Empirical Software Engineering (2021) 26: 38
https://doi.org/10.1007/s10664-020-09925-5

GreenHub: a large-scale collaborative dataset to battery consumption analysis of android devices

Rui Pereira[1,2] · Hugo Matalonga[3] · Marco Couto[2,3] · Fernando Castor[4] · Bruno Cabral[5] · Pedro Carvalho[5] · Simão Melo de Sousa[6] · João Paulo Fernandes[5]

Accepted: 11 December 2020 / Published online: 20 March 2021

Abstract

Context The development of solutions to improve battery life in Android smartphones and the energy efficiency of apps running on them is hindered by diversity. There are more than 24k Android smartphone models in the world. Moreover, there are multiple active operating system versions, and a myriad application usage profiles.

Objective In such a high-diversity scenario, profiling for energy has only limited applicability. One would need to obtain information about energy use in real usage scenarios to make informed, effective decisions about energy optimization. The goal of our work is to understand how Android usage, apps, operating systems, hardware, and user habits influence battery lifespan.

Method We leverage crowdsourcing to collect information about energy in real-world usage scenarios. This data is collected by a mobile app, which we developed and made available to the public through Google Play store, and periodically uploaded to a centralized server and made publicly available to researchers, app developers, and smartphone manufacturers through multiple channels (SQL, REST API, zipped CSV/Parquet dump).

Results This paper presents the results of a wide analysis of the tendency several smartphone characteristics have on the battery charge/discharge rate, such as the different models, brands, networks, settings, applications, and even countries. Our analysis was performed over the crowdsourced data, and we have presented findings such as which applications tend to be around when battery consumption is the highest, do users from different countries have the same battery usage, and even showcase methods to help developers find and improve energy inefficient processes. The dataset we considered is *sizable*; it comprises 23+ million (anonymous) data samples stemming from a large number of installations of the mobile app. Moreover, it includes 700+ million data points pertaining to processes running on these devices. In addition, the dataset is *diverse*. It covers 1.6k+ device brands, 11.8k+

Communicated by: Yasutaka Kamei, Andy Zaidman

This article belongs to the Topical Collection: *Mining Software Repositories (MSR)*

✉ Rui Pereira
rui.alexandre.pereira@ubi.pt

Extended author information available on the last page of the article.

smartphone models, and more than 50 Android versions. We have been using this dataset to perform multiple analyses. For example, we studied what are the most common apps running on these smartphones and related the presence of those apps in memory with the battery discharge rate of these devices. We have also used this dataset in teaching, having had students practicing data analysis and machine learning techniques for relating energy consumption/charging rates with many other hardware and software qualities, attributes and user behaviors.

Conclusions The dataset we considered can support studies with a wide range of research goals, be those energy efficiency or not. It opens the opportunity to inform and reshape user habits, and even influence the development of both hardware (manufacturers) and software (developers) for mobile devices. Our analysis also shows results which go outside of the common perception of what impacts battery consumption in real-world usage, while exposing new varied, complex, and promising research avenues.

Keywords Green software · Green mining · Android · Battery consumption analysis

1 Introduction

Battery life is known to be one of the major factors influencing the satisfaction of mobile device users (Thorwart and O'Neill 2017). A recent survey with 1,894 smartphone users in the US placed battery life as the most important factor impacting smartphone purchasing decisions (Richter 2018). Battery life is such a growing concern that it has been hypothesized that 9 out of 10 users suffer from low battery anxiety (Mickle 2018).

Developers are also very concerned with the impact their applications have on battery life. Excessive battery consumption is one of the most common causes for bad app reviews in app stores (Fu et al. 2013; Khalid et al. 2015). In fact, developers are aware of the battery consumption problem, and many times seek help in solving this, even if they rarely receive adequate advice (Pinto et al. 2014; Manotas et al. 2016; Pang et al. 2016). Mobile device manufacturers recognize this issue and have tried to offer help by publishing developer guides aimed at extending battery life.[1,2,3]

Reducing the energy that is consumed by mobile devices is also an important problem from a sustainability point of view. Indeed, the billions of phones that are in use these days have a considerable environmental footprint. Our digital consumption (that includes but is not limited to mobile device usage) is bound to have a greater impact on global warming than the aviation industry (Harris 2018).

Despite its importance, optimizing, or even analyzing energy consumption for mobile devices is a difficult and labor-intensive task for both users and/or developers. Developers are using different monitoring tools (Nucci et al. 2017; Hu et al. 2017; Cruz and Abreu 2017) according to specific needs, which often results in a non-systematized procedure and context-specific findings (Li and Halfond 2014; Li et al. 2016; Cruz and Abreu 2017; Oliveira et al. 2019). Monitoring the energy consumed by an application often results in extensive tests under several different scenarios and devices (Li et al. 2013; Linares-Vásquez et al. 2014; Jabbarvand et al. 2015), both very time consuming and potentially requiring

[1] https://developer.android.com/topic/performance/power/
[2] https://developer.android.com/guide/topics/location/battery
[3] https://developer.android.com/docs/quality-guidelines/building-for-billions-battery-consumption

large initial investments. In addition, Android is an heterogeneous platform. In 2015, there were already more than 24,000 Android device models available in the world (Tung 2015). A recent study found out that there are more than 2.7 million apps in the Google Play Store. The Android operating system is currently in its 10th major release, with multiple minor releases throughout the years. These numbers compound with the different ways in which apps and devices are used to produce a virtually infinite number of potential usage scenarios.

For users, understanding the energy consumption of their devices is an even harder exercise. Their knowledge regarding the behavior of the hardware is limited to their own devices. Furthermore, without the proper tools and skills, they cannot compare the energy behavior of the apps they use, nor observe how such apps perform on other devices or under specific settings and conditions. Moreover, different usage contexts of the same app, e.g., within different OS versions and with different hardware components switched on, result in different energy behaviors. This has to be taken into account when performing any comparison.

In this paper, we present a descriptive and qualitative empirical study through indirect observation of a large real-world representative dataset, of day-to-day usage of Android devices. The aim of our qualitative study is to highlight associations between various characteristics and tendencies within a smartphone's ecosystem (considering real-world interaction and varying usage profiles) and battery usage. This means that our study is not intended to, nor capable of, directly *blaming* a particular characteristic for an observed high battery consumption. Instead, we reveal tendencies which occur with concrete characteristics, that need to be isolated in further dedicated studies in order to confirm/contradict the responsibility of a given characteristic in such consumption scenarios. Our qualitative empirical study provides rich contextual data to better help understand the current status and phenomenon between smartphone usage and battery consumption.

The dataset under descriptive and qualitative analysis was gathered as part of the GreenHub initiative,[4] a collaborative approach to Android energy consumption analysis. While in this paper we propose to contribute with an empirical study over this gathered data, the dataset itself, as a scientific contribution, originally appeared in Matalonga et al. (2019), from the same authors of this paper.

The entries in the GreenHub dataset include multiple pieces of information, e.g., active sensors, memory usage, battery voltage and temperature, running applications, model and manufacturer, and network details. This raw data was obtained by continuous crowdsourcing through a mobile application called BatteryHub. It is worth noting that all such data is publicly available, while maintaining the anonymity and privacy of all its users. Indeed, it is impossible to associate any data with the user who originated it. The dataset is *sizable* and thus far it comprises of 23+ million unique samples, including more than 700+ million data points pertaining to processes running on these devices. The dataset is also *diverse*. It includes data stemming from 1.6k+ different brands, 11.8k+ smartphone models, from over 50 Android versions, across 160 countries.

GreenHub's vision is to provide developers, researchers, device manufacturers, users, and any interested party with a rich set of data on how mobile devices and the apps running on them behave and use resources. It is expected that this data can promote the analysis and identification of opportunities to optimize energy consumption in Android devices, for both developers and users, and our work in this paper attempts to provide a concrete step in that direction.

[4] https://greenhubproject.org/

For our qualitative analysis, we present a detailed characterization of the data and information within GreenHub's dataset. This characterization allows not only a better understanding of the collected data for people trying to understand energy patterns in mobile devices, but also for those who might see the GreenHub dataset as a potential basis for studies with different goals (not necessarily associated with energy and battery consumption) across a wide range of directions. Indeed, we argue that this data is both rich and diverse enough to support other analyses in the context of Android.

In the case of developers, we believe our analyses will trigger further analyses that are beyond GreenHub's dataset itself. These may explore the potential energy gains that have, e.g., been proposed in the context of location services (Lin et al. 2010), contrast (Linares-Vásquez et al. 2015), color scheme (Linares-Vásquez et al. 2015; Wan et al. 2017), data structure (Hasan et al. 2016; Pereira et al. 2018; Pinto et al. 2016; Pereira et al. 2016), programming language (Oliveira et al. 2017; Pereira et al. 2017; Couto et al. 2017; Lima et al. 2016), network usage (Li et al. 2016), and API (Linares-Vásquez et al. 2014) usage.

After thoroughly detailing GreenHub's dataset, we present a set of 15 different research questions (RQs), and analyze the data in order to answer such questions (ARQs), regarding the battery consumption of Android smart-phones according to different scenarios and characteristics, such as: what are the battery/discharge tendencies across different countries, devices, network operators or brands. This lead us to several interesting findings. For instance, we noticed that with each new Android major version battery consumption actually tended to improve, but only after a particular version. Also, we were able to identify potential anomalies within a popular Android application, and even gather results that defied common intuition.

In order to answer such questions, we defined a new battery charging/discharging metric to be used within our dataset, which maintains a notion of data periods originating from the same device. This metric, which we named PPM, or *percentage-per-minute*, was used for understanding how a wide range of factors impact the battery consumption of Android devices and for identifying common tendencies.

PPM is a numerical value, either positive PPM^{+} or negative PPM^{-} reflecting battery charge or discharge respectively, calculated based on battery percentage change within one minute. PPM does not represent an absolute battery, energy, or efficiency metric, but the **tendency** a given scenario has on the charging/discharging of the battery. This metric factors in real-world human interaction, and aspects such as different user profiles across vastly different devices, brands, countries, battery settings, and application usage.

In order to illustrate the potential of our dataset, we will also be presenting new and diverging potential research paths, which spontaneously sprouted from our work and analysis. These RPs further complement the RQs presented in this paper, by focusing on more specific questions and findings, while also opening research paths to establish the causality of our highlighted associations and suggesting hypotheses which can be tested in analytical studies. We believe they are as equally promising to be addressed and should so in dedicated studies, either by fully, or partially (through means of combining other datasets), using our presented dataset or data present within the dataset. Note that some of these prospective research avenues are not specifically aiming for energy/battery consumption research, but also other areas of research on Android devices. For example, our data can help with research paths such as: how do new applications spread worldwide?; How frequent do users update their OS versions?; Is there a relationship between a country's average temperature and battery drain; etc.

Not only is our dataset completely public, open-source, and permanently available, but also, for reproducibility, all the tools and processes used to generate the results and information in this paper are also publicly available. This includes the GreenHub Pipeline based on Jupyter Notebooks and the Python Pandas library, which allows for human-readable documents to perform data analysis in a modular and easy to visualize manner. This Pipeline performed all the data cleaning, processing, and analysis which produced the results and answers to our RQs within this paper.

As we have previously mentioned, the work presented in this paper extends previous work (Matalonga et al. 2019) in which we presented the GreenHub infrastructure, the dataset model, and quantified the collected samples. This prior work is presented in Section 2, which we have added a more detailed explanation of how to access the data through our REST API, and examples of how to query the Farmer database. Additionally, this public dataset itself has been updated to contain half a year's worth of new data samples. The novel contributions include the detailed exploration of the dataset, study on the battery tendencies based on human-smartphone interactions, proposal of other research avenues based on this fruitful dataset, and an integrated data cleaning and analysis pipeline.

To summarize, the main contributions of our research are:

1. A detailed characterization of the GreenHub dataset, thus providing knowledge and understanding of the data in the dataset;
2. A qualitative empirical analysis of the collected human-smartphone interaction data to help identify the tendencies between battery charging/discharging of Android smartphones and various smartphone characteristics;
3. The proposal of several generic and energy/battery related research paths enabled by the data in the GreenHub dataset;
4. A replication package consisting of the GreenHub Pipeline, for cleaning data, characterizing and querying the dataset, and analyzing battery charging/consumption tendencies.

The remainder of this paper will describe:

- The setup and infrastructure for data collection, analysis, and storage **(Section 2)**;
- A detailed characterization of the GreenHub dataset **(Sections 3)**;
- An empirical study and results on the battery consumption tendencies of Android devices **(Section 4)**;
- Proposal of several research opportunities based on the data collected in GreenHub and in Section 4 **(Section 5)**;
- A discussion and analysis on a peculiar aspect of the GreenHub dataset (**Section 6**);
- The GreenHub Pipeline notebook replication package **(Section 7)**;
- A look at the threats to the validity of our study and results **(Section 8)**;
- Description of related work **(Section 9)**;
- Our conclusions and final comments of this paper **(Section 10)**.

2 Collaboratively Collecting and Sharing Data

In this section, we describe GreenHub, the initiative from which the dataset studied in this paper stemmed. While the dataset itself has already been introduced in Matalonga et al. (2019), we go back to describing the data collection procedure in order to provide readers a

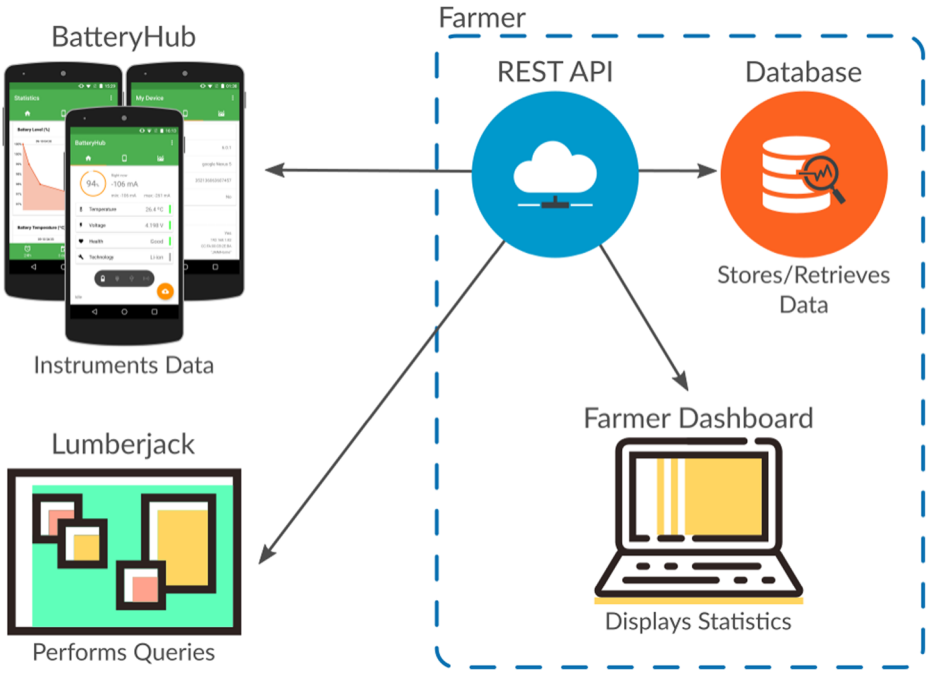


Fig. 1 GreenHub platform architecture

comprehensive description of our context. In this section, we also extend the description of GreenHub's dataset with concrete examples of how it can be accessed and utilized.

GreenHub is committed to provide the means to support a symbiotic relationship with the mobile community. The success of the initiative is dependent on its data, and to keep such data coming in, we plan to give back to the community in concrete and valuable ways.

The initiative relies on a fully open-source multi-component technological platform,[5] whose architecture overview is shown in Fig. 1. This platform includes our data collection Android app called BatteryHub, a command-line application interface called Lumberjack, and the Farmer REST API for prototyping queries, dashboard interface, and database for storing data. These components are further defined in the following sub-sections.

