集——模型训练与保存——格式转化——模型在APP中的使用**几个步骤，下面将进行逐一说明：

### 正常数据采集

对于正常数据的定义与异常检测模型有关，比如对于CPU异常检测模型，GPU异常与内存异常不算异常，而算作正常数据，只有CPU异常才算异常，相应地在采集时可以选择是否开启GPU与内存异常。对于GPU与内存正常数据的采集亦是如此。

在不开启对应异常模拟程序而且的确不存在对应异常时进行正常数据的采集（如果需要采集某一特定前台应用的异常检测模型的正常数据，需要保持监控时前台应用为该应用，如果需要采集整机异常检测模型的正常数据，前台应用可以改变），打开性能异常检测APP并在运行adb root以及adb tcpip 5555之后点击开始监控按钮，选择不进行性能异常检测，开始进行正常数据集的采集。采集到的文件数据保存在data/data/com.wwh.updatemonitor/files/result.csv中，注意如果不想保留之前的历史数据，则需要在开始监控前删除result数据，cmd窗口执行以下删除命令：

|  |
| --- |
| adb shell rm data/data/com.wwh.updatemonitor/files/result.csv  adb shell rm data/data/com.wwh.updatemonitor/files/result.txt |

每次完成采集之后点击UpdateMonitor将result.csv文件pull到计算机训练模型项目performance\_Decision的相应目录下，使用的python脚本如下，对于非淘宝、抖音、原神的应用一律放在整机异常检测模型训练的目录下，文件最后命名为Normal.csv放在performance\_Decision/前台应用/异常检测类型/normal下，Python脚本如下：

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  s = input("输入performance\_Decision项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\performance\_Decision：\n")  y = int(input("当前采集正常的环境属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("当前采集的正常数据适用模型，请输入数字：1：CPU异常检测，2.GPU异常检测，3.内存异常检测\n"))  x1 = ['', 'taobao\_data', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpuData', 'gpuData', 'memData']  os.system("adb pull data/data/com.wwh.updatemonitor/files/result.csv " + s + "\\" + x1[int(y)] + "\\" + x2[int(z)] + "\\normal")  os.system("cd " + s + "\\" + x1[int(y)] + "\\" + x2[int(z)] + "\\normal && rename result.csv Normal.csv") |

### **异常数据采集**

对于异常数据的定义也与异常检测模型有关，比如对于CPU异常检测模型，GPU异常与内存异常而CPU正常的数据不算异常，而算作正常数据，只有CPU异常才算异常，相应地在采集时开启CPU异常，可以选择是否开启GPU与内存异常。对于GPU与内存异常数据的采集亦是如此。

在开启对应异常模拟程序而且的确存在对应异常时进行正常数据的采集（如果需要采集某一特定前台应用的异常检测模型的异常数据，需要保持监控时前台应用为该应用，如果需要采集整机异常检测模型的异常数据，前台应用可以改变），打开性能异常检测APP并在运行adb root以及adb tcpip 5555之后点击开始监控按钮，选择不进行性能异常检测，开启对应异常，进行异常数据集的采集。采集到的文件数据保存在data/data/com.wwh.updatemonitor/files/result.csv中，注意如果不想保留之前的历史数据，则需要在开始监控前删除result数据，cmd窗口执行以下删除命令：

|  |
| --- |
| adb shell rm data/data/com.wwh.updatemonitor/files/result.csv  adb shell rm data/data/com.wwh.updatemonitor/files/result.txt |

每次完成采集之后点击UpdateMonitor将result.csv文件pull到计算机训练模型项目performance\_Decision的相应目录下，使用的python脚本如下，对于非淘宝、抖音、原神的应用一律放在整机异常检测模型训练的目录下，文件最后命名为Normal.csv放在performance\_Decision/前台应用/异常检测类型/abnormal下，Python脚本如下：

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  s = input("输入performance\_Decision项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\performance\_Decision：\n")  y = int(input("当前采集异常的环境属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("当前采集的异常数据适用模型，请输入数字：1：CPU异常检测，2.GPU异常应用，3.内存异常应用\n"))  x1 = ['', 'taobao\_data', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpuData', 'gpuData', 'memData']  os.system("adb pull data/data/com.wwh.updatemonitor/files/result.csv " + s + "\\" + x1[int(y)] + "\\" + x2[int(z)] + "\\abnormal")  os.system("cd " + s + "\\" + x1[int(y)] + "\\" + x2[int(z)] + "\\normal && rename result.csv Abnormal.csv") |

