机器学习与动静态分析结合的安卓电量预测

梁文杰[1,2] **周扬帆**[3] **彭鑫**[1,2] **赵文耘**[1,2]
（复旦大学 软件学院 上海 201203）[1]
（上海市数据科学重点实验室 上海 201203）[2]
（复旦大学 计算机科学技术学院 上海 201203）[3]

摘 要 安卓开发者经常需要预测安卓应用的耗电功率，从而改善安卓应用的性能。针对这一目的，提出了一种动态测电和静态分析相结合的安卓应用耗电功率预测方法。该方法利用静态分析构建安卓 activity 这一安卓基础组件的特征向量，利用动态分析，使用测电仪器测出一些安卓应用的 activity 的耗电功率。基于这些数据，利用机器学习的方法得到安卓 activty 特征向量和耗电功率的预测测量。此模型可以被用于预测新应用的 activty 耗电功率。实验结果表明，此方法能够以较低的误差预测耗电功耗。

关键词 动态测电，静态分析，特征向量，预测模型

中图法分类号 TP311 **文献标识码** A

Predicting Power Consumption of Android Apps: A Machine Learn-ing-based Approach via Joint Static and Dynamic Analysis

LIANG Wen-jie[1,2] ZHOU Yang-fan[3] PENG Xin[1,2] ZHAO Wen-yun[1,2]
(School of Software，Fudan University， Shanghai 201203，China)[1]
(Shanghai Key Laboratory of Data Science，Fudan University， Shanghai 201203，China)[2]
(School of Computer Science and Technology，Fudan University， Shanghai 201203，China)[1]

Abstract Android developers often require to predict the power consumption of Android applications, so as to improve its energy efficiency. To this end, this paper proposes a power consumption predicting method for Android applications with both static and dynamic program analysis. This method conducts feature selection for Android activities, key components of Android applications, with a static analysis approach. Based on a dynamic analysis mechanism, the activity power consumption of a set of applications with a power meter is obtained. This can allow us to fit a power consumption model to the measurements with a machine learning approach. Such a model can then be employed to predict power consumption of new applications. Experimental results show that the method can predict power consumption with low error.

Keywords Dynamic power measurement，Static analysis，Eigenvector，prediction model

1 引言

如今，智能移动设备已经深深融入大众的日常生活，智能手机即是其中的代表。使用智能手机的一个巨大缺陷是手机电池电量的有限性，与家用 PC 相比，由于用户在使用手机的大多数场景中并不能保证电量的连续供应，因此，在有限的电池电量条件下，智能手机的使用时间也被限制。智能手机的使用时间受手机应用耗能影响，若用户使用耗能较大的应用，而智能手机又未能及时充电，那么手机电量很快会被耗尽，这会极

到稿日期： 返修日期：
梁文杰(1991-)，男，硕士，主要研究方向为安卓程序分析，E-mail: wjliang15@fudan.edu.cn(通信作者)；周扬帆(1979-)，男，博士，副教授，主要研究方向为云计算系统和移动计算系统及其软件性能和可靠性；彭鑫(1979-)，男，博士，教授，主要研究方向为智能化软件开发，移动云计算；赵文耘(1964-)，男，硕士，教授，主要研究方向为软件工程。

大地影响用户体验。可以预见到，在电池技术领域未能有较大革新的情况下，电池电量将仍是限制智能手机使用时间和用户体验的瓶颈。

由前所述，电池电量的有限性就给开发者带来了更大的挑战。传统意义上，开发者仅仅只需分析、获取用户需求，设计并开发出满足用户需求的应用。在用户满意度方面，仅仅需要考虑应用流畅性、易用性等特点，而在如今移动环境下，开发者开发移动设备应用时，还需要考虑移动应用的能耗。根据中文互联网数据资讯中心的2017年7月22日的一篇报告[1]，谷歌的 Android 系统份额从 2016 年的 63% 上升至 2017 年同期的 67%. 因此，本文旨在针对安卓智能设备，在安卓的 Activity 层次，为开发者提供了一种简便可行的应用能耗预测方法。

该方法所采取的基本思路是使用相关仪器，利用动态测试的方法测出安卓应用中某个 activity 运行时的耗电功率；同时利用静态分析的方法，提取出 activity 的一些属性构建其特征向量，由此对每个 activity 就构建出形如(特征向量，耗电功率)这样的一组数据。然后利用机器学习的方法，得到形如(特征向量—>耗电功率)这样的预测模型。当开发者获得这个预测模型后，只要简单的根据新的安卓应用 activity 的属性构建其特征向量，就可以预测对应 activity 的能耗，进而改进应用。