2.1 Data Collection

A key component of the platform is BatteryHub, an Android app whose development was inspired by Carat (Oliner et al. 2013). Carat collects data regarding apps running on a device, and uses it to provide battery-saving recommendations, such as when to close an app. Initially, we forked its open-source code to take advantage of the data collection and storage mechanisms. We also updated its data model to consider more details on modern devices, such as NFC and Flashlight usage, for example. In the same spirit of Carat, BatteryHub

[5] https://github.com/greenhub-project

is entirely open-source. In contrast with Carat, however, all our collected data is permanently and publicly available, so as to strongly encourage and help others in collaborating, inspecting and/or reusing any artifact that we have developed or collected.

BatteryHub is an Android app that is available at Google's Play Store.[6] It tracks the broadcast of system events, such as changes to the battery's state and, when such an event occurs, obtains a sample of the device's current state. BatteryHub either uses the official Android SDK or custom implementations for universal device compatibility support, and periodically communicates with the server application (over HTTP) to upload, and afterwards remove, the locally stored samples. Each sample characterizes a wide range of aspects that may affect battery usage, such as sensor usage, temperature, and the list of running applications. Section 2.3 presents the complete set of attributes that BatteryHub collects. These attributes are explicitly mentioned in the application's terms and conditions and privacy policy. In addition, it is important to mention that the data collected from each user is made anonymous by design. Each installation of BatteryHub is associated with a random unique identifier and no personal information, such as phone number, location, or IMEI, is collected. This means that it is (strictly) not possible to identify any BatteryHub user, nor is it possible to associate any data with the user from whom it originates.

In order to start giving back to users as early as possible, BatteryHub already provides detailed information about the status of their device. Currently, it indicates: (i) the electric current level, temperature, voltage levels in a given period, and (ii) model specifications, network information, memory usage, and storage details. Information in (i) is re-actively updated when the battery's state changes, and in (ii) when a system event occurs. Coincidentally, in regards to sample collection frequency, a new data measurement is collected (to be sent to the GreenHub server) when the battery's state changes. In most cases, this translates to a sample being sent at each 1% battery change (which accounts to 95% of the time according to our data).[7]

A fully featured task manager is included and the application also provides interactive charts throughout different time periods showing changes pertaining to different aspects of the battery.

The app allows for configurable alerts, e.g. when the battery reaches a certain temperature, and our plan is to use BatteryHub to give suggestions to users, based on their usage profiles, on how to reduce the energy consumption of their device.

Having deployed BatteryHub, our main challenge in constructing the dataset was the acquisition of a large user base. For this, we were helped by our institutions and their media outlets to bring attention to and attract the general audience. Our strategy achieved circa 50 dedicated publications from national and institutional venues, through news,[8] magazines,[9] newspapers,[10] and radio shows,[11,12] in Portugal and Brazil alone. This attracted an initial

[6] https://play.google.com/store/apps/details?id=com.hmatalonga.greenhub

[7] The remainder accounts for battery changing by 2% or more

[8] www2.cin.ufpe.br/site/lerNoticia.php?s=1&c=94&id=1697

[9] www.visao.sapo.pt/actualidade/sociedade/2017-10-11-Bateria-do-telemovel-invista-agora-para-poupar-depois

[10] https://www.publico.pt/2017/10/09/tecnologia/noticia/desenvolvida-aplicacao-para-poupar-bateria-de-dispositivos-moveis-1788153

[11] www.90segundosdeciencia.pt/episodes/ep-443-joao-paulo-fernandes/

[12] www.rtp.pt/play/p2063/e342304/ponto-de-partida

large group of users to the application, naturally propagating outside of the host countries. Thus, all the samples present in our dataset are of real-world usage.[13]

Besides BatteryHub, our infrastructure includes four additional components, as depicted in Fig. 1. We envision they can be used in different stages of mining our dataset. The REST API within the Farmer component and Lumberjack, which are described in Section 2.2, are more appropriate for fast prototyping and to get acquainted with the data and the structure of the dataset. The use of the Database, which is described in Section 2.3 is probably mandatory for extensive and detailed mining of the dataset. Finally, our infrastructure also includes a web dashboard interface[14] that provides access to up-to-date statistics about the collected samples.

2.2 Prototyping Queries

The Farmer REST API was designed as a means to quickly interface with and explore the dataset. As every request made to the API must be authenticated, users must first obtain an API key in order to access the data in this fashion.[15] The API provides real-time, selective access to the dataset and one may query, e.g., all samples for a given brand or OS version. Since the API is designed according to the REST methodology, this allows us to incrementally add new data models to be reflected within the API itself as the data protocol evolves over time. After an API key has been successfully generated, one may request his/her own user profile from the API:

https://farmer.greenhubproject.org/api/v1/me?api_token=yourTokenHere

Every successful API response is a JSON formatted document, and in this case the server will reply with the user details, as shown next.

```
{   "data": {
        "id": XX,
        "name": "Your Name",
        "email": "your@email.com",
        "email_verified_at": "YYYY-MM-DD HH:MM:SS",
        "created_at": "Mon. DD, YYYY",
        "updated_at": "YYYY-MM-DD HH:MM:SS",
        "roles": [ ... ]   }   }
```

It is now possible to use the API, for example, to list devices:

https://farmer.greenhubproject.org/api/v1/devices?api_token=yourTokenHere

[13] The dataset does not include any data collected during development and testing of the BatteryHub application

[14] https://farmer.greenhubproject.org/

[15] https://docs.greenhubproject.org/api/getting-started.html

This request can take additional parameters for *page* and devices *per page*. A full description of all the available parameters for each request can be found in the API Reference.[16] Do note that the *App Processes* data is not present in our API as the size is extraordinary (over 749M data entries) and would overload the prototyping system. In this case, we would suggest that analysts opt to directly access the data. The expected response from the request above is as follows:

```
{   "data": [
        {   "brand": "asus",
            "created_at": "2017-10-28 02:51:09",
            "id": 2518,
            "is_root": false,
            "kernel_version": "3.1835+",
            "manufacturer": "asus",
            "model": "ASUS_X008D",
            "product": "WW_Phone",
            "os_version": "7.0",
            "updated_at": "2017-10-28 02:51:09"
        },  ... ],
    "links": { ... },
    "meta":  { ... }   }
```

To get more detailed information, e.g., about a particular device whose identifier is 123, it is possible to request its samples:

https://farmer.greenhubproject.org/api/v1/devices/123/samples?api_token=yourTokenHere

A complementary approach to interface with the API is to use its command-line application interface Lumberjack. Using this tool, users can perform flexible, on-demand queries to the data repository, to support quick prototyping of data queries applying different filters and parameters. Furthermore, users can quickly fetch subsets of the data without the need to download a snapshot of the entire dataset. The following is an example of a Lumberjack query to obtain the list of Google brand devices:

```
$ greenhub lumberjack devices \
  brand:google > -o googleDevices.json
```

The following example queries the dataset for samples whose model is nexus and that were uploaded before May 31st 2018:

```
$ greenhub lumberjack samples \
  model:nexus -R ..2018-05-31
```

[16] https://docs.greenhubproject.org/api-reference/

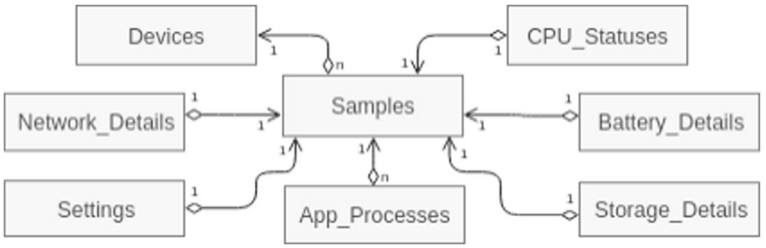


Fig. 2 Entity Relationship diagram of the dataset

2.3 Extensive Mining

The samples accessible through Farmer have been collected since November, 2017[17] and are available as a zip archive file, in CSV format.[18] The dataset is also available in Parquet[19] binary format,[20] which can be analyzed more efficiently than a plain text dump. The dataset is also available as a MariaDB relational database. The samples sent by BatteryHub are queued to be processed by a PHP server application built using the Laravel framework.[21] Each sample is received as a JSON formatted string that is deconstructed and correctly mapped within the database.

The (simplified) data model that we employ is shown in Fig. 2, where each box represents a table (or a CSV file) in the dataset. Samples is the most important of them, including multiple features of varied nature, e.g., the unique sample id, the timestamp for each sample, the state of the battery (charging or discharging), the level of charge of the battery, whether the screen was on or not, and the free memory on the device.

App_Processes is the largest among the tables of the dataset, containing information about each running process in the device at the time the sample was collected, e.g., whether it was a service or an app running on the foreground, its name, and version. Battery_Details provides battery-related information such as whether the device was plugged to a charger or not and the temperature of the battery. Cpu_Statuses indicates the percentage of the CPU under use, the accumulated up time, and sleep time.

Devices provides device-specific information, such as the model and manufacturer of the device and the version of the operating system running on it. Network_Details groups network-related information, e.g., network operator and type, whether the device is connected to a wifi network, and the strength of the wifi signal. The Settings table records multiple yes/no settings for services such as bluetooth, location, power saver mode, and nfc, among others. Finally, Storage_Details provides multiple features related to the secondary storage of the device.

Table 1 presents a more detailed description (excluding primary and foreign keys) of the dataset tables. It includes information on the attribute's name, type, and an example of each.

[17] The dataset includes samples collected prior to that date, but they correspond to the infrastructure testing period.
[18] https://farmer.greenhubproject.org/storage/dataset.7z
[19] http://parquet.apache.org
[20] https://farmer.greenhubproject.org/storage/dataset.parquet.7z
[21] https://laravel.com/

Table 1 Details on the GreenHub dataset tables

Attribute name	Type	Example
App_Processes		
name	varchar	"com.facebook.katana"
application_label	varchar	"Facebook"
is_system_app	tinyInt	0
importance	varchar	"Service"
version_name	varchar	"8.2.0"
version_code	int	802000871
installation_package	varchar	com.android.vending
Battery_Details		
charger	varchar	"unplugged"
health	varchar	"Good"
voltage	decimal	4.03
temperature	decimal	29.20
Cpu_Statuses		
usage	decimal	0.03
up_time	bigInt	409480
sleep_time	bigInt	141369
Devices		
model	varchar	"Nexus"
manufacturer	varchar	"LGE"
brand	varchar	"google"
product	varchar	"hammerhead"
os_version	varchar	"6.0.1"
kernel_version	varchar	"3.4.0-gcf10b7e"
is_root	tinyInt	0
Network_Details		
network_type	varchar	"WIFI"
mobile_network_type	varchar	"lte"
mobile_data_status	varchar	"connected"
mobile_data_activity	varchar	"inout"
roaming_enabled	tinyInt	0
wifi_status	varchar	"enabled"
wifi_signal_strength	int	-71
wifi_link_speed	int	39
wifi_ap_status	varchar	"disabled"
network_operator	varchar	"verizon"
sim_operator	varchar	"unknown"
mcc	varchar	"311"
mnc	varchar	"480"

Table 1 (continued)

Attribute name	Type	Example
Samples		
timestamp	timestamp	2017-10-08
app_version	int	11
database_version	int	3
battery_state	varchar	“Charging”
battery_level	decimal	0.90
memory_active	int	505296
memory_inactive	int	502392
memory_free	int	1442060
memory_user	int	60724
triggered_by	varchar	“android.intent.action.BATTERY_CHANGED”
network_status	varchar	“lte”
screen_brightness	int	-1
screen_on	tinyInt	1
timezone	varchar	“America/Chicago”
country_code	varchar	“us”
Settings		
bluetooth_enabled	tinyInt	0
location_enabled	tinyInt	1
power_saver_enabled	tinyInt	0
flashlight_enabled	tinyInt	0
nfc_enabled	tinyInt	1
unknown_sources	tinyInt	0
developer_mode	tinyInt	0
Storage_Details		
free	int	3922
total	int	9634
free_external	int	3922
total_external	int	9634
free_system	int	637
total_system	int	3390
free_secondary	int	0
total_secondary	int	0

Let us now look at some simple examples of how we can query the GreenHub Farmer dataset within a database, using SQL. We start by looking at what are the top 5 most represented countries:

```
SELECT country_code, count(country_code) as qty
FROM samples GROUP BY country_code
```

The results for this query are presented next.

country_code	qty
pt	4362680
id	4015493
Unknown	2013308
us	1759824
br	1462220

The majority (4.3M) of our samples are from Portugal, followed by India (4M). Roughly 2M samples are from unknown countries.

Out of curiosity, let us inspect how many samples we have, for example, from South Korea:

```
SELECT country_code, count(country_code) as qty
FROM samples WHERE country_code = 'kr'
GROUP BY country_code;
```

From South Korea we have a sample size of $\approx$ 77K:

country_code	qty
kr	77307

Inspecting these samples deeper, we may query the top 10 brands of devices from South Korea:

```
SELECT brand, count(brand) as qty
FROM devices d INNER JOIN samples s ON d.id = s.device_id
WHERE country_code = 'kr' GROUP BY brand
ORDER BY qty DESC LIMIT 10;
```

Interestingly, while we have many more different brands represented in our dataset, only 6 different are represented in South Korea, with a Samsung dominance:

brand	qty
samsung	53132
lge	19540
Sony	2318
HUAWEI	1306
VEGA	931
google	80

Finally, we dig into one particular device, #7485, a Samsung device from South Korea.

```
SELECT model, battery_state, battery_level,
network_status, 'timestamp', bluetooth_enabled,
location_enabled
FROM devices d
   INNER JOIN samples  s  ON d.id = s.device_id
   INNER JOIN settings st ON st.sample_id = s.id
WHERE device_id = 7485;
```

The (partial) results for this query are shown in Table 2.

We can see the user began charging the device when the battery level was at 79% and up to 80%, as `battery_state` changed from `Discharging` to `Charging`. At this time, the WIFI was also turned on. After approximately 5 minutes the user disconnected the charging cable, and turned off the network (neither WiFi nor lte networks were activated). Additionally, we also see how neither the bluetooth nor the location setting was changed during these samples.

Based on sequences of samples such as this, we can, e.g., estimate how much battery was consumed during a given period by looking at `battery_level` and `timestamp`. For example, the last two lines of Table 2 show that it took almost 22 and a half minutes to discharge the battery by 1%, from 79% to 78%. One can combine such samples with the `app_processes` table and understand which applications were involved and correlate the relationship between applications and battery discharges.

3 Characterizing the Collected Data

The previous section detailed the complete construction and definition of the GreenHub infrastructure, in order to support a collaborative environment to collect and share Android usage data. As our GreenHub initiative aims to give back to the community (both researchers and practitioners), our first research question within the initiative is very simple:

RQ_1 Can our collaborative platform collect large, diverse, and representative amounts of Android user usage data across the world?

The direct answer to RQ_1 is yes, and this straight answer is backed up by the empirical study described in this paper (and specifically within this Section), along with the characterization of the data and our findings thus far ($\boldsymbol{ARQ}_1$).

The construction and deployment of the GreenHub infrastructure allowed us to collect a very large dataset with 23+ million unique samples, including more than 700+ million data points pertaining to processes running on mobile devices.

Table 2 Details for device 7485

model	battery state	battery level	network status	timestamp	bluetooth enabled	location enabled
...						
SM-J330L	Discharging	0.79	disconnected	2018-07-08 13:38:10	0	1
SM-J330L	Charging	0.80	WIFI	2018-07-08 13:44:15	0	1
SM-J330L	Discharging	0.79	disconnected	2018-07-08 13:49:53	0	1
SM-J330L	Discharging	0.78	disconnected	2018-07-08 14:12:15	0	1
...						

Within the remainder of this Section, we perform a first characterization of the collected GreenHub dataset in a structured and detailed way, focusing on several of the attributes present in each dataset. For the full visualization on the remainder attributes, please refer to Section 7.

The focus of each characterization effort is to understand how far wide, and how representative is our dataset. Thus we will showcase how fruitful the GreenHub dataset is, and how it can be used to answer relevant research questions, by other researchers, on Android smart-phone systems.

Prior to processing the data, in order to characterize it, we carefully analyzed the distribution of the data values, detected any inconsistent, incomplete, or non uniform data, treated any detected syntax errors, added complementary information, etc,.