### **模型训练与保存**

首先在VScode或者Pycharm中打开performance\_Decision项目，再根据需要训练的模型进入待训练模型的对应目录，如训练前台为淘宝的cpu异常检测模型需要进入performance\_Decision/taobao\_data/cpuData目录中，然后右键运行main.py对数据进行预处理并训练异常检测模型，最后模型在对应的res目录中保存为pkl模型文件以及pmml模型文件。训练使用Conda虚拟环境并安装了相应的库：

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  # 虚拟环境名称  env\_name = "AnomalyDetection"  # 检查是否存在虚拟环境  command = f"conda env list | findstr {env\_name}"  output = os.system(command)  print(output)  # 如果虚拟环境不存在，则创建新的虚拟环境  if output != 0:  create\_command = f"conda create --name {env\_name} python=3.7 && conda activate {env\_name} && pip install scikit-learn==0.23.1 && pip install sklearn2pmml==0.63.1 && pip install joblib==0.16.0 && pip install matplotlib && pip install ipython"  os.system(create\_command)  s = input("输入performance\_Decision项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\performance\_Decision：\n")  y = int(input("当前采集异常的环境属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("当前采集的异常数据适用模型，请输入数字：1：CPU异常检测，2.GPU异常应用，3.内存异常应用\n"))  x1 = ['', 'taobao\_data', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpuData', 'gpuData', 'memData']  os.system(f"conda activate {env\_name} && cd " + s + " && python " + x1[int(z)] + "\\" + x2[int(y)] + "\\main.py") |

### **模型格式转化**

由于上述生成的pmml格式部署到手机上需要进行模型格式转化，相关的方法利用到pmml-android项目，即将pmml模型文件序列化为ser文件后在Android设备上执行，具体转化步骤如下：

(1)将生成的pmml文件中的xml头部的PMML-Version以及jpmml-model版本化为一致的4.3版本：

即将：

*<?xml version="1.0" encoding="UTF-8" standalone="yes"?>*

*<PMML xmlns="http://www.dmg.org/PMML-4\_4" xmlns:data="http://jpmml.org/jpmml-model/InlineTable" version="4.4">*

修改为：

*<?xml version="1.0" encoding="UTF-8" standalone="yes"?>*

*<PMML xmlns="http://www.dmg.org/PMML-4\_3" xmlns:data="http://jpmml.org/jpmml-model/InlineTable" version="4.3">*

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  s = input("输入performance\_Decision项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\performance\_Decision：\n")  y = int(input("PMML模型属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("PMML模型目标，请输入数字：1：CPU异常检测，2.GPU异常应用，3.内存异常应用\n"))  x1 = ['', 'taobao\_data', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpuData\\res\\', 'gpuData\\res\\', 'memData\\res\\']  x3 = ['', 'cpuDecisionTreeClassifier.pmml', 'gpuDecisionTreeClassifier.pmml', 'memDecisionTreeClassifier.pmml']  a = input("输入pmml-android-master项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\pmml-android-master：\n")  # 复制模型文件到pmml-android-master  os.system("copy " + s + "\\" + x1[int(y)] + "\\" + x2[int(z)] + x3[int(z)] + " " + a + "\\pmml-android-example\\src\\main\\pmml")  # 删除原来的文件  if os.path.exists(a + "\\pmml-android-example\\src\\main\\pmml\\model.pmml"):  os.system(f"del \F {a}\\pmml-android-example\\src\\main\\pmml\\model.pmml")  os.system("cd " + a + "\\pmml-android-example\\src\\main\\pmml && rename " + x3[int(z)] + " model.pmml")  # 替换内容  file\_path = a + "\\pmml-android-example\\src\\main\\pmml\\model.pmml"  with open(file\_path, "r+") as file:  lines = file.readlines()  file.seek(0)  new\_line2 = "新的第二行内容"  lines[1] = "<PMML xmlns=\"http://www.dmg.org/PMML-4\_3\" xmlns:data=\"http://jpmml.org/jpmml-model/InlineTable\" version=\"4.3\">\n"  file.writelines(lines)  file.close() |