而研究工作的一大亮点是结合了动态测试和静态分析，同时又考虑到开发者可能没有电量测试仪器，无法进行动态测电的情况。开发者可以按照本研究工作的静态分析方法，构建 activity 的特征向量，同时使用本研究工作得到的预测模型，预测 activity 的耗电功率，这方便了开发者改进自己的应用。本研究的另一亮点在于静态分析中特征向量的构建，同时这也是本研究的难点：只有选取合适的属性，构建与 activity 耗电密切相关的特征向量才能实现准确的电量预测方法。

本文第 2 节介绍了有关移动应用电量测试的相关工作；第 3 节介绍了本研究的研究方法；第 4 节介绍了本研究的实验步骤；第 5 节展示了对实验结果的分析；最后一节是结束语。

2 相关工作

移动应用能耗测量的相关工作可以被划分为两大类，一类是电量预测，另一类是电量测量。电量预测没有相关硬件仪器来测量手机耗电功率，因此只能使用开源数据或静态分析方法来预测应用耗电的多少。而另一类，电量测量，在研究过程中使用电量测试仪器，从而能够通过动态测量获得移动应用的耗能情况，进而对耗能情况进行分析。本文即可归为后一类工作。

在第一类工作中，Shuai Hao 等人[2]研究出一种新颖的研究方法——eLens. eLens 着眼于细粒度的静态程序分析，基于 SEEP (Software Energy Environment Profile）提供的指令能耗函数，提出了一种细粒度的能耗预测方法。Tao Li 等人[3]及 T.K. Tan 等人[4]通过对应用进行插桩，从而监控系统层次的 API 调用个数，以此预测应用消耗电量的高低。Karan Aggarwal 等人[5]设计了一种检测安卓应用修改前后能耗改变程度的方法，他们也是通过对应用进行插桩，监控系统调用，以系统调用的多少来表现耗能多少，从而通过对比应用修改前后系统调用个数的变化来代表耗能变化。Xiao Ma 等人[6]开发出一种面向用户的电池异常耗尽检测工具——eDoctor. 此工具通过学习历史资源使用情况和电池电量耗尽事件，向用户提出避免电池电量耗尽的使用建议。这些研究工作与本文的区别在于，它们只着眼于静态分析和预测，没有结合动态测试进行验证。而单纯基于静态分析进行电量预测，而不进行动态测试验证，其得到预测模型的精准度可能较低。

在第二类工作中，Ding Li 等人[7]改进了[2]中的 eLens 方法，实现了动态测电的 vLens 工具。vLens 使用电量测试仪器，同时对结合 eLens 方法中细粒度的静态程序分析，实现了细粒度的动态能耗测试方法。Wonwoo Jung[8]等人设计实现了 DevScope 工具，它通过在安卓操作系统中添加探针，利用动态测电的方式获取每个系统调用

的耗电。Chanmin Yoon 等人[9]设计实现了 appScope 工具，它通过监控安卓应用在 Linux 内核的系统调用，利用[8]中 DevScope 给出的这些系统调用的耗电数据，提供了一种针对安卓应用的测电方法。Abhinav Pathak 等人[10,11]，Lide Zhang 等人[12]及 Jaewoo Chung 等人[13]监控了 CPU 使用率、网络使用率等系统资源使用情况，并通过动态测电建立起它们与应用耗能的关系。Seo Chiyoung 等人[14-16]通过修改 Java 虚拟机和对 Java 字节码文件进行修改，研究了 Java 字节码的耗电情况。这些研究与本文的区别在于，为了提供较细粒度的能耗测试方法，需要对在安卓操作系统中加入探针或对安卓应用进行插桩，而本研究工作无需对安卓系统或安卓应用进行修改即可完成 activity 的电量测量和预测，这方便了开发者的使用。

3 研究方法

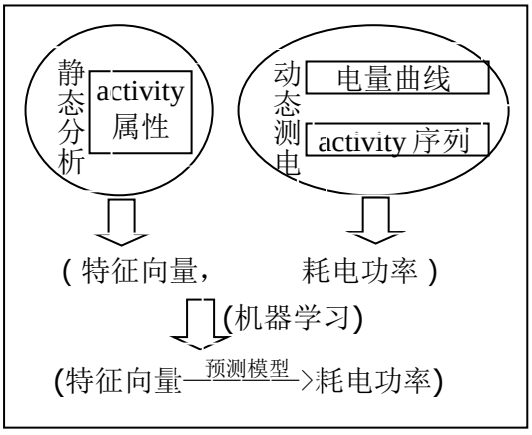

图 1　基本研究方法

本文的基本研究方法如图 1 所示。具体阐述如下：

3.1 动态测电

目前安卓操作系统更加流行，因此本研究基于一些流行的安卓应用进行的，利用 Monkey 自动化测试技术对其进行了测试。通过编写 windows 脚本，获取被测应用的 activity 运行序列，同时使用 Power Monitor 仪器进行动态电量测量，以此获得特定 activity 的电量消耗曲线，并通过计算得到其耗电功率。