For example, the representation of the Android OS version often times differs according to brand/model, and we have cases such as: *Android 8, Android 8.0.0, OS 8,* etc. In these cases, we normalized the representation, thus when we were to aggregate the data for corresponding queries, we would have accurate and consistent results. Additionally, following the same example, we complemented the versions with their corresponding *Codename*, in this case *Android Oreo*, by adding a new column in our dataset pipeline. Another normalization example included String treatment to be all upper-case (once again, each brand/model would present the data in various ways, such as upper/lower/camel-case.) to correctly aggregate data. In addition, during the data cleaning process, we also discarded all statistically calculated PPM values and their periods. Such data cleaning steps are automated and implemented within our GreenHub Pipeline, which is described in Section 7.

In Section 3.1, we characterize the dataset regarding the countries and time zones where the samples have been collected. In Section 3.2 we describe the brands, the models, the operating system versions and respective codenames that are present in the dataset. In Section 3.3 we look at the top 15 most represented public apps within our dataset. Additionally, we also drill-down into 3 popular apps and describe in finer detail the information available about the various processes each app is executing. In Section 3.4 we describe the prevalence of samples with associated sensor usage and within different charging states. Finally, Section 3.5 describes the different network types, operators, network statuses present in our dataset for each sample.

3.1 Samples Data

As of July 2019, the Samples database table had 23,600,501 entries. A characterization of the countries and time zones that are most frequently represented is shown in Fig. 3. In Fig. 3a, we present the 10 (out of 160) countries that have the highest number of entries, which means that they have the highest number of associated samples in the dataset. We also show an aggregation for the number of entries of the remaining countries. The same approach is followed in Fig. 3b for time zones.

We see that the most represented country is Portugal (`pt`), followed by Indonesia (`id`) and the United States of America (`us`). These 3 countries alone represent circa 50% of the samples.

Country code label is available only when the device is registered to a network. Some results may be unreliable on CDMA networks, and for these particular cases, of circa 10% of samples, we label them as `unknown`.

Regarding the most represented time zones, `ASIA/JAKARTA`, `EUROPE/LISBON` and `ATLANTIC/MADEIRA` consist of the top 3. We also see that circa 20% of the samples are coming from time zones whose representation is below the 3.16% threshold.

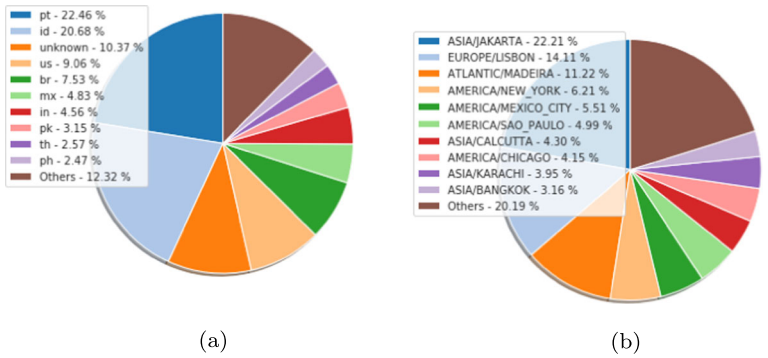


Fig. 3 Distribution of Samples data

3.2 Devices Data

In our Devices dataset, we have 87k+ different represented devices installations which imply that each time a user downloads and installs *BatteryHub*, a unique hash identifier is associated to the device, to track the samples coming for that given installation.[22]

Shown in Fig. 4a we have the top 20 (out of 1.6k+) most represented device brands in our dataset, and shown in Fig. 4b we have the top 20 (out of 11.8k+) most represented smartphone models. We see that *Samsung* has a large representation, accounting for 34% of our entries in the top 20, followed by *Xiaomi* with 10%. The *Samsung* dominance continues with the smartphone models, where the two most represented ones are the *Samsung J200G* and *Samsung G532G* with around 10% in both cases. These are followed by two *Xiaomi REDMI* models with around 8% each.

Our dataset contains devices across 50 different individual Android versions, of which the top 20 are shown in Fig. 4c. Here we see that Android version 7.0 is the most represented, appearing in 14% of our devices, with version 6.0 close behind with 13%. It also contains information on older versions such as 4.4.2, of which 5% of the devices have installed.

Finally, shown in Fig. 4d, we have the distribution of all Android versions by their code name. We see how *Marshmallow* (which are versions 6.0 - 6.0.1) is the most represented with 26% of our devices. *Lollipop* (versions 5.0 - 5.1.1) is the second most represented with 25%. The more recent versions such as *Oreo* and *Pie* have a representation of 12% and 1% respectively.

3.3 App Processes Data

The App Processes database table is by far our largest one, with over 749M data entries. As each Samples entry can have an `N` amount of App Processes running, we have roughly 31:1 (App Processes:Samples) ratio. An App Process can be system processes as well as user apps.

Shown in Fig. 5a are the top 15 represented publicly available user apps in our dataset, which refer to almost 160M entries. This top 15 list excludes any system processes, and apps

[22] In fact, as we will discuss in detail in Section 6, this number includes installations of a clone which was made of BatteryHub, which at least up to a certain version, contributed with data to the dataset.

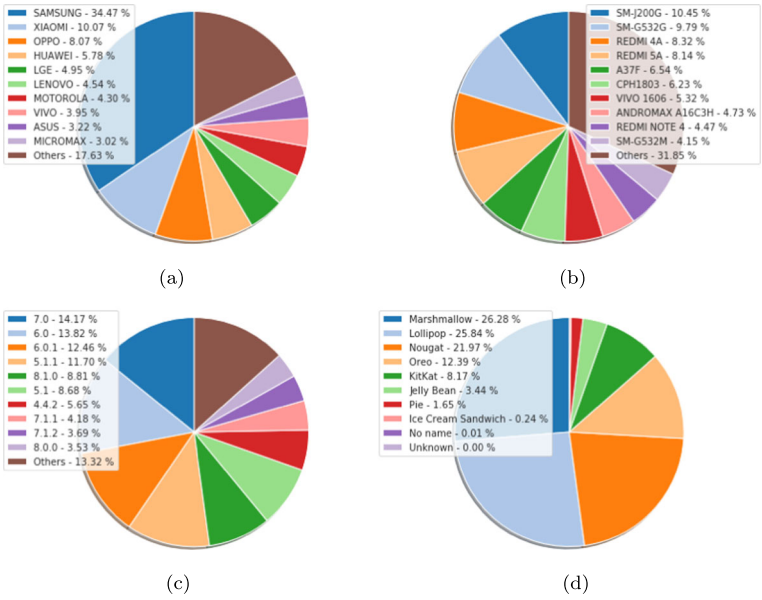


Fig. 4 Distribution of Devices data

which are unobtainable from publicly available app stores, such as the *Google Playstore*, *Aptoide*, etc,. This excluded brand specific app stores (such as *Galaxy Store*). To obtain this list, after counting the number of occurrences of each process within our dataset, we: i) filtered out processes which were defined as a system app (this data is present within the App Processes table under the `is_system_app` attribute) and ii) manually verified if the most represented apps were publicly obtainable by searching for them on the public Android app stores.

Additionally, the list also excludes the processes and apps which send data to GreenHub Farmer. Coincidentally upon thorough analysis of the App Processes data, we discovered that a clone of BatteryHub had a large user-base which was also contributing with samples to the dataset. To properly understand if these data samples can be assumed clean, and thus can be mined, we ran a study which is detailed in Section 6. The conclusions of such study provided strong evidence that the data can indeed be considered valid and correct, and thus are included within our analysis.

Within these top 15, the very popular and common *Google*[23] app unsurprisingly has a large user base, with a representation within the top 15 of 23%. Closely following behind are the *Facebook*[24] and *Messenger*[25] apps with 12% and 11%. This list shares similarities with the *AndroidRank* list,[26] while representing more of the BatteryHub user base's habits.

In Fig. 5b–d, we show another level of information contained in our dataset. Here we show 3 popular social apps, *Facebook*, *Messenger*, and *Instagram*,[27] with the distribution of

[23] Google app: https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox
[24] Facebook app: https://play.google.com/store/apps/details?id=com.facebook.katana
[25] Messenger app: https://play.google.com/store/apps/details?id=com.facebook.orca
[26] AndroidRank: https://www.androidrank.org/
[27] Instagram app: https://play.google.com/store/apps/details?id=com.instagram.android

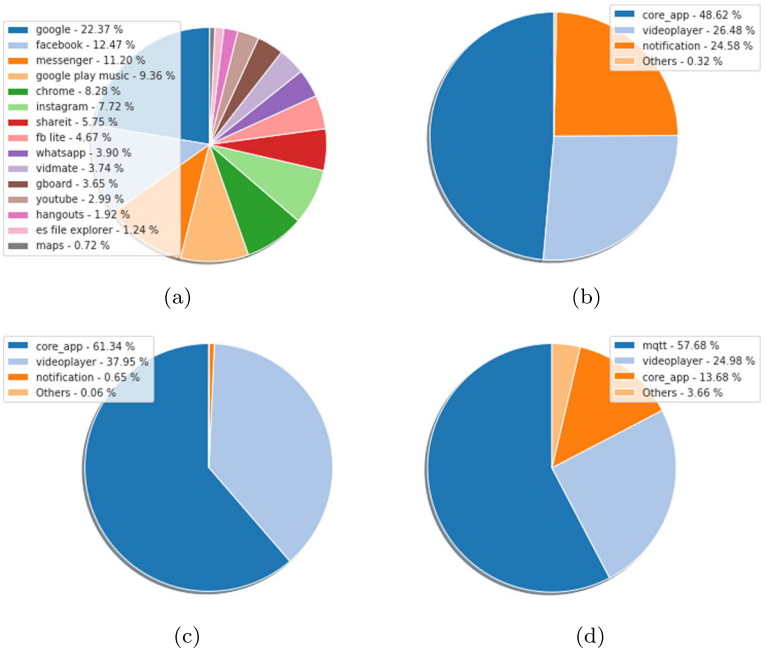


Fig. 5 Distribution of App Processes data

each of their sub-processes, including for example the main process (core_app), video player processes (videoplayer), notification processes (notification), etc. We can see in *Facebook*, the core_app is almost half the representation, with the video player and notification occupying the other half. For *Messenger*, the notification process is under 1%, with a dominance for the core_app (61%) and video player (37%). However for *Instagram*, the mqtt protocol process makes up for over half (57%), followed by the video player (24%), and then the core_app (13%). This finer grain level of information in the real-world can help developers better understand their applications, identify possible problems, and help choose where to focus attentions to optimize.

3.4 Settings and Battery Details Data

Similarly to the Samples database table, and as a consequence of the data model shown in Fig. 2, both the Settings and the Battery Details tables had, as of July 2019, 23.6M+ entries.

In Fig. 6, we characterize our dataset under the perspective of how often popular sensors are active in the collected samples (Fig. 6a–e). We also show how different charging states are represented (Fig. 6f).

The vast majority of the samples that we collected occurred when Bluetooth, NFC or power saving mode were turned off, with a percentage of circa 85%, 90% and 95%, respectively (Fig. 6a-c). This was, however, not the case for the location sensor, which was active in circa 40% of the instants in which a sample was collected (Fig. 6d).

Figure 6e focuses on the samples that, when collected, had at least one sensor active, which correspond to 16.9M+ samples, circa 70% of the dataset. We see that circa 60% of them had at least location enabled, 20% bluetooth, 14% NFC and 8% powersaver.

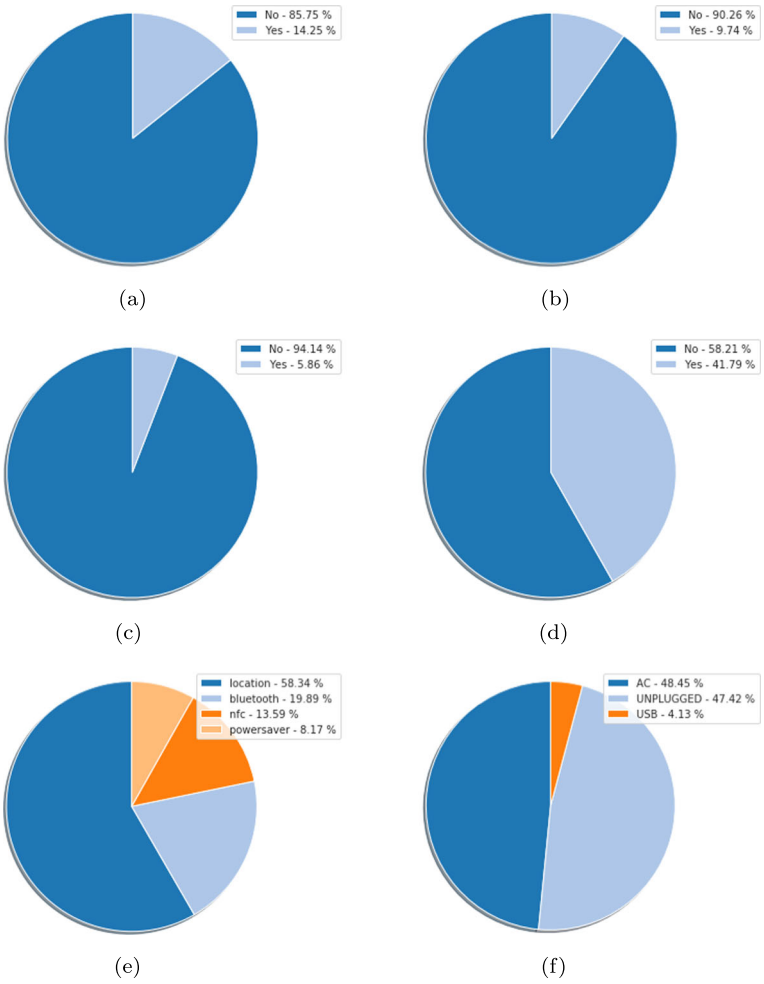


Fig. 6 Distribution of Settings data (**a**-**e**) and Battery Details (**f**)

Finally, Fig. 6f shows that the samples in the dataset were marginally collected, circa 4% of the times, when the device was charging plugged into an USB port. The remaining samples were, quite evenly, collected when the device was either unplugged of any power source or plugged to an AC source.

3.5 Network Details Data

Some entries were dropped from the Network Details data due to the fact that there were missing values in a few particular columns, invalidating those specific entries. In other cases, the values received from the devices, at the time of mapping and storing into the dataset, were incompatible with the expected data types therefore those were considered outliers and consequently dropped at that stage. In practice, the Network Details table included 23,533,845 entries.

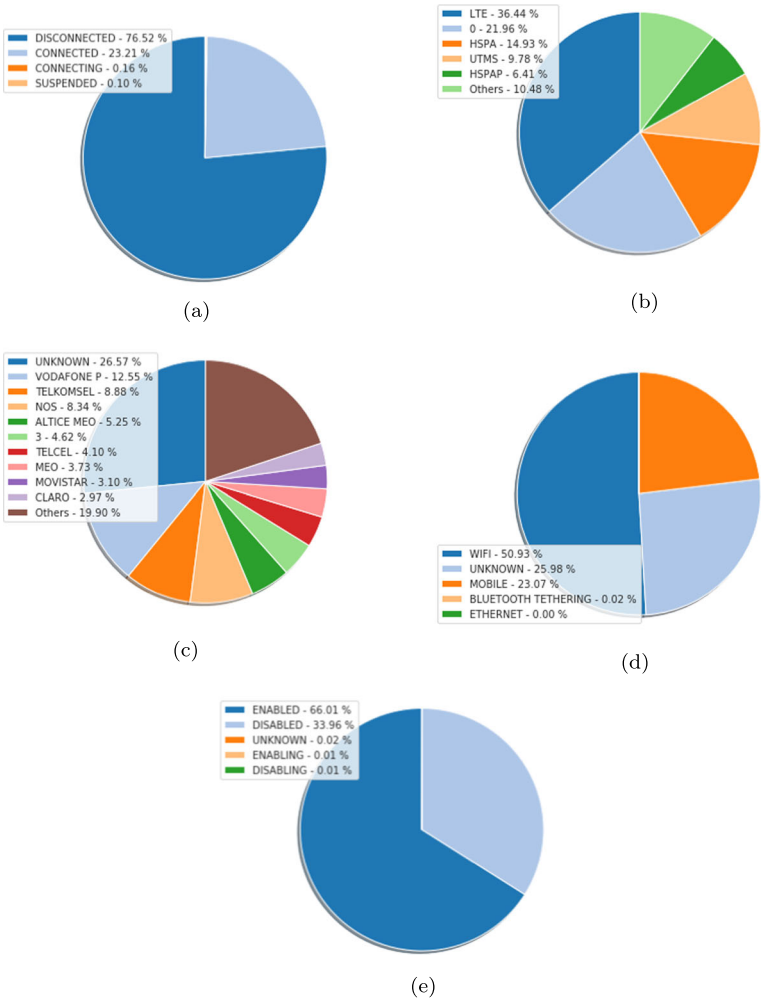


Fig. 7 Distribution of Network Details data

Figure 7 presents the characterization of our dataset regarding the collected Network Details dataset for each Sample (for instance, if Wi-Fi is connected or not).