1. sklearn模型转换pmml格式后，需要先进行序列化操作。使用Idea打开pmml-android项目，项目编译支持Android版本3.0以上，jdk版本为版本8以下。进入根目录后，首先将模型的pmml文件放入pmml-android-example/src/main/pmml/目录下，在项目根目录下执行命令，打开Idea的Terminal终端，输入mvn clean install编译项目，这里需要已经配置好Maven仓库相关设置，项目需要在pmml-android-example的pox.xml的<properties><android.sdk.path>XXXX</android.sdk.path></properties>中的XXXX配置本地SDK路径，在<sdk><platform>XX</platform></sdk>中的XX配置sdk平台版本，可以使用最新发布的Android API Level 34,编译结束后在目录pmml-android-example/target/generated-sources/combined-assets/下能够找到生成的序列化之后的.ser文件，整个项目由于版本问题编译可能失败，但是只需用到生成的.ser文件。

### **模型在APP本地与在Server端的使用**

对于模型文件在APP本地的使用，只需将2.4节生成的pmml-android-1.0-SNAPSHOT.jar库放在之后的性能异常检测APP的UpdateMonitor的app/libs目录下，使用implementate进行引入使用，这里已经放过了。将生成的ser序列化文件重命名后放在性能异常检测APP的app/src/main/assets目录下，重新编译安装之后即可作为决策树模型作为资源文件部署到本地使用，编译安装APK步骤见第一节。

对于模型文件在云端的使用，只需将训练过程中生成的pkl文件复制到Server项目的相应目录中即可，在服务器运行Server.py程序即可实现服务器端的模型使用。Python脚本如下：

|  |
| --- |
| import os  if \_\_name\_\_ == '\_\_main\_\_':  x = int(input('请输入：云端部署——0，本地部署——1\n'))  if x == 1:  a = input("输入pmml-android-master项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\pmml-android-master：\n")  b = input("输入UpdateMonitor项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\UpdateMonitor：\n")  y = int(input("ser模型属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("ser模型目标，请输入数字：1：CPU异常检测，2.GPU异常应用，3.内存异常应用\n"))  x1 = ['', 'taobao', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpu', 'gpu', 'mem']  name = x1[int(y)] + x2[int(z)] + ".pmml.ser"  # 复制模型文件到pmml-android-master  os.system("copy " + a + "\\pmml-android-example\\target\\generated-sources\\combined-assets\\model.pmml.ser " + b + "\\app\\src\\main\\assets")  # 删除原来的文件  if os.path.exists(b + "\\app\\src\\main\\assets\\" + name):  os.system(f"del \F {b}\\app\\src\\main\\assets\\{name}")  os.system("cd " + b + "\\app\\src\\main\\assets && rename model.pmml.ser " + name)  if x == 0:  a = input("输入performance\_Decision项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\performance\_Decision：\n")  b = input("输入Server项目根路径，例如：D:\\OPPO\\stage3\\code\\performance\\Server：\n")  y = int(input("pkl模型属于哪一类，请输入数字：1：固定前台应用淘宝，2.固定前台应用抖音，3.固定前台应用原神，4.无固定前台应用\n"))  z = int(input("pkl模型目标，请输入数字：1：CPU异常检测，2.GPU异常应用，3.内存异常应用\n"))  x1 = ['', 'taobao\_data', 'douyin', 'yuanshen', 'entire']  x2 = ['', 'cpuData\\res\\cpuDecisionTreeClassifier.pkl', 'gpuData\\res\\gpuDecisionTreeClassifier.pkl', 'memData\\res\\memDecisionTreeClassifier.pkl']  x3 = ['', 'taobao', 'douyin', 'yuanshen', 'entire']  # 复制模型文件到Server  print("copy " + a + "\\" + x1[int(y)] + "\\" + x2[int(z)] + " " + b + "\\" + x3[int(z)])  os.system("copy " + a + "\\" + x1[int(y)] + "\\" + x2[int(z)] + " " + b + "\\" + x3[int(z)]) |