若要准确测得 activity 的耗电功率，需要将 Power monitor 仪器测得的电量消耗曲线和 Monkey 测得的 activity 运行序列之间的时间对齐。进而通过积分的方式获得此段时间 activity 的电量消耗，除以时间长度，即得到其耗电功率。

3.2 静态分析

使用 Soot 等静态分析工具对被测 app 进行静态分析，获取 activity 的相关属性，并将这些属性的集合提取为此 activity 的特征向量。

本研究中，选取了认为的与耗电相关的 activity 属性总计 11 个，对每个 activity 构建了形如 <代码行数，方法个数，调用个数，系统调用个数，方法平均代码行数，循环/递归个数，传感器调用个数，唤醒锁个数，蓝牙调用个数，音视频播放调用个数，网路传输使用次数> 这样包含 11 个属性的特征向量。选择每个属性的理由依次说明如下：

1. 代码行数

Activity 的代码行数越长，运行时资源消耗可能越大，因此其耗电功率可能越大。

2. 方法个数、调用个数、系统调用个数

Activity 的方法个数、调用个数、尤其是系统调用个数越多，此 activity 运行时资源消耗可能更多，因此耗电功率可能更大。

3. 方法平均代码行数

这个属性与代码行数和方法个数相关，单个方法的代码行数越长，其执行时消耗电量可能越大，即耗电功率越高。

4. 循环/递归个数

循环/递归的存在涉及代码重复运行，而代码一旦重复运行，其运行时资源消耗可能会增加，因此循环/递归的个数很可能与耗电功率有关。

5. 传感器调用个数、唤醒锁个数、蓝牙调用个数、音视频播放调用个数、网络传输使用次数

这五种属性对应的操作均为较为耗能的操作，因此这五种操作数量越多，耗电功率可能越大。

3.3 机器学习

获取 activity 的特征向量和耗电功率后，为

了得到预测模型，需要获取两个数据集间的相关关系。在机器学习方法中，一些经典的分析和回归算法能够较为精准地预测数据集间的相关关系，因此本研究工作中使用了机器学习方法，通过使用 Weka 这一机器学习工具，将所测得的所有(特征向量，耗电功率)数据作为输入，使用 Weka 的分析和回归功能提供的模拟函数，获取(特征向量—>耗电功率)的预测模型。对最终得到的预测模型还需要进行误差分析。

4 实验步骤

实验分准备工作、动态测电、静态分析、机器学习四个步骤，最终获取 activity(特征向量—>耗电功率)的预测模型。

4.1 准备工作

对于市场上的大多数被测应用，其代码经常都是经过混淆的。如果直接进行动态测电，使用本实验的方法，动态测电过程中获取的 activity 名称将会是其原名称，而静态分析中获取的却是经过混淆后的，形如 a,b,c 这样的名称。这会造成两个步骤中 activity 名称的不一致，因此需要进行一些预处理。

具体做法是：对于被测应用，下载其 apk 后，使用相关工具进行反编译，重编译和重新签名，可运行的待测 apk，并且解决了上文所述不同步骤中 activity 名称不一致的问题。

4.2 动态测电

图 2 Power monitor 设备及测试手机

如图 2 所示，实验使用的测试手机是搭载了安卓 4.4 系统的三星 GALAXY Note4 手机，使用的 power monitor 仪器是 Monsoon 的功率测试仪(型号 FTA22D)。使用此款测试手机的理由如下：Power monitor 仪器通过输出电压，代替电池对手机进行供电，以此测量手机的运行时功率。本实验中，为了方便操作，就使用这款电池可拆的手机作为测试手机。这款手机存在的一个缺点是其搭载的安卓系统版本较低。市场上搭载较高安卓系统版本的手机多为电池一体机，对其进行动态测电较为困难。这也是本工作需要改进的一点。

4.2.1 动态测电步骤

动态测电具体步骤中，首先运行如下 windows 脚本，获取被测应用包名：

```
adb shell "dumpsys activity | grep mFocusedActivity"
```