More than 76% of our entries were collected during a time when the mobile data connection was disabled. As for Wi-Fi, 66% of our samples had it enabled. This can be respectively seen in Fig. 7a and e.

In Fig. 7d, we have an overview of the data regarding samples collected when there was an active connection, either Wi-Fi or mobile data. We can see that the majority (50%) used Wi-Fi as the active connection, while mobile connections represent around 23%. We were even able to obtain samples where a Bluetooth tethering connection was on, and in nearly 26% of the times we were not able to obtain data on what was the type of the active network connection.

In order to represent information related to the mobile data connection, we created the plots in Fig. 7b and c. Specifically, the former contains the percentage of times that specific

mobile data connection type was found on a collected samples, and the later represents the distribution of the top 20 network operators. As we can see, there is no dominant network operator in our samples, and the top 20 does not represent all the existing mobile operators. Also, in a considerable amount of samples we could not detect what was the name of that operator (circa 26.5% of the times). Regarding network types, *4G LTE* and *HSPA* (a.k.a. 3G) are the most dominant (36.44% and 14.83%, respectively), and again we have a considerable amount of samples (almost 22%) with an unknown connection type.

4 Battery Consumption Analysis of Android Devices

In this section, we explore the GreenHub dataset on one specific avenue of research, which targets battery consumption within the Android ecosystem. Our focus here is well aligned with the goals and ambitions of the GreenHub Initiative.

Thus, we present several research questions (RQ), in addition to their motivation and interest for users, researchers and practitioners, that we aim to answer by using our dataset. These following 14 RQs focus on understanding the impacts and tendencies on battery charging/discharging based on real-world usage, considering various factors inherent of such real usage.

Additionally, these RQs are focused on the need to provide further knowledge on the topic of battery consumption for researchers, users, and practitioners. A distinctive characteristic of these RQs, when compared with RQ1, is that they are explicitly exploratory in nature. By this we mean that, while their answer may substantially address the lack of knowledge based on real-world data in this field, they highlight tendencies whose further exploration may often times suggest further dedicated studies. As pointed out by Pinto and Castor (2017) and Manotas et al. (2016), one of the major roadblocks to energy efficiency software development, hardware development, and awareness by users, is the lack of knowledge.

RQ_2 Do battery charge/discharge tendencies differ between countries and timezones? *While this RQ might not have a direct benefit for users, researchers, nor practitioners, it is an interesting curiosity which we can try to answer with our data. As the nature of our study is descriptive and qualitative, we can highlight such associations between these real-world factors and battery usage in order to further obtain knowledge on this topic. In addition, looking at such associations may actually open up new research paths if there seems to be a pattern in the data worth exploring.*

RQ_3 Are there observable battery charge/discharge tendencies differences across brands? *Users can use such answers to help them choose their preferred brand if battery consumption is of a concern, while manufacturers/practitioners can understand how their brand stands up to the competition. Researchers can take advantage of such data and try to understand what actually leads to such observable differences between brands.*

RQ_4 Are there observable battery charge/discharge tendencies differences across models? *Users can use such answers to help them choose their preferred model if battery consumption is of a concern, while manufacturers/practitioners can understand how their new models stands up to the competition. Researchers can take advantage of such data and try to understand what actually leads to such observable differences between smartphone models.*

RQ_5 Does the battery charge/discharge tend to improve with new Android versions (minor updates)? *For a user, understanding how the battery consumption tends to be affected between minor Android versions can give them more knowledge when deciding if they wish to go ahead with an update now (battery tends to improve), or hold off for major updates (if battery tends to worsen). For Android developers and researchers, new paths open up allowing them to drill-down and try to understand what exactly makes it so such tendencies occur. If battery tends to worsen, they may try to understand what programming choices were made as to avoid them, or if battery improves, in order to further replicate them.*

RQ_6 Does the battery charge/discharge tend to improve with new Android codenames (major updates)? *The motivation for this RQ is similar to the previous one, just in a more general sense based on major Android versions. Users may wish to wait for minor version updates if they know major updates have a tendency to worsen the battery, or vice-versa. Researchers and developers may further use this knowledge in order to reflect on programming choices in order to better the battery consumption.*

RQ_7 What are the battery charge/discharge tendencies when the most popular apps are present? *The answer to this RQ will help users obtain more knowledge on which of their applications have a tendency to be present when their battery consumption is worsened, and thus search for alternative and more efficient apps.*

RQ_8 Can developers use this dataset to understand if their apps, also when compared with others, appear more often in good/bad battery charge/discharge tendencies? *Developers too are in need of understanding how their app compares to others. Oftentimes, it is difficult for developers to obtain information on battery consumption tendencies of their applications from real usage. As this dataset represents a large amount of users, it may help developers search for information of their application's battery tendencies and compare it to similar apps.*

RQ_9 Can this dataset help developers identify possible battery hogging tendencies and miss-managed processes? *If a developer knows there is a battery inefficiency issue in their app, it may be difficult to understand what may be triggering such occurrences. The problem may come from specific battery hogging processes, or even conflicts with other apps. Understanding if our dataset contains enough fine-grain data to allow the developer to help pinpoint such problems will help these practitioners.*

RQ_{10} Is there a clear and obvious battery charge/discharge tendency between having Bluetooth ON or OFF? *Understanding how Bluetooth tends to affect battery consumption will help users have more awareness on if such a setting should only be turned on when needed, if it may be left on indefinitely or with little importance, or if they should be avoided. Software developers may use this information to know if they should reduce their apps Bluetooth usage (if they wish to make a battery efficient app), and hardware developers and researchers may explore how to improve the Bluetooth battery usage if it indeed has a tendency to be inefficient.*

RQ_{11} Is there a clear and obvious battery charge/discharge tendency between having Location ON or OFF? *The motivation for this RQ is comparable to RQ_{10}, but focused on Location services*

RQ_{12} Is there a clear and obvious battery charge/discharge tendency between having NFC ON or OFF? *The motivation for this RQ is comparable to RQ_{10}, but focused on NFC.*

RQ_{13} Does having power saver ON helps with battery charging/discharging? *The motivation for this RQ is comparable to RQ_{10}, but focused on the Power saver setting.*

RQ_{14} How do the different network types tend to affect the battery charge/discharge? *The motivation for this RQ is comparable to RQ_{10}, but focused on different network types.*

RQ_{15} Are there visible differences between battery charge/discharge tendencies according to different network operators? *The motivation for this RQ is comparable to RQ_{10}, but focused on different network operators.*

In order to address the research questions of focus in this work, and to understand how our data on battery discharging and charging relates to usage within the real-world, we have defined a battery tendency metric called *Percentage Per Minute*, or PPM. The following section will further explain the GreenHub PPM metric.

4.1 Battery Metrics in GreenHub - PPM

Prior to calculating, or even defining, the PPM metric, we first defined the notion of a *period slice* within the Farmer data. This defining of a *period slice* is important, as the data reaches Farmer in an asynchronous manner due to the large amount of devices from which data is being collected from.

A *period slice* is essentially a timely sequence of our battery samples belonging to the same mobile device, and follow the same direction (either charging, or discharging).

Additionally, when analyzing the distribution of deltas in battery level between consecutive samples, shown in Fig. 8, we found that in 95% of the cases, the recorded battery level delta varied between [-1,1]%, and in 98% of the cases it varied between [-2,2]%. As to preserve the most amount of data, while also removing outliers, we decided on maintaining the

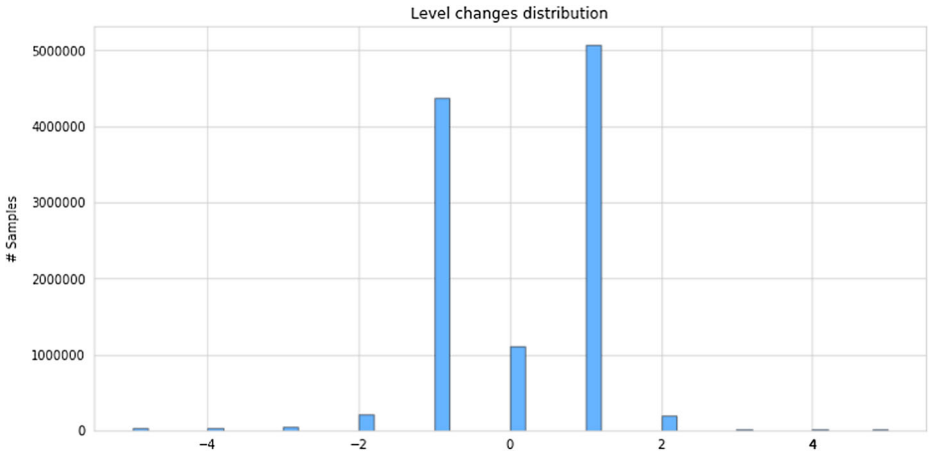


Fig. 8 Battery level change distribution of Samples

later. Thus, a new period is also considered if the battery level changes less than 2% or -2% while charging or discharging, in the given direction, respectively.

Therefore, the following rules are applied to define if the sample belongs in the same *period slice*. These are applied after sorting our `Samples` data according to the `device_id` and then the `timestamp`. If the condition fails at any rule, a new *period slice* is considered:

1. The sample is of the same device;
2. The battery level follows the same direction as the period's direction;
   - If the battery level change is 0, the same direction is followed
3. The battery level change is within threshold limits [-2,2];
   - If battery direction is charging, then change limit ≤ 2
   - If battery direction is discharging, then change limit is ≥ -2

Shown in Fig. 9, are examples of the 3 possible scenarios of defining a new *period slice*. As previously stated, the data is sorted according to the *device_id* and *timestamp*. The *sample data* column represents the various data columns present within a `Sample`, the *battery_level* represents the battery level of the sample at that given instance, and the *period* represents the period number given to the group of samples from the same *period slice*.

Here we see 3 different occasions where a slicing occurred after `A`, `B`, and `C`. After `A`, a new *period* (56) was created as the device went from a discharging to a charging state, and thus the battery changed direction. Another *period* was created after `B`, since the data is of a completely new device (device 6612), and thus the first rule took effect. Finally, our last *period slice* occurred after `C` since the battery level changed from 34% to 99%, exceeding our threshold of [-2,2]. Such a case occurs when, for example, a user turns off their smartphone or stops using it for a very long period of time.

After slicing the data into appropriate *period slices*, we filtered out those which have less than 10 samples, and more than 100 samples. The lower sample limit for each *period*

	device_id	timestamp	sample data	battery_level	period	
	5567	2019-1-07 13:14:10	···	88%	55	
	5567	2019-1-07 13:15:58	···	87%	55	
	5567	2019-1-07 13:17:20	···	86%	55	
			⋮			
A	5567	2019-1-07 13:28:05	···	70%	55	Rule 2: battery changed direction (charging)
	5567	2019-1-07 13:29:15	···	71%	56	
	5567	2019-1-07 13:32:11	···	73%	56	
			⋮			
B	6612	2019-1-07 17:11:05	···	15%	88	Rule 1: different device
	6612	2019-1-07 17:13:05	···	16%	88	
			⋮			
C	6612	2019-1-07 17:32:05	···	34%	88	Rule 3: battery level changed out of threshold [-2,2]
	6612	2019-1-07 17:50:12	···	99%	89	
	6612	2019-1-07 17:52:12	···	98%	89	
			⋮			

Fig. 9 Examples of period slicing

allows us to have *periods* of a significant size, while the upper sample limit of 100 is roughly around where *periods* begun to have a low representation and become outliers.

Currently, these are the rules used to define the *period slices*, which can be further easily expanded upon if needed a more strict slicing or sub-slices.

With these rules and conditions, we currently have 552k+ periods with an average of 32 samples per period and min/max of 10/100 respectively. Of these 552k, we have 274k charging periods and 278k discharging periods.

Having constructed *period slices*, we can define PPM as the charging/discharging rate of a given *period* based on the amount of time within that period and how much the battery level changed. This can be formulated as:

$$PPM_p = \frac{\mid (battery\_level_{p_n} - battery\_level_{p_0}) \mid}{(time_{p_n} - time_{p_0}) \div 60}$$

where,

$$\begin{aligned} p &= \text{the given period;} \\ battery\_level_{p_0} &= \text{the battery level of the first sample in the given } p \text{ period;} \\ battery\_level_{p_n} &= \text{the battery level of the last, or n}^{th}\text{, sample in the given } p \text{ period;} \\ time_{p_0} &= \text{the timestamp of the first sample in the given } p \text{ period;} \\ time_{p_n} &= \text{the timestamp of the last, or n}^{th}\text{, sample in the given } p \text{ period;} \end{aligned}$$

Looking back at our example in Fig. 9, the PPM for *period* 55 would be:

$$PPM_{55} = \frac{\mid (88 - 70) \mid}{(13h28m05s - 13h14m10s) \div 60} = \frac{\mid (18) \mid}{(835s) \div 60} = 1.29\ \%/min$$

After calculating the PPM of each *period slice*, we used a z-score (Shiffler 1988) test to detect the PPM outliers present in the PPM data, using the standard threshold of 3.

The z-score was individually calculated for both the charging and discharging PPM values, respectively. This is because the rate at which the battery charges and discharges, and how a user uses their device when charging or discharging, highly differs. With this, we removed 1,251 charging and 2,004 discharging *periods*.

To note, a charging *period* and a discharging *period* in our analyses are synonymous to a positive PPM^+ and negative PPM^-, respectively. It is preferable to have a PPM^+ value to be as high as possible, which indicates a better charging percentage-per-minute. On the other end, it is preferable to have a PPM^- value to be as low as possible, as the higher it is means it is discharging faster per-minute. Additionally, a *period* has a PPM value attributed to it. Within a *period* we have a list of *samples* where each sample contains information from its' *device*, *battery details*, *network details*, *settings*, and the list of the *app processes* as shown in Fig. 2.

4.2 Characterization with PPM

It is highly important to understand that the interpretation of a PPM value should not be seen as an absolute metric to judge the (energy/battery) efficiency of a given device, country, OS version, or any other attribute being analyzed. It indicates the *tendency* that a single (or group of) characteristic has on the battery, factoring in all the wide ranges of (secondary)

factors which exist, such as real-world human interaction, varied user usage profiles, and vastly different devices.

Let us take into consideration, for example, a video viewing app which is shown to have a consistently higher reported PPM$^-$ value than others. This indicates that this app has, across a broad range of samples and across varied usage patterns (i.e., devices, brands, setting configurations, operation systems, background/foreground apps, etc., essentially across all the possible characteristics which differ across devices), a tendency to be present when there is a higher discharge pattern in smartphones. This, however, does not indicate that the app in question is without a doubt inefficient, but that there is a high possibility of it contributing to such inefficient tendencies. If it is consistently showing such patterns across highly varied settings, it indicates that it is present within such inefficient usage scenarios, and its presence may be bringing about underlying inefficiency problems. In other words, PPM does not intend to, nor is capable of, pointing to a particular concrete characteristic (i.e. a specific application, sensor, model) and directly "blaming" it for battery inefficiency (or efficiency). Instead, it reveals their tendencies on the battery when present, considering all the wide range of real-world usage factors. However, these tendencies of concrete characteristics can help researchers and practitioners begin exploring further dedicated studies to confirm/contradict the responsibility of the given characteristic, through isolating and comparing with other directly comparable usage samples.