Power monitor 仪器即开始测电。之后运行如下 Monkey 脚本：

```
adb shell monkey --throttle 1000 -p 应用包名 -v 1000
```

指定 Monkey 对被测应用运行 1000 个随机事件进行测试，每个随机事件的时间间隔为 1000 毫秒。运行 Monkey 脚本后，立即运行如下脚本：

```
:loop
    echo %time% | adb shell "dumpsys activity | grep mFocusedActivity" >> recordActivity.txt
goto loop
```

此脚本会动态监控被测应用中 activity 的运行情况，并将形如(时间，activity 名称)的记录输出到 recordActivity.txt 文件中。通过进一步对此文件进行分析，则可以得到 activity 运行序列。

Power monitor 仪器测得的电量曲线，如图 3 所示，横坐标是时间，纵坐标是功率,时间轴从 0 开始，因为利用工具控制从时间点 t 开始测量，在 4.2.2 节中验证延迟干扰很小后，可以认为时间轴上的时间加 t 后即为实际时间，这与 activity 运行序列的时间是对齐的。Power monitor 测得的电量曲线是由被捕获的时间间隔为 0.2ms 的测量点组成的。将被测数据存储为 csv 文件，则实际存储在 csv 文件中的即为这些测量点。由 activity 运行序列可以得到某个特定 activty 的运

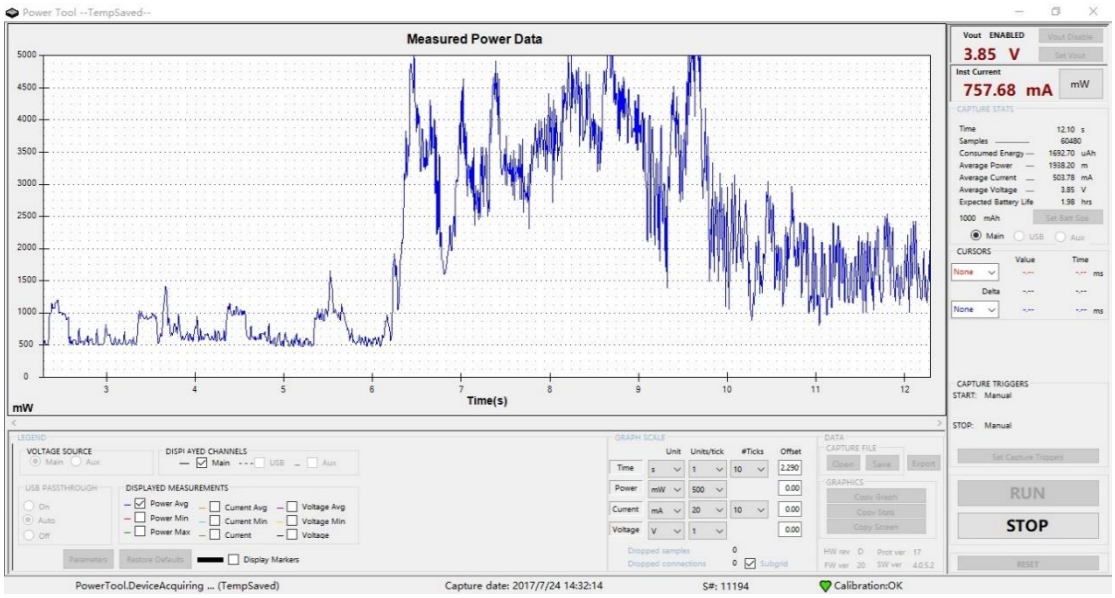

图 3　Power monitor 用户界面

行时间 t0 ~ t1，t2 ~ t3，…在 csv 文件中寻找对应时间段，通过计算积分的方式可以得到测试手机此时间段消耗的电量，除以时间即为此 activity 运行时平均功率 P1。通过测出手机闲置时的功率 P0，则可以认为 P = P1 － P0 即为纯粹的、由 activity 运行引起的耗电功率。

4. 2. 2　　时间对齐验证实验

4.2.1 节中，考虑电脑和 Power monitor 仪器之间的传输延迟，可能导致鼠标点击图 3 中的“RUN”按钮的时间点 t 到电量曲线的时间点 0 之间存在时间延迟。本小节通过实验验证这一时间延迟极小，因此可以认为对 4.2.1 中的电量曲线和运行序列时间对齐无影响。