The following sub-sections will detail some of the interesting results of various attribute's PPM$^+$ and PPM$^-$ values.

Naturally, real-world usage of smartphones vary greatly, depending on a vast amount of factors which may affect the PPM values. Thus, to better represent and interpret the results, they are shown as violin-plots in order to not skew nor bias the data. This also allows us to have a better understanding of the data, and show all relevant information (i.e. PPM outliers, quartiles, medians, densities, and distributions) which one should look at when concluding findings. In addition, as PPM is to be understood as a tendency, looking at the $3^{rd}/1^{st}$ quartile ranges and the median, we can better understand how one characteristic tends to influence the battery through comparison of such values.

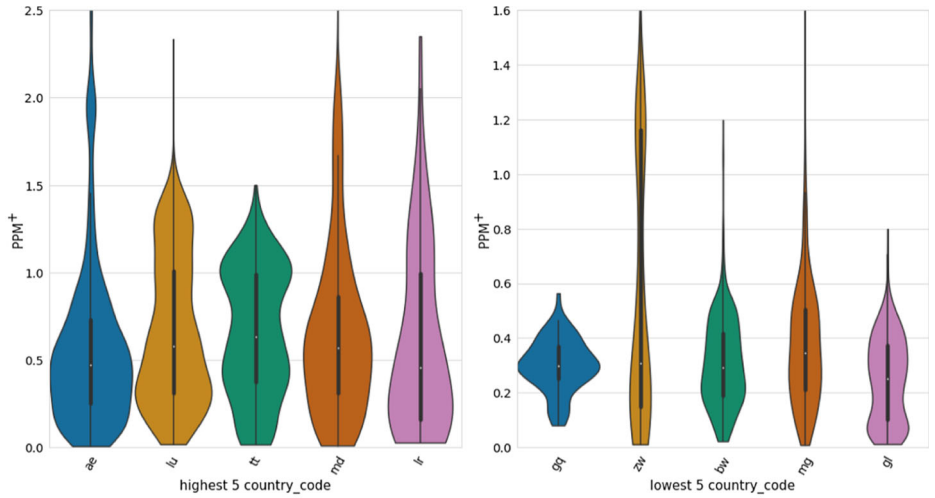
Finally, as to facilitate the visualization, we removed the extreme outliers shown in the plots. The full images can be viewed in higher detail, along with more data, such as count, average, and standard deviations, in each of the notebooks on our GitHub page.[28]

As previously mentioned, the characterization we make is complemented with possible research paths which deserve dedicated research studies.

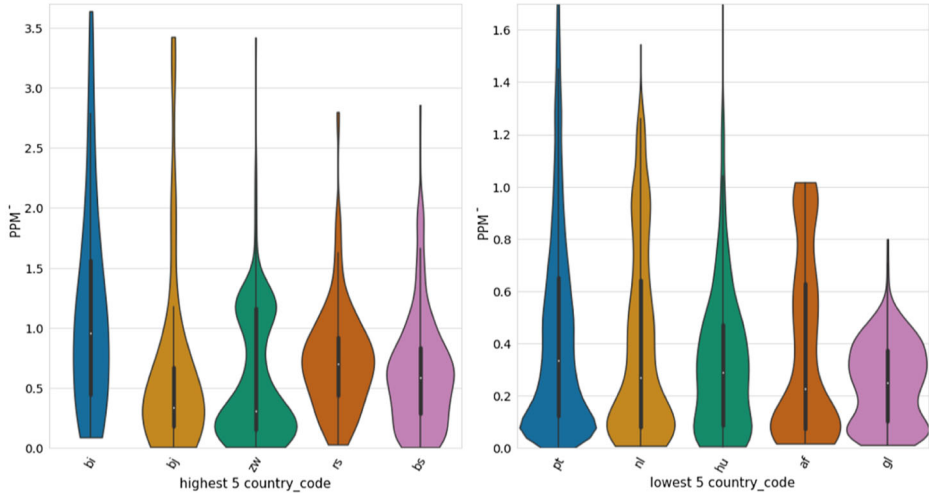
4.2.1 Samples PPM Data

Shown in Fig. 10 are the PPM results for a Sample's country and timezone. Fig. 10a and c show the PPM$^+$ results, while Fig. 10b and d show the PPM$^-$ results, for the highest and lowest 5 (based on their average PPM) countries and time zones respectively. As we quickly notice, there is indeed clear differences between the tendencies based on the PPM results across both countries and time zones ($\boldsymbol{ARQ}_2$). The scale in both cases also varies, where between the highest 5 and lowest 5 countries/time zones, it can go between roughly 1 PPM and roughly 0.2 PPM.

[28] Our GreenHub Pipeline notebooks are detailed in Section 7. The notebooks can be found at https://github.com/greenhub-project/notebooks

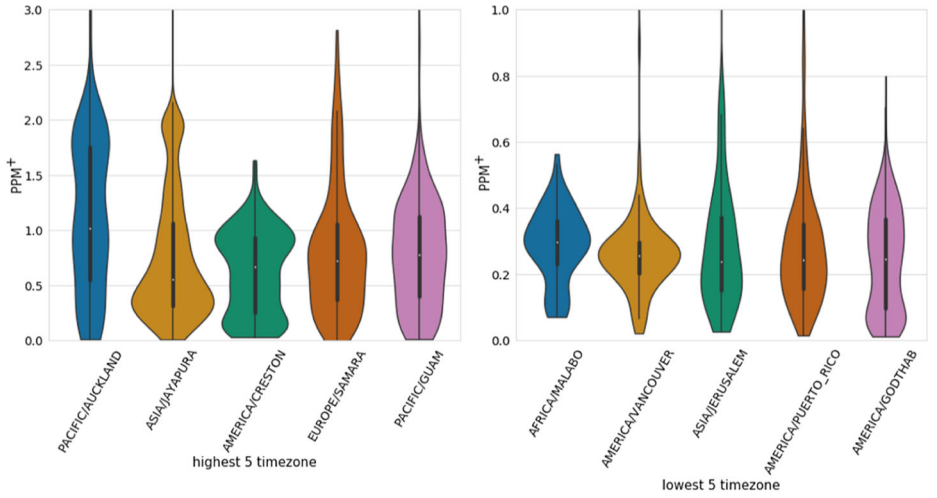


(a)

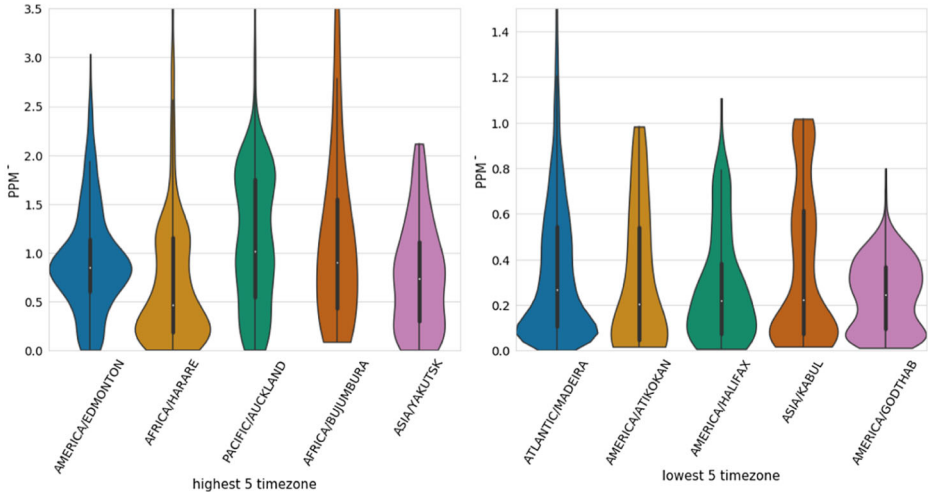


(b)

Fig. 10 Samples PPM Data



(c)



(d)

Fig. 10 (continued)

Let us look closely at Fig. 10a and b for the *lowest 5 PPM*$^+$ and *highest 5 PPM*$^-$ (which equate to the lowest 5 charging, and highest 5 discharging PPMs) respectively. Between these two groups, we see countries such as: *Equatorial Guinea*, *Zimbabwe*, *Botswana*, *Burundi*, and *Benin*, which according to the United Nations are developing or least developed countries.[29,30] This might mean that there is less access to more quality and modern smartphones, which could affect the PPM$^-$, or electrical infrastructures for charging, affecting the PPM$^+$. Another interesting thing to point out is how *Greenland* is both in the lowest 5 PPM$^+$ and the lowest 5 PPM$^-$, meaning devices in this country seem to have a tendency to both discharge and charge slowly. This could also be due to the temperatures in the country (as it is already known that colder temperatures help in battery discharging rates).

4.2.2 Devices PPM Data

Shown in Figs. 11 and 12 are the PPM$^+$ and PPM$^-$ results for the Device dataset's top 5 brands (Fig. 11a), model (Fig. 11b and c), OS version (Fig. 12a and b), and OS version codename (Fig. 12c). In regards to RQ_3, we see there are indeed differences in the PPM values between the top 5 different brands ($\boldsymbol{ARQ}_3$). For example, looking at Fig. 11a on the left-hand side, the *OPPO* brand of devices tends to have a slower charge-per-minute, closely followed by *LGE*, with *XIAOMI* having the highest median PPM but also the highest variance. On the right-hand side, *OPPO* tends to be the faster discharging-per-minute brand, with *HUAWEI* being the slowest. Do note once again that these results reflect not the energy efficiency of each, but is a proxy which equates to the tendency each has on battery charge/discharge during real-world usage, and factors such as different user profiles and types of applications vary between brands.

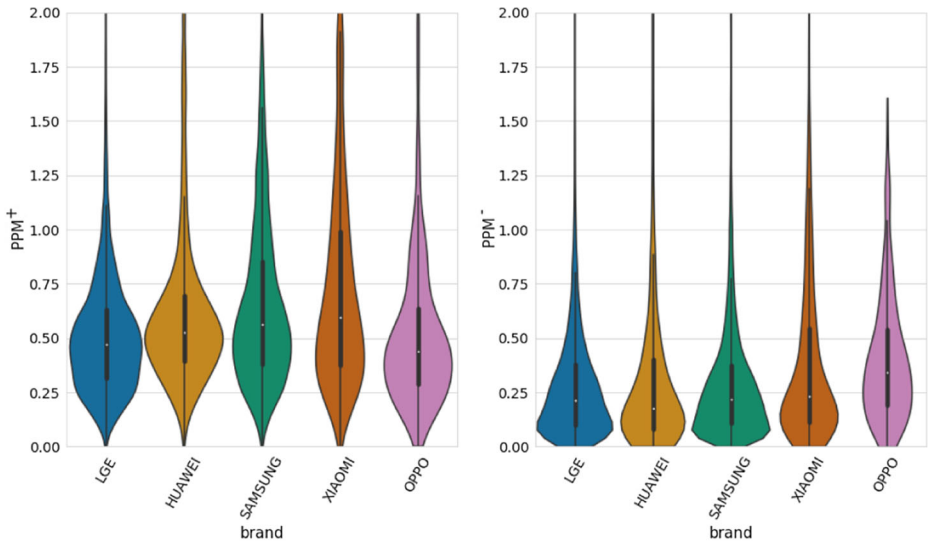
Shown in Fig. 11b and c, we can yet again see the very different PPM values between models ($\boldsymbol{ARQ}_4$). Differences even more so evident than between brands.

In Fig. 12a, we see the different PPM values across minor Android versions. This data shows that, in a general sense, more recent versions tend to have better charging rates (PPM$^+$), and better discharging rates (PPM$^-$). Nevertheless, we see other interesting observations such as how sometimes the PPM between two consecutive minor versions can actually degrade, for example version *5.1* has a better PPM$^+$ than version *5.1.1*, and version *6.0.1* has a better PPM$^-$ than version *6.1* ($\boldsymbol{ARQ}_5$).

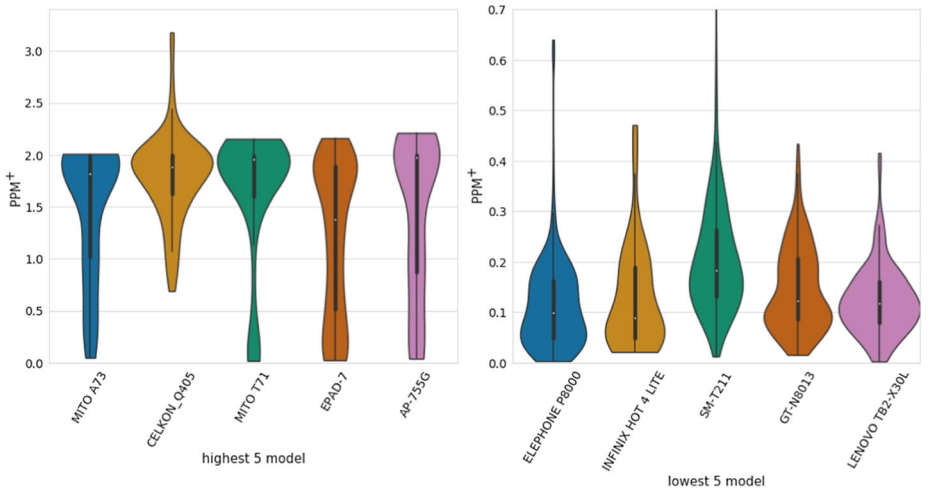
The results are however very promising, as it shows that in terms of major Android versions, there seems to be a tendency for battery usage improving as we can see in Fig. 12c, where the codenames are ordered according to release date. Here we see there was a tendency for an improvement after each version for battery charging, peaking at *Oreo*. For battery discharging, it began to become slightly faster, and then slowly reduces yet again before stopping roughly in the middle with *Oreo* ($\boldsymbol{ARQ}_6$).

RQ_3 is particularly interesting for manufacturers, as it allows them to understand how their smartphones compare to competitors, and how their own different models compare with each other. A further step to explore would be if such differences are due to factors such as hardware. On the other hand, RQ_4 shows Android OS developers that there are various differences between each Android version. Often times, an update (major/minor) can affect battery performance in ways they were not expecting. Having access to this data, they can

[29] https://unstats.un.org/unsd/mi/worldmillennium.htm
[30] https://unstats.un.org/unsd/mi/ldc.htm

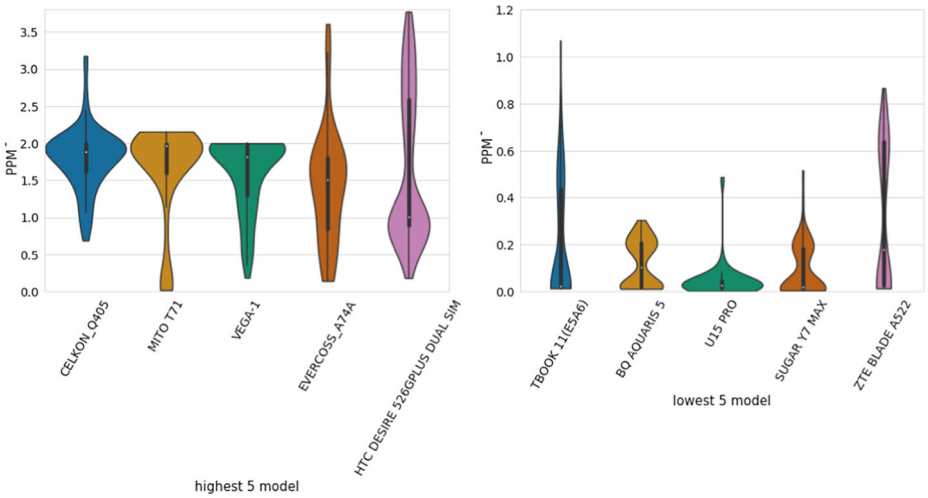


(a)



(b)

Fig. 11 Devices PPM Data



(c)

Fig. 11 (continued)

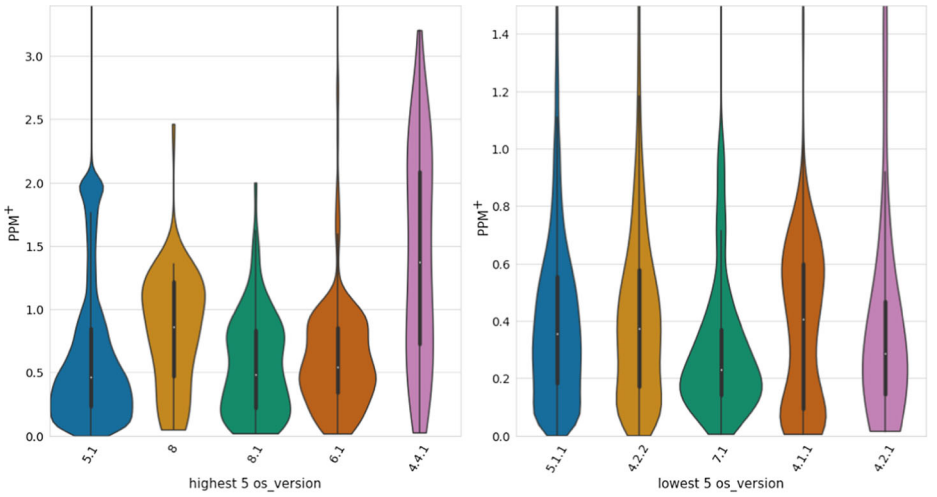
try to understand what exactly happened between such versions to affect their performance in such a way.