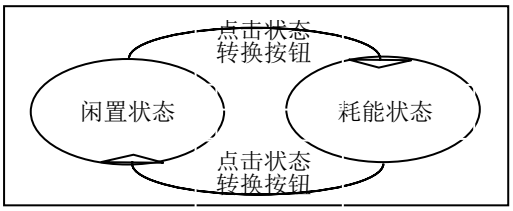

图 4　验证程序状态机图

验证实验通过设计并实现一个验证程序，此程序的状态机图如图 4 所示。它有两个 activity，代表了两个不同的状态：第一个状态为闲置状态，运行时功率很低；第二个状态为耗能状态，运行时功率较高。利用 4.2.1 节中的方法测得电量曲线和 activity 运行序列，若电量曲线中某一点由低功率状态进入高功率状态(或相反)，即可认为在这一点发生了状态转换。实验发现，电量曲线中的状态转换点时间与 activity 运行序列中 activity 变化时间点的时间误差在 0.001ms 以内，而 4.2.1 节中两个相邻测量点的时间间隔为 0.2ms，因此可以认为电脑和 Power monitor 仪器之间的传输延迟对实验无影响。

4. 3　　静态分析

对 3.2 节中每个属性的获取方法依次介绍如下：

1.　代码行数

对反编译后的 apk，使用 SourceCounter 工具可以获得其中某个特定 activity 的代码行数，此代码行数是去除空行和注释后的代码行数。

2.　方法个数、调用个数、系统调用个数

这三个属性是利用 Soot 对 apk 进行静态分析获得的。Soot 中的 SootClass, SootMethod, 和 Body.Unit 对象即分别对应 activity，activity 中的方法，和方法中的调用。通过分析这三个对象，即可获得某个特定 activity 中的方法个数和调用个数。通过判断是否有关键字“android”或“java”即可判断某个调用是否为系统调用，也即可获得某个 activity 中的系统调用的个数。

3. 方法平均代码行数

首先通过文本分析获取全局变量等所占代码行数，并由 1 和 2 中得到的数据，根据以下公式计算得到结果：

方法平均代码行数 = (代码行数 – 全局变量等代码行数) /方法个数

4. 循环/递归个数

此属性主要是通过文本分析的方式获取的。对于循环个数，认为其等于 activity 代码中关键字 for 和 while 的个数和。对于递归个数。主要是通过判断某个函数是否调用其本身来得到的。两者之和即此属性的值。

5. 传感器调用个数

此属性同样是通过文本分析的方式获取的。通过分析 activity 中是否存在安卓常用的 9 种传感器调用关键字可以判断是否使用了对应的传感器，进而得到传感器调用个数。

6. 唤醒锁个数、蓝牙调用个数、音视频播放调用个数、网络传输使用次数

这四个属性仍然是通过文本分析的方式获取的。通过检索 WekaLock 关键字获取唤醒锁个数；通过检索 BlueToothAdapter 关键字获取蓝牙调用个数；通过检索 MediaRecorder 类的使用获取音视频播放调用个数；通过检索 HttpURL-Connection 和 Socket 关键字获取网络传输使用次数。

4.4 机器学习

使用 Weka 这一机器学习工具。将测得 activity 的(特征向量，耗电功率)作为输入，在 Weka 中选取合适的模拟函数，进而生成预测模型。通过选取特征向量的不同子集作为新的特征向量，生成新的预测模型，并比较不同预测模型的精准度。若存在子特征向量的预测模型更精准，则可以去除其不包含的属性，即认为此属性与 activity 耗电弱相关。

5 实验结果

实验对当今较流行应用进行了分类，总计分为 8 类，每类软件选择了 1~2 个代表应用进行测试。分类及选择的软件分别是：通讯软件（微信），社交软件（百度贴吧、知乎），购物软件（手机淘宝、京东），音视频播放软件（网易云音乐、优酷客户端），阅读软件（动漫之家），地图软件（百度地图），浏览器（UC 浏览器），游戏软件（王者荣耀、愤怒的小鸟）。此 8 类 12 个应用基本能够代表用户生活中使用的大多数应用。

实验总计测得了 12 个应用中的 81 个不同 activity 的(特征向量，耗电功率)数据，表 1 展示了每个应用在实验中被测到的 activity 个数。

表 1 应用测得的 activity 个数

应用名称	activity 个数
微信	4
百度贴吧	7
知乎	1
手机淘宝	11
京东	9
网易云音乐	10
优酷客户端	9
动漫之家	3
百度地图	9
UC 浏览器	6
王者荣耀	5
愤怒的小鸟	7

5.1 模拟函数和特征向量的选取

如 4.4 节所述，选取特征向量的子集构造新的特征向量。称初始特征向量为元特征向量，通过分别去除元特征向量的一个属性，构造出 11 个各含有 10 个属性的子特征向量作为新特征向量。

Weka 的用户界面如图 5 所示。在图 5 中的 Preprocess 模块读取输入文件后，转到 Classify 模块，在 Choose 按钮下选择不同的模拟函数对输入数据进行处理，实验中共选取了 11 个可用的模拟函数。对于每次数据处理，Weka 都会在图 5 的 Classifier output 模块输出此次模拟的相关系数、相对误差和绝对误差。相关系数越接近 1，

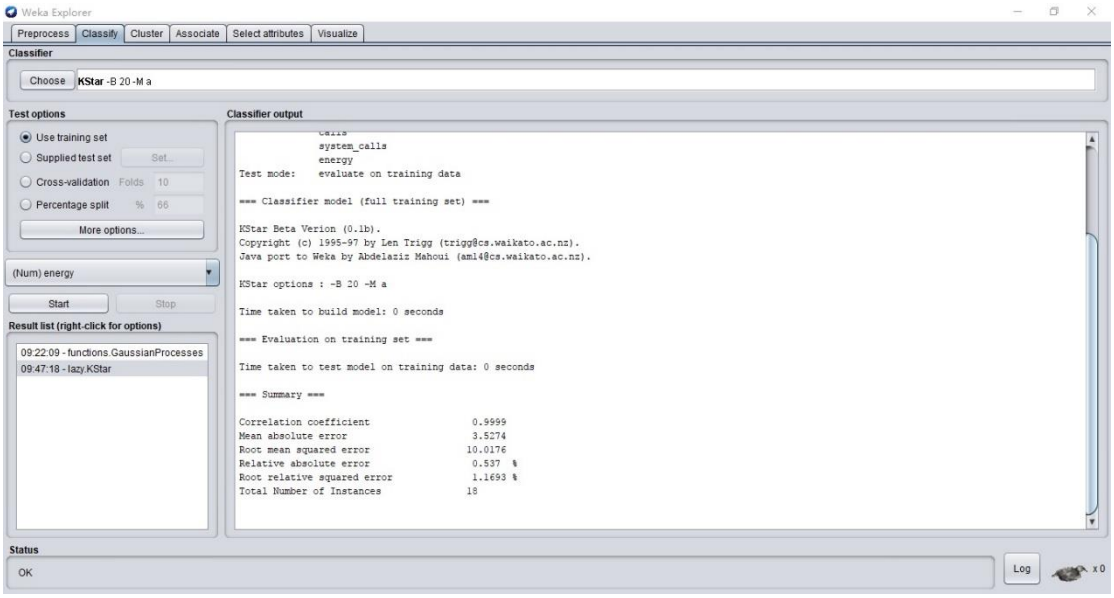

图 5　Weka 用户界面

则相对误差越小。不同的特征向量和模拟函数对应的相关系数如表 2 所示。

表 2 的第一行表示不同的特征向量，“-”表示从元特征向量中取出某属性后剩下的子特征向量；第一列表示 Weka 提供的不同模拟函数名；表 2 中的数据表示不同的特征向量和模拟函数对应的相关系数，其中每行最接近 1 的相关系数用粗体字标出。从表 2 中的数据可以看出，在 11 个模拟函数中，大多数(13 个)模拟函数在元特征向量下的预测模型是最精确的。这表明大多数情况下，去除任意一个属性后预测精度均会下降。说明 4.2 节中选择的属性是较为合理的。

在其中，KStar 这一模拟函数得到的预测模型的相关系数为 0.9999，最接近于 1，其相对误差为 0.01%。因此得出结论：KStar 模拟函数对元特征向量的预测模型是最精确的。

这一结论面向开发者，开发者通过构建自己应用的 activity 的元特征向量，使用 Weka 的 KStar 模拟函数，即可对自己的 activity 进行耗电功率预测。

5.2　实验结果误差分析

为了进行误差分析，实验继续对两款开源软件——2buntu 和 Aard 进行了测试。2buntu 是一款社交软件，测得其 4 个不同 activity（表 3 中前 4 个）；Aard 是一款阅读软件，测得其 5 个不同 activity（表 3 中后 5 个）。总计测得了 9 个不同 activity 的(特征向量，耗电功率)数据。这里耗电功率指的是实际功率，即按 4.2.1 节中提到的动态测电方法先积分求得电量消耗后除以时间得到的平均耗电功率。同时使用 Weka 的预测功能，基于 5.1 小节结论中的预测模型(KStar 函数对元特征向量的预测模型)，在图 5 的 Classify 模块中，以这 4 个 activity 的特征向量作为输入，获得其预测耗电功率，即预测功率。进而可以按照以下公式计算相对误差：