4.2.3 App Processes PPM Data

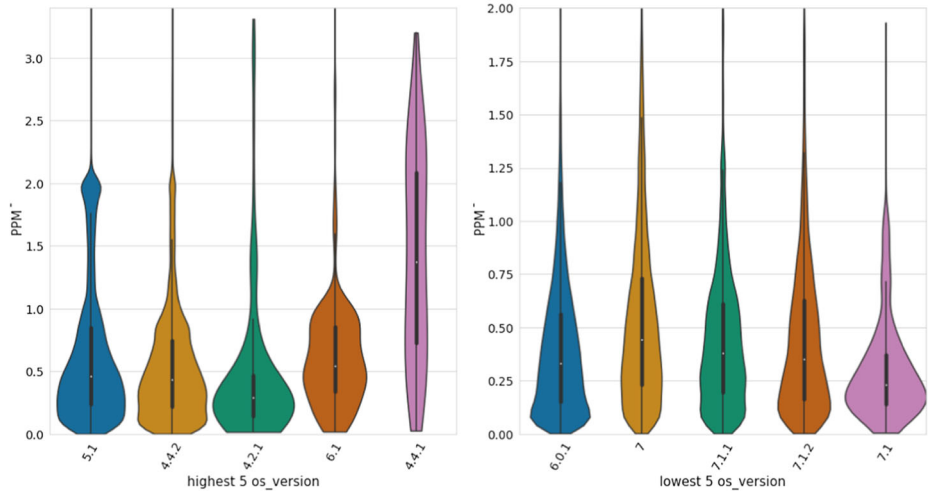
In this section we will look at the PPM results for our top 15 most represented apps within Farmer and the App Processes dataset, and drill-down into 3 popular social platform apps. Shown in Fig. 13, we see the PPM^{+} and PPM^{-} results for our top 15 most represented apps. On the left hand side of Fig. 13, we see that *Google Maps*, *Youtube*, and *VidMate* tend to have a lower PPM^{+} compared to the rest. It isn't too shocking as these are a GPS/Navigation app, and Video viewing apps. What does seem interesting is how *Messenger*, *Facebook*, and *Facebook Lite* actually tend to have a better PPM^{+}. These apps are infamous for being battery hungry.[31] A hypothesis, as we are measuring tendency of battery consumption with human-interaction, could be that these apps are often times active (in background) while being charged, while *Google Maps* for example, might be more often in foreground (used temporarily for driving navigation) when being charged ($\boldsymbol{ARQ}_7$).

On the right hand side of Fig. 13, we see that *Chrome* tends to have the lowest PPM^{-}, even though it is the 5th most represented app in our dataset. This can mean that it may be efficient, or that users spend less time (quickly using the browser for a quick search for example) within the app, and in turn consumes less. Another very interesting observation is *Facebook Lite* is supposed to be a lighter app than *Facebook*, but has a much higher PPM^{-} value. A hypothesis might be that users who tend to use *Facebook Lite* might already have battery issues or older phones which cannot support the *Facebook* app, and thus opt in for the lighter version. This type of information we present can help both users decide on alternative applications if battery is a concern based on tendencies, while also helping developers understand how their app tends to perform in the real-world (continuing $\boldsymbol{ARQ}_7$).

[31] https://www.nytimes.com/2017/11/24/technology/personaltech/facebook-battery-drain.html

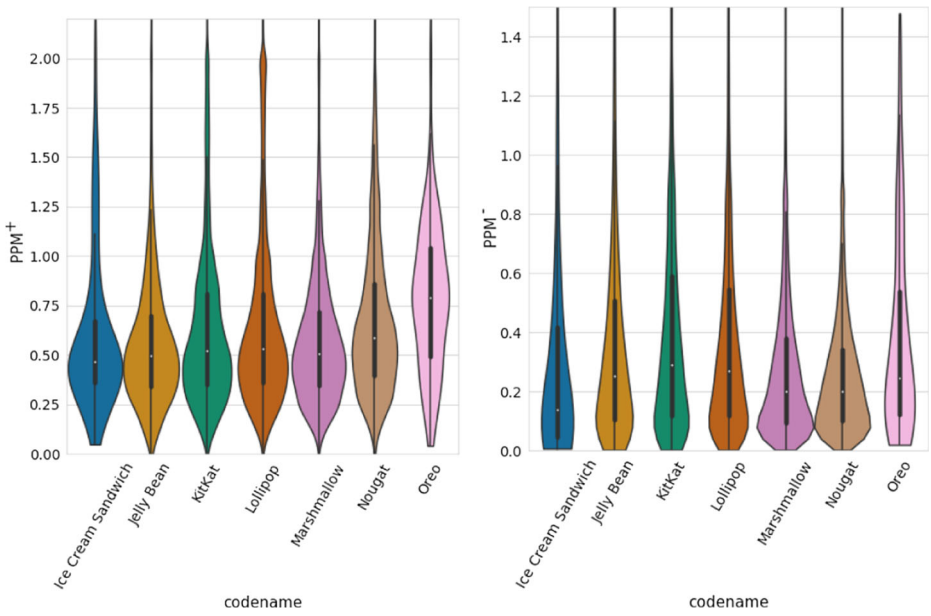


(a)



(b)

Fig. 12 Devices PPM Data (cont.)



(c)

Fig. 12 (continued)

We showed with RQ_7 how our dataset can provide users with deeper information on the tendency of the battery consumption when using their apps, providing them with the knowledge to influence their consumption patterns. But the dataset can also be very impactful for app developers, allowing them to compare their own apps' battery consumption tendencies against others (in a real setting) (RQ_8), and even detect battery hogging processes (RQ_9), as a means to know where to possibly optimize.

Focusing more on a developer oriented analysis, let us see how the GreenHub data can answer such questions. Fig. 14a–c shows the PPM^+ and PPM^- values of the top 5 various sub-processes that three very popular applications have: *Facebook*, *Messenger*, and *Instagram*. Such data allows a developer to understand how each of their app's sub-process impacts the battery out in the wild. For example, *Messenger*'s core_app tends to have a PPM^- value roughly about the same as the other 4 sub-processes, while in *Facebook* the core_app has the second highest PPM^-, slightly below optsvc. When comparing the similar sub-processes between the 3 apps, we see all have the videoplayer sub-process with high PPM^-, which means that when it is present the battery drains faster.

Upon closer inspection, we notice that while in *Facebook* and *Messenger*, videoplayer is rated lower than the core_app. But in *Instagram*, it is the highest PPM^- sub-process. This could mean that the videoplayer isn't properly managed within the application, when compared to the others. When crossing these results with Fig. 7, we found that for both *Facebook* and *Messenger*, the videoplayer accounts for only 26.48% and 37.95% of the samples compared to 48.62% and 61.34% of the core_app samples. *Instagram* on the other hand has 24.98% and 13.68% of the samples for the videoplayer and core_app respectively. For every 1 core_app *Instagram* process, there are 2 videoplayers. This might be another sign of improper videoplayer management.

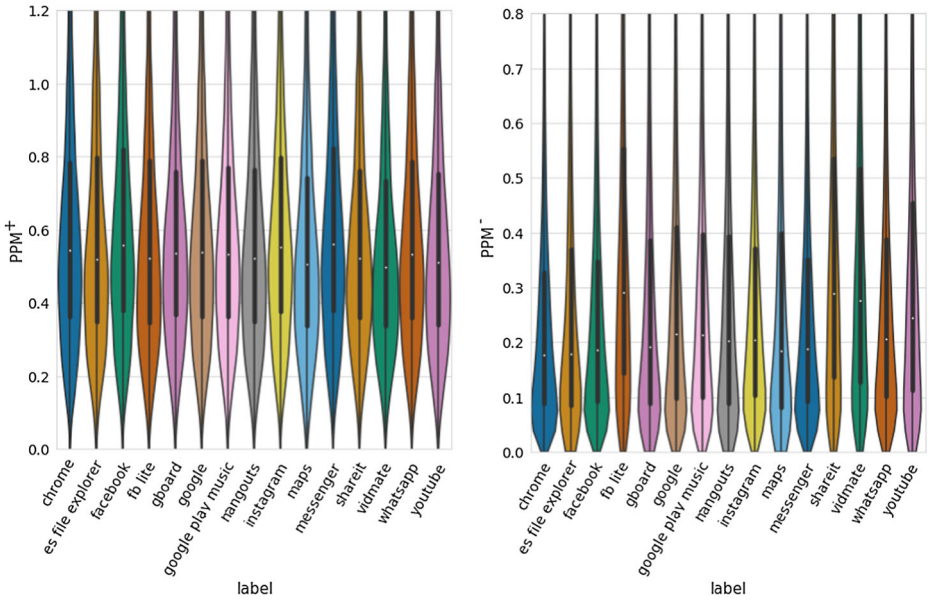


Fig. 13 Top 15 Represented Apps PPM$^+$

As shown, the finer grain data which GreenHub contains can be used to help developers both compare their apps with others ($\boldsymbol{ARQ}_8$), and even drill-down into their apps to help find possible problems ($\boldsymbol{ARQ}_9$).

4.2.4 Settings and Battery Details PPM Data

In this section we have, in Fig. 15, the PPM comparisons between samples that have a certain setting on/off. For instance, in Fig. 15a we compare the PPM of all samples with Bluetooth enabled against samples where it was disabled. The same happens for location services (Fig. 15b), NFC (Fig. 15c), and the power saving setting (Fig. 15d). We also included the same comparison for samples collected when the device was connected via USB, charging or unplugged (Fig. 16).

As the plotted data suggests, for Bluetooth and location there tends to be almost no difference between having those settings enabled or disabled, when it comes to the charging rate (PPM$^+$). Even when considering the discharge rate (PPM$^-$), we see only a slight variation in the data, but with no clear tendency as the median is almost identical ($\boldsymbol{ARQ}_{10}$ **and** $\boldsymbol{ARQ}_{11}$). This may imply that the devices which provided such samples might be leveraging significantly from the *low-usage* states of both these settings.[32]

A similar observation to the previous one can be made for the NFC and power saving settings. Yet this time we see that, in relation to the PPM$^+$, a slight tendency can be observed, in favor of this settings being enabled ($\boldsymbol{ARQ}_{12}$ **and** $\boldsymbol{ARQ}_{13}$). Considering the power saver setting in particular, it makes sense that while ON, the device will charge faster, as it normally stops resource-greedy background services and reduces the screen brightness. However,

[32]It is common that, when such components have a low data throughput, the device automatically puts them under a low level of usage or even idle state, limiting its capacity but saving resources.

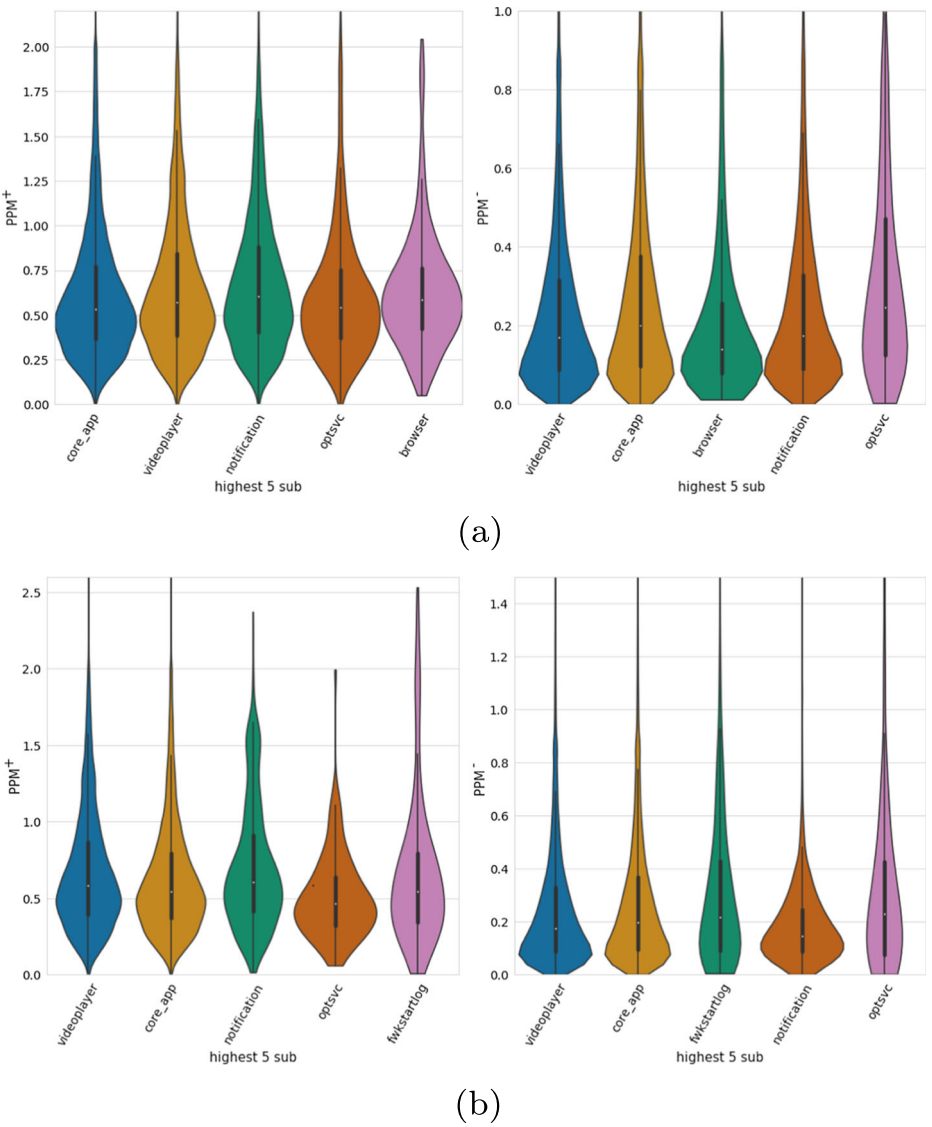
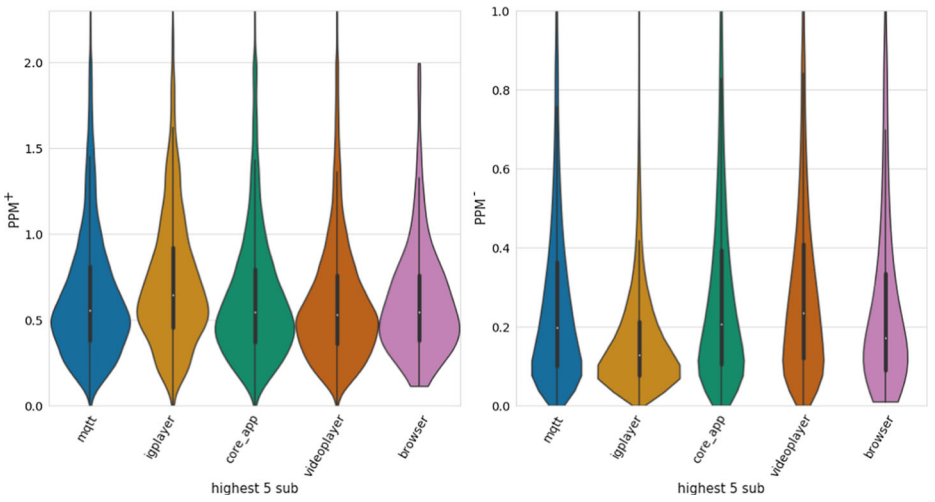


Fig. 14 App Processes PPM Data



(c)

Fig. 14 (continued)

when the battery is in discharge it does not seem to be making any difference at all. This can mean several things, for example, maybe users don't moderate their usage when the phone is placed in power saver. While in practice, the power saver mode is for the smartphone user, understanding what actually happens in this mode can possibly help improve this feature. In fact, Android OS developers can further explore this path and use the gathered data to motivate work on this feature, and even validate such improvements by again looking at the data collected in GreenHub.