相对误差 = | 实际功率 – 预测功率 | / 实际功率

表 3　对 activity 的预测和误差分析

	实际功率 /mw	预测功率 /mw	相对误差 /%
articleListActivity	2130.21	2187.422	2.686
articleDetailActivity	1796.04	1534.582	14.557
CommentActivity	1517.75	1390.819	8.363
SettingsActivity	1213.18	1317.222	8.576
ArticleViewActivity	1023.33	1047.409	2.353
BaseDictionaryActivity	1892.35	1806.835	4.519
DictionariesActivity	2343.42	2349.161	0.245
DictionaryInfoActivity	1232.72	1215.006	1.437
LookupActivity	2031.94	1909.434	6.029

表 2　不同模拟函数和特征向量的相关系数

相关系数	元特征向量	-代码行数	-方法个数	-调用个数	-系统调用个数	-方法平均代码行数
GaussianProcesses	**0.8121**	0.8118	0.7644	0.7323	0.7335	0.8027
LinearRegression	**0.8318**	0.8113	0.7478	0.7685	0.7643	0.8007
MultilayerPerceptron	**0.9327**	0.8723	0.8494	0.8576	0.8434	0.8632
KStar	**0.9999**	0.9978	0.9954	0.9889	0.9943	0.9964
LWL	0.8631	0.8748	0.8643	0.8237	0.8432	0.8775
AdditiveRegression	**0.9808**	0.9806	0.9754	0.9461	0.9541	0.9803
RandomSubSpace	0.8183	**0.937**	0.8123	0.8584	0.8326	0.9239
DecisonTable	**0.9434**	0.8906	0.8841	0.8516	0.8345	0.8913
M5Rules	0.9374	**0.939**	0.9231	0.9198	0.9243	0.9365
M5P	**0.9122**	0.9031	0.8838	0.8756	0.889	0.9019
RandomForest	**0.9829**	0.9807	0.9321	0.9523	0.9723	0.9713

续表 2：

相关系数	-循环/递归个数	-传感器调用个数	-唤醒锁个数	-蓝牙调用个数	音视频播放调用个数	网络传输使用次数
GaussianProcesses	0.7953	0.7743	0.7965	0.8013	0.7831	0.7965
LinearRegression	0.784	0.7817	0.789	0.7892	0.7951	0.8234
MultilayerPerceptron	0.8564	0.8675	0.8695	0.9024	0.8943	0.9124
KStar	0.9875	0.9912	0.9923	0.9931	0.9981	0.9976
LWL	0.8423	0.8469	0.8576	0.8412	0.8214	**0.8942**
AdditiveRegression	0.9345	0.938	0.9332	0.9760	0.9565	0.9752
RandomSubSpace	0.8792	0.8474	0.8367	0.9212	0.9012	0.9124
DecisonTable	0.8721	0.8456	0.8546	0.9313	0.9021	0.9132
M5Rules	0.9134	0.9276	0.9134	0.9132	0.8932	0.9075
M5P	0.8545	0.8721	0.8643	0.9043	0.8923	0.8824
RandomForest	0.9634	0.9459	0.9443	0.9763	0.9584	0.9789

由表 3 中的数据可见，除个别数值外，相对误差均在 10%以内，70%数据误差在 7%以内，平均为 5%。

以下相关工作给出了他们的预测方法的精准度：Tao Li 等人[3]监控系统层次 API 调用个数，其预测方法的精准度为 5%以下。T.K. Tan[4]等人同样的预测方法，预测精准度波动较大，平均约 10%，Ding Li 等人[7]的 vLens 工具，平均预测精准度约为 6%，Chanmin Yoon 等人[9]的 appScope 工具，平均预测精准度约为 7%。Lide Zhang 等人[12]监控 CPU 使用率、网络使用率等系统资源使用情况，其预测方法的平均精准度约为 9%。本研究工作与这些工作比较，预测模型误差相当或小于其预测误差，因此可以认为本实验得到的 activity 耗电功率预测模型是较为准确的。