4.2.5 Network Details PPM Data

In order to understand the relationship between PPM values and different network related properties, we have built the plots displayed in Fig. 17, which compare the network connection type, the top 5 best/worst mobile network providers in terms of PPM, and the status of the connection when using a mobile network.

Beginning with Fig. 17a we compare the PPM^+ and PPM^- of samples collected when there was an active network connection, and whether through Wi-Fi, mobile (e.g. 3G or 4G), Bluetooth tethering, or unknown.[33]

The plots suggest that a Bluetooth tethering connection is the less impactful on the battery consumption, as the PPM^+ is in general higher and the PPM^- slightly lower in the worst observed case. Hence, it is safe to assume that a device charges faster and drains battery slightly slower with this type of connection. In addition, we see that overall there is no significant difference between having a Wi-Fi connection when opposed to a mobile data connection, either on a charging or discharging scenario ($\boldsymbol{ARQ}_{14}$). This may go against a common conception that keeping a mobile data connection enabled makes the battery drain faster. Once again, as we pointed out in Section 4.2.4, the mobile data connection may also be nowadays used by devices only when needed, hence the impact on the energy drain rate

[33] Can be one of the other 3, but the Android API could not identify which one

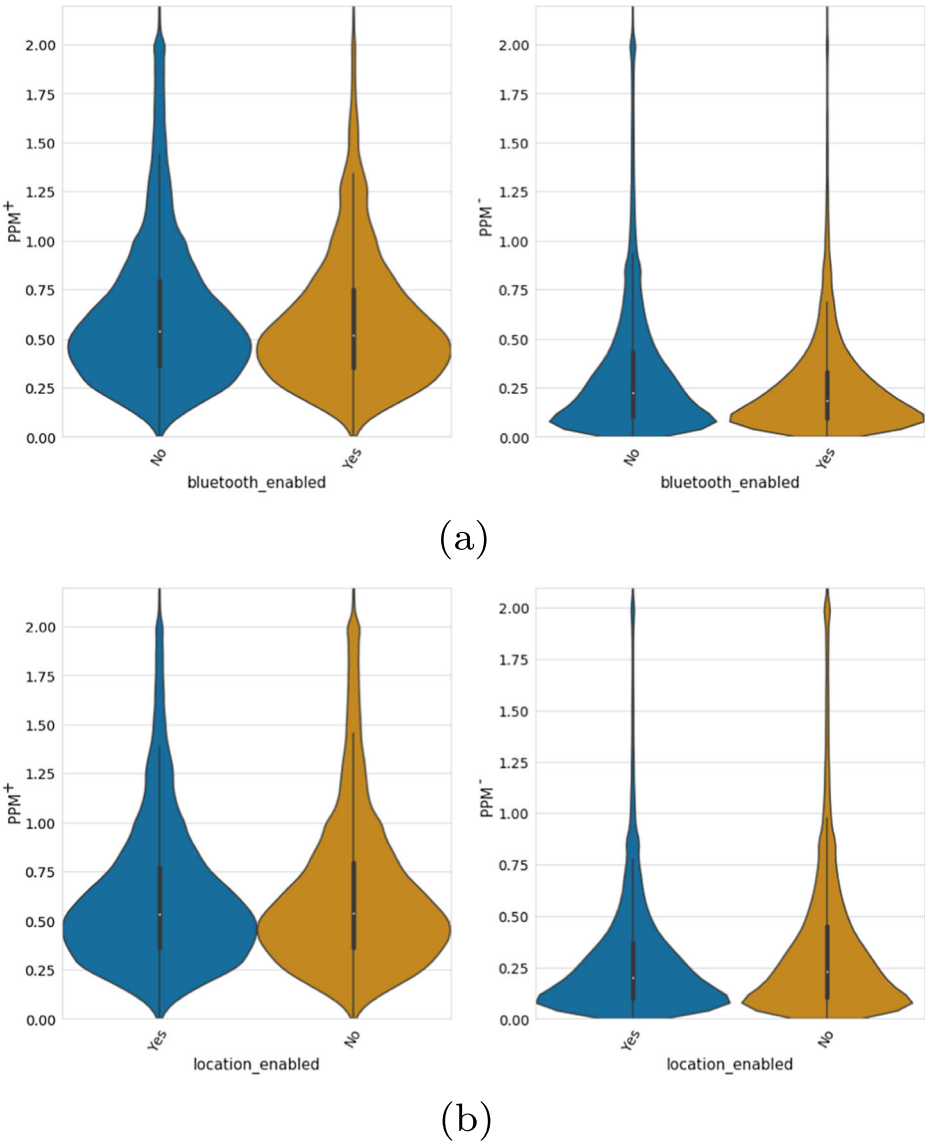
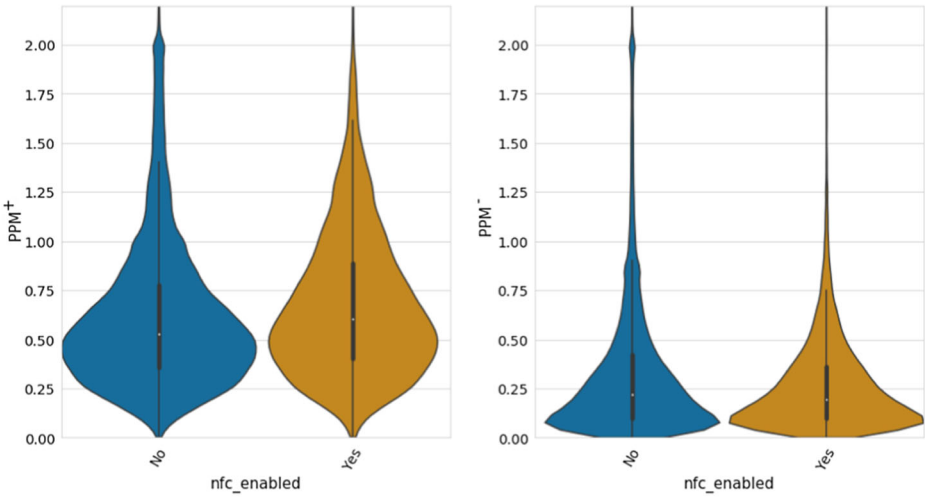
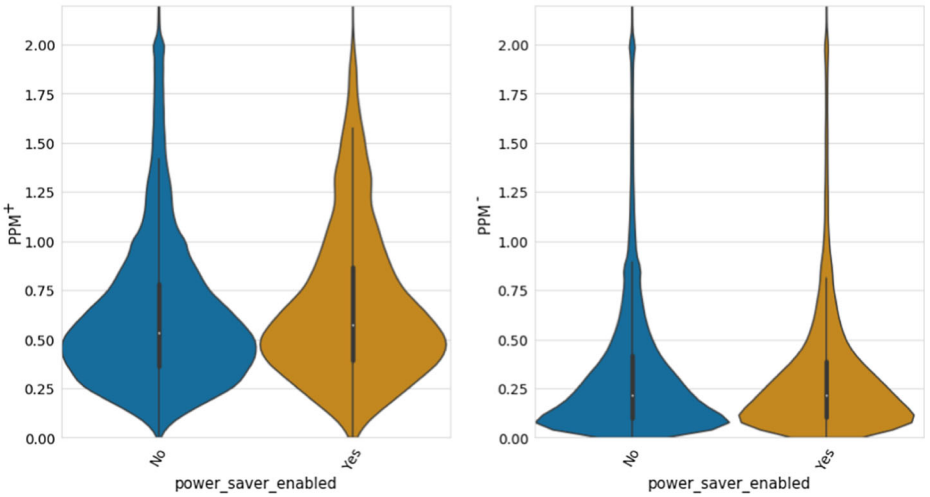


Fig. 15 Settings PPM Data



(c)



(d)

Fig. 15 (continued)

might only be felt when one is actively using the device. On the other end, it could mean that users have a tendency to have a less intensive usage when using a mobile network (to save data), than when using a Wi-Fi connection, thus evening out between the two cases. This could explain the unexpected results, and thus deserves a deeper analysis.

Nevertheless, if we try to compare different mobile operators, then the PPM values appear to be quite different between them. This can be seen in Fig. 17c, where we grouped the samples where a mobile data connection was enabled per mobile operator, and selected

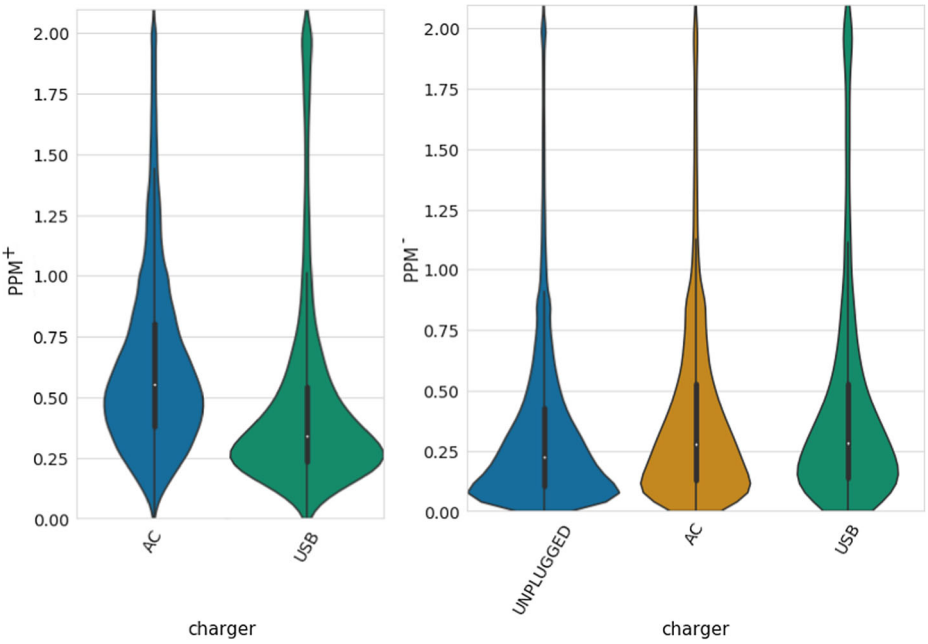


Fig. 16 Battery Details PPM Data

the 5 operators with highest/lowest PPM^{+} (on average). The same was done for the PPM^{-}, and the results are depicted in and Fig. 17d ($\boldsymbol{ARQ}_{15}$).

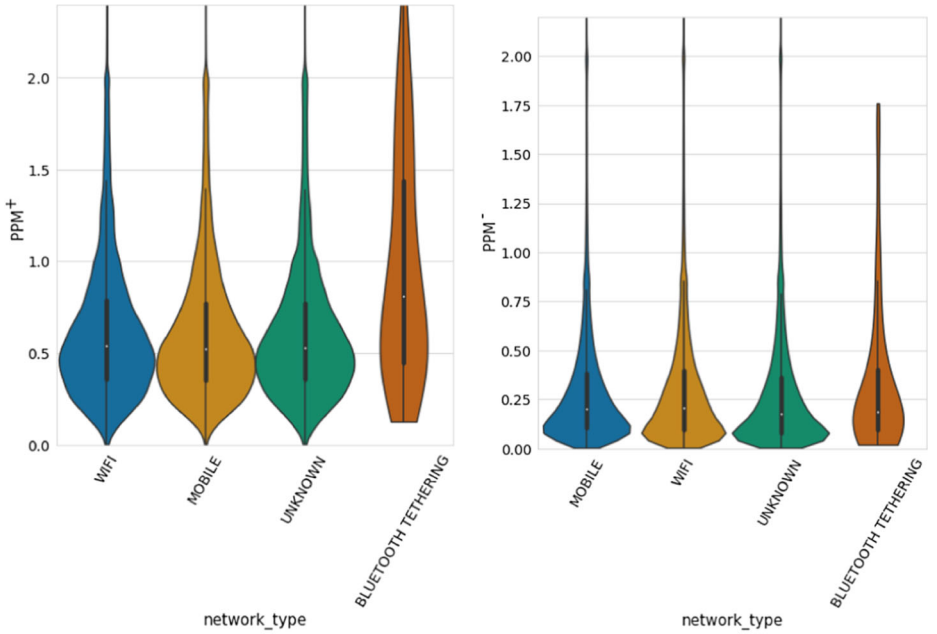
5 Research Opportunities Supported by GreenHub

Looking back at Section 3, we have shown the wide range of data stored within GreenHub Farmer, spreading across many countries, devices, and usage patterns. In Section 4, we have explored the dataset with a clear focus on assessing energy efficiency within Android.

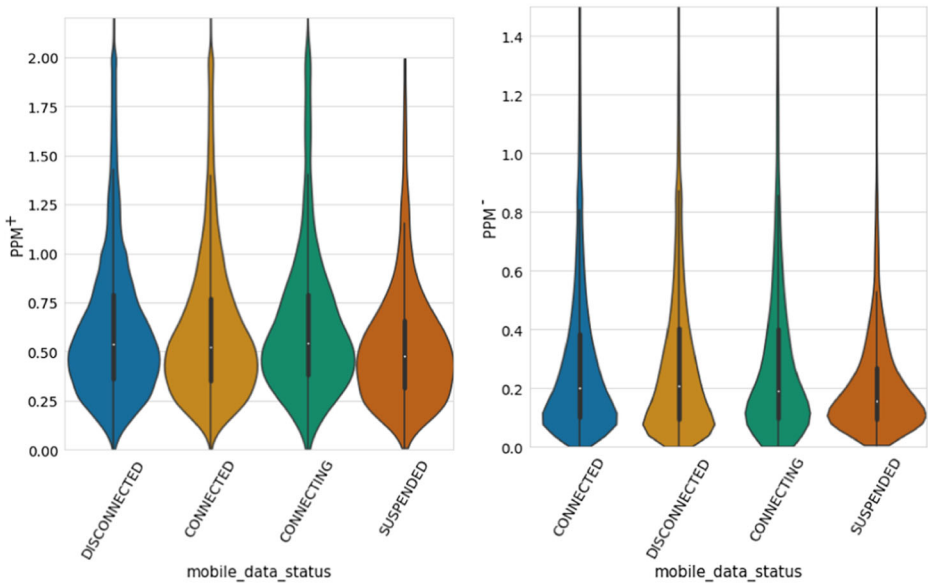
We believe, however, that the dataset itself is rich and diverse enough to be used to support further studies in a wide range of directions and research or even support already existing ones.

In order to illustrate the potential of our dataset, we will present RPs which not only further complement RQs presented in this paper, by focusing on more specific questions and findings, but also open new avenues of research to establish the causality of our highlighted associations and suggesting hypotheses which can be tested in analytical studies. Note that some of these prospective research avenues are not specifically aiming for energy/battery consumption research, but also other more generic areas of research on Android devices. As such, the following two sub-sections will list generic RPs (Section 5.1) and battery consumption related RPs (Section 5.2), respectively, with several examples and explanations of how one might approach the paths in question.