6　结束语

本文提出了一种动态测试和静态分析相结合的预测安卓应用 activity 的耗电方法。在动态测试方面，该方法使用 Monkey 自动化测试技术和

Power Monitor 测电仪器进行动态电量测量，以此得到被测 activity 的电量消耗功率。在静态分析方面，利用 Soot 等静态分析工具获取被测 activity 的 11 个属性，根据这 11 个属性构建其特征向量，由此得到每个 activity 的(特征向量，耗电功率)数据。最后使用机器学习工具 Weka，得到 (特征向量—>耗电功率)的预测模型。此模型面向开发者，使其可以预测安卓应用 activity 的耗电功率。对实验结果的误差分析表明，最终得到的预测模型的准确度是较高的。

本文所做的研究工作还存在一些不足之处。第一点是对安卓应用能耗预测的粒度不够细，仅仅在安卓的 activity 层次进行电量测量和预测。后续工作需要进一步在方法层次或代码行层次进行电量测量和预测，这需要对被测应用进行插桩，以进行更加细粒度的动态测试，获取安卓 activity 的方法执行序列或代码行执行序列。第二点是针对 activity 构建的特征向量的属性值偏少，只有 11 个，如果利用更多合适的 activity 属性构建其特征向量，其应该可以更好地描述 activity，进而通过机器学习得出更精确的预测模型。最后一点即 4.2 小节中提到的测试手机的安卓系统版本较低，后续需要对高版本的安卓系统进行测试，这需要搭载高版本安卓系统的电池一体机进行拆机，以使 Power monitor 仪器能够对其进行测电。以上几点不足之处也是今后工作改进的方向。

参考文献

[1] http://www.199it.com/archives/615604.html

[2] HAO S, LI D, HALFOND W G J, et al. Estimating Mobile Application Energy Consumption Using Program Analysis[C]. 35th International Conference on Software Engineering(ICSE), 2013(13): 92-101.

[3] LI T, JOHN L K. Run-time Modeling and Estimation of Operating System Power Consumption[C]. Measurement and Modeling of Computer Systems (SIGMETRICS), 2003, 31(1): 160-171.

[4] TAN T K, RAGHUNATHAN A, JHA N K. Energy Macromodeling of Embedded Operating Systems[J]. ACM Transactions on Embedded Computing Systems (TECS), 2005, 4(1): 231-254.

[5] AGGARWAL K, ZHANG C, CAMPBELL J C, et al. The Power of System Call Traces: Predicting the Software Energy Consumption Impact of Changes[C]. Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, 2014(14): 219-233.

[6] MA X, HUANG P, JIN X, et al. eDoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphones[C]. Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, Berkeley, 2013, (13): 57-70.

[7] LI D, HAO S, HALFOND W G J, et al. Calculating Source Line Level Energy Information for Android Applications[C]. Proceedings of the 2013 International Symposium on Software Testing and Analysis, 2013: 78-89.

[8] JUNG W, KANG C, YOON C, et al. Devscope: a Nonintrusive and Online Power Analysis Tool for Smartphone Hardware Components [C]. Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2012(12): 353-362.

[9] YOON C, KIM D, JUNG W, et al. AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring[C]. Proceedings of the 2012 USENIX Conference on Annual Technical Conference USENIX Annual Technical Conference, 2012(12): 36-36.

[10] PATHAK A, HU Y C, ZHANG M, et al. Fine-grained Power Modeling for Smartphones Using System Call Tracing[C]. Proceedings of the 6th Conference on Computer Systems, 2011(11): 153-168.

[11] PATHAK A, HU Y C, ZHANG M. Where is the Energy Spent Inside My App?: Fine Grained Energy Accounting on Smartphones with Eprof[C]. Proceedings of the 7th ACM European Conference on Computer Systems, 2012(12): 29-42.

[12] ZHANG L, TIWANA B, DICK R P, et al. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation For Smartphones[C]. 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2010: 105-114.

[13] CHUNG J, JUNG H R, KOO J, et al. A Development of Power Consumption Measurement System for Android Smartphones[C]. Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, 2017(17): 95.

[14] SEO C, MALEK S, MEDVIDOVIC N. An Energy Consumption Framework for Distributed Java-Based Systems[C]. Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, 2007(07): 421-424.

[15] SEO C, MALEK S, MEDVIDOVIC N. Component-Level Energy Consumption Estimation for Distributed Java-Based Software Systems[J]. Proceedings of the 11th International Symposium on Component-Based Software Engineering, 2008(08): 97-113.

[16] SEO C, MALEK S, MEDVIDOVIC N. Estimating the Energy Consumption in Pervasive Java-Based Systems[C]. 6th Annual IEEE International Conference on Pervasive Computing and Communications, 2008: 243-247.