We believe they are as equally promising to be addressed, yet require additional and dedicated quantitative/analytical studies of their own to be pursued, with each characterized by their own threats to validity. In addition, such additional studies can be supported either by

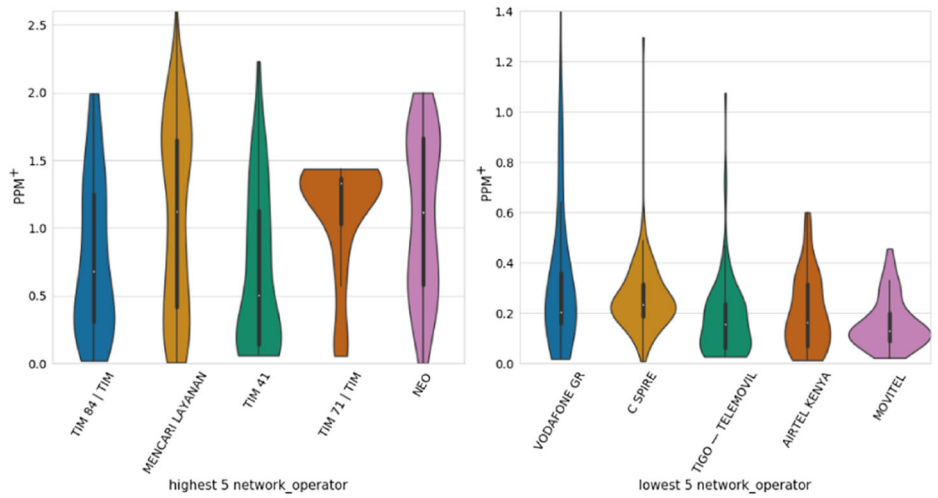


(a)

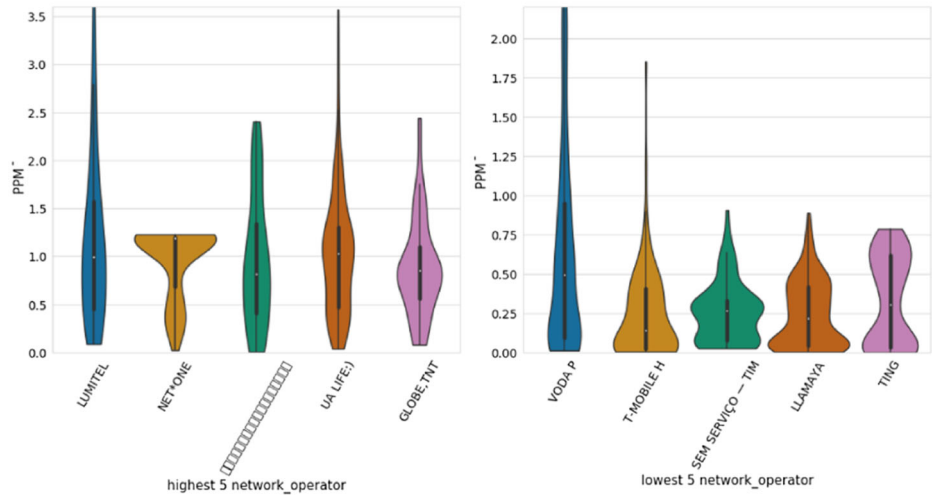


(b)

Fig. 17 Network Details PPM Data



(c)



(d)

Fig. 17 (continued)

fully, or partially (through means of combining other datasets), using our presented dataset or data present within the dataset.

5.1 Generic Research Paths

In this section, we introduce and discuss research paths that may leverage the data in our dataset in order to address generic research question, i.e., research questions that are not necessarily focused on energy and/or battery consumption.

RP_1 Understanding how people from different countries and time zones use their mobile devices daily.

RP_2 Understanding when, and under what circumstances, people from different regions of the globe charge their equipment's batteries.

RP_3 Understanding what type of devices and applications are most/less popular in each country.

RP_4 Understanding how new applications spread worldwide.

These first RPs focus on the *Samples* data. They can be supported by looking at, for example, the data we have stored under `country_code`, `timezone`, and `battery_state` in the *Samples* table, and relating them with their objectives. Researchers can even join data in this table with data in the *Devices* and *App Processes* tables, which we will introduce shortly.

RP_5 Understanding the life expectancy of different device models and relate it with usage profiles and other attributes.

RP_6 Understanding the impact of major technological advances (e.g., 5G introduction, multiple cameras, etc) on the release of new Operating Systems and popularity of devices.

RP_7 Predicting the activity in the mobile devices market, increase/decrease in demand, for instance.

RP_8 Models that are used more by users from certain profiles (gaming, business, influencers, etc.).

RP_9 The frequency users update their smartphone OS versions.

RP_{5-9} look at the data in the *Devices* tables. Considering the large amount of samples and devices in our dataset, if we were to group different device models and organize them across a timeline, we can have a better understanding of their popularity. If a certain device begins to become less represented, we can possibly consider this to be due to users changing their smartphones, thus reducing the device's life expectancy in a real-world setting (which could help answer RP_5). With this in mind, one can correlate dates of when new technologies are introduced, such as 5G, and understand if a certain device looses popularity (this is related to RP_6).

On the other hand, if we were to use outside data sources regarding what type of applications are most used by certain profiles, we can understand what models are most used by such profiles, as proposed in RP_9.

RP_{10} Understanding if there are combinations of apps that change the battery or data consumption behavior of each app individually.

RP_{11} Understanding if hardware and software advances have influenced the popularity of apps or their behavior.

RP_{12} Understand what different usage profiles can be created based on app usage.

These 3 RPs focus on the *App Processes* data table. While we previously proposed, for RP_9, possibly using outside data sources on smartphone user profiles, our dataset can also be used to help support the creation of such profiles. Within the *App Processes* table, we

have a wide range of information on the different applications, and joining this data with the *Sample* and *Devices* data, we can associate different devices with application usage. This can help researchers answer RP_{12} and RP_{11}.

RP_{13} Understanding how the way people use and charge their devices influences the performance of the devices' batteries.

RP_{14} Understanding the impact of wireless charging in the usage patterns of mobile devices and in the health of batteries.

RP_{15} Understanding which recharge cycle better promotes the health of batteries.

If focusing on the *Settings* and *Battery_Details* tables, we can look at RP_{13-15}. For example, let us join *Battery_Details* and *Devices*, and filter those who have their battery's `health` labeled as "bad". With this data, we can try to analyze and understand if users who frequently charge their devices (or leave them in a charging state for long periods) cause their battery's health to degrade over time. Additionally, we can understand what type of charging practices take place when the battery's health is good. Such answers would help answer RP_{15}.

RP_{16} Understanding how people use and control network connections on mobile devices.

RP_{17} Understanding the impact of different connection types on the battery consumption.

RP_{18} Understanding if there is a relationship between apps and connection types.

We believe these 3 paths can be completed with data from `network_type` and `mobile_network_type`, for example, from table *Network_Details*, and even join this data with that present in *App_Processes* as to help relate network and app usage.

5.2 Battery Consumption Related Research Paths

As in the previous section, we present and discuss several research opportunities, supported by GreenHub, which focus on understanding battery consumption tendencies of Android mobile devices.

RP_{19} Is there a relation between a country's average climate and PPM?

RP_{20} How does PPM relate to developing or least developed countries?

RP_{21} Are the less favorable PPM values for the aforementioned countries due to outdated or low spec devices?

To answer RP_{19} and RP_{20} we could, for example, combine the GreenHub *Samples* datable with other outside data sources such as countries' average climates,[34,35] or the United Nation's data on developing nations.[36,37] Additionally, by grouping the *Devices* data with the different `country_code` data, researchers can understand which are the most popular devices in each country. Thus, if older or low spec smartphones are most popular in countries with a low PPM^{+} or high PPM^{-}, this will help answer RP_{21}.

[34] http://berkeleyearth.org/data/
[35] https://datahelpdesk.worldbank.org/knowledgebase/articles/902061-climate-data-api
[36] https://unstats.un.org/unsd/mi/worldmillennium.htm
[37] https://unstats.un.org/unsd/mi/ldc.htm

RP_{22} Are the improvements in PPM across versions correlated with improvement of software energy efficiency in Android, or hardware energy efficiency of devices?

RP_{23} What changed between versions to cause PPM to degrade? Were Android energy smells introduced?

These 2 research paths focus on the *Devices* data table. Currently, our GreenHub PPM analysis details the battery consumption tendencies between major and minor Android versions, which can help support these two prior research paths. Additionally, for these, one would need more external data such as Android release change logs,[38] smartphone specifications,[39] or even Android OS developer insights to better understand the causation of these PPM results.

RP_{24} How do apps' PPM differ when in background and foreground?

RP_{25} How does an app's PPM correlate to the time a user spends using it?

RP_{26} How does an app's PPM differ across older/newer devices?

RP_{27} Why does the lighter version of an app (Facebook), have a tendency to be around when the battery consumption is higher? Is it the app, or is it the device?

RP_{28} How do the PPM values differ across different groups/types of apps, i.e. social media vs. video vs. messaging.

These 5 research paths are research-oriented paths which the GreenHub dataset can support by looking at the *App Processes* data tables, with the objective of understanding what app development or usage practices affect an app's battery consumption.

RP_{29} Which of my app's sub-processes tends to impact battery consumption the most?

RP_{30} How does my app's sub-processes compare to similar app's sub-processes?

RP_{31} Is there any indication of improper process management in my app?

RP_{32} Why is the videoplayer sub-process in *Instagram* impacting the battery more than the core_app itself?

While the previous 5 research paths, based on the *App Processes* data table were more researcher-oriented, these 4 can better help developers. A developer with interest in improving their app could ask themselves the following research paths (RP_{29-31}), and have their answer using our dataset. Additionally, such fine grain data has found a possible improper videoplayer management by the popular *Instagram* social app, which opens up another path (RP_{32}) for researchers or even the app developers to try to understand and improve upon.

RP_{33} Is it possible that having several settings enabled has actually no significant effect on battery drain, as they can be automatically put in a use-when-needed mode by the operating system?

RP_{34} What does the power saving setting actually do to save energy, and why does the effect seem so small when in discharge mode?

[38] https://developer.android.com/preview
[39] https://www.devicespecifications.com/

RP_{33} and RP_{34} focus on the *Settings* and *Battery_Details* data tables. To answer these, we would need knowledge from Android OS developers in combination with the presented PPM data within GreenHub to properly explore such research opportunities aimed for practitioners.

RP_{35} How significantly can the choice of a network operator influence the battery drain, or charging speed, in a device? And what are the reasons of that happening?

RP_{36} Do different network operators in fact influence the battery drain or is this a case of "correlation does not imply causation", with other factors influencing this?

RP_{37} How different are the usage scenarios when connected to Wi-Fi or a mobile network? Do users tend to be more conservative in one or the other?

Finally, to answer RP_{35} and RP_{36} (focused on *Network Details*, a more thorough understanding of the domain (network hardware and network operators) would be needed to understand the causes of the presented results. RP_{37}, on the other hand, can be answered by joining the *App Processes* and *Samples* data and thoroughly analyzing the different usage scenarios present when in Wi-Fi or mobile network use.

6 The Battery Double Conundrum

In the characterization that we made in Section 3.3, when looking through and analyzing the *App Processes* table, we were made aware of a peculiar occurrence. We were already anticipating our GreenHub BatteryHub app process to be present in each of our samples, as it is the process responsible for data collection, but noticed there was another app process called Battery Double[40] heavily present in our samples, when BatteryHub was not. In fact, roughly 25% of our samples were found to originate from the GreenHub app, while roughly 75% originated from Battery Double.

Upon inspection of the Battery Double, we quickly realized it was a clone of our BatteryHub app, with a layer of advertisements, and very slight changes such as showing two batteries' percentage instead of a percentage wheel. Additionally, when viewing our Fabric[41] dashboard for BatteryHub, we also verified that it was tracking the Battery Double reports. This means that the Battery Double contains the same developer id, which further confirms that Battery Double is a clone.

This conundrum, of which we were not previously aware, made us question if the sample data coming from Battery Double is un-tampered (a clone made only for ad revenue), or if the developer altered something on the data collector (which would reduce our confidence in the data). Thus, we decided to verify if data coming from both sources could be considered valid, independently of the app where it originated from, or should be discarded whenever it originated from the Battery Double app. To do so, we opted to *(a)* analyze the source-code of both applications and *(b)* statistically conclude if the data from both cases can be considered as a single entity.

First, for *(a)* we sought to further understand if the Battery Double doppelganger application introduced any modification that could somehow compromise the data model and/or

[40] Battery Double: https://play.google.com/store/apps/details?id=com.mansoon.BatteryDouble&hl=en
[41] Fabric is a usage and crashlytics reporter: https://get.fabric.io/

the data collection mechanisms, or if it was, as we first presumed, just a full clone of BatteryHub with an advertisement layer. In order to do so, we first retrieved the Battery Double APK files from all (currently) existing 12 versions from various APP gathering App Markets.[42,43,44] Looking at the current, and all previous versions allows us to verify if at any point in time the data collection mechanism has been compromised. Afterwards, we used reverse engineering tools[45,46] to obtain the source code from those files, and with it we were able to establish a comparison between all 12 versions of the Battery Double code (as of June 2020 with version 1.1.6) and the one from BatteryHub.

The source code comparison task could not be performed automatically, through using an AST or code difference tool. This is due to the fact that the code retrieved from the APK file was already subject to standard optimizations performed at compilation time. Ultimately, this means that variables had their names changed, and expressions with syntactic sugar were transformed. Therefore, we manually verified if 3 properties were the same in both apps: *(i)* if the data collection mechanisms were performing the exact same tasks and *(ii)* were being triggered exactly the same way, and *(iii)* if the data model was not tampered or altered in any form.

Upon performing this manual inspection, we concluded with certainty that our initial presumptions were confirmed. In other words, from when BatteryHub was cloned and integrated into Battery Double (from Battery Double version 1.0.6 on wards), the whole BatteryHub data collection mechanism was strictly kept intact.

Second, for *(b)*, we split the dataset into two parts. The part comprising only observations stemming from the original BatteryHub app is called **original** dataset. Conversely, the part of the dataset comprising samples stemming from Battery Double is called **doppelganger** dataset. This was easily done by analyzing the Farmer data, in which we cross-examined each sample's processes to determine if either BatteryHub or Battery Double was present.

In order to verify whether we can combine the original and doppelganger datasets, we adopt the following procedure. First, we draw a random, 1000-observation sample[47] from each dataset. Hereafter we refer to these samples as **generic samples**. Second, we identify the most common smartphone model (between both datasets) considering observations from both original and doppelganger datasets, the Samsung G920F, and then draw a random, 1000-observation sample from each dataset comprising only observations that were captured by this specific model. Hereafter, we refer to these samples as **model-specific samples**. In the first step, we draw samples to directly compare the two datasets. However, the observations in the doppelganger dataset have a strong bias towards countries that are not amongst the most popular ones in the original dataset. To account for potential regional differences, e.g., in terms of popularity of specific smartphone manufacturers and models, the second step accounts for potential regional differences by focusing on a single smartphone model. If the Samsung 920F users in the two model-specific samples tend to in fact send different (or altered) data, it is likely that we cannot treat the dataset as a single entity.

[42] DownloadAPK: https://downloadapk.net/Battery-Double.html

[43] APKPure: https://apkpure.com/br/battery-analytics/com.mansoon.BatteryDouble/versions?fbclid=IwAR2gmTl73T17fRsSg1vpnJdUqapSFoAbFUGQ7eV6IfcriOnljx_rqAoLpQA

[44] AppBrain: https://www.appbrain.com/app/battery-analytics/com.mansoon.BatteryDouble

[45] Jadx: https://github.com/skylot/jadx

[46] APKtool: https://ibotpeaches.github.io/Apktool/

[47] Number of samples required to obtain a confidence level of 95% and a confidence interval of ±3, considering each dataset as the population.

The third step consists of obtaining the rows corresponding to both the generic and model-specific samples for all the tables in our data model, filtering out non-numerical, timestamp, and key-related columns. In the fourth and final step, we use two different statistical, multivariate, nonparametric tests to quantify the difference between samples drawn from the original and doppelganger datasets, considering both generic and model-specific samples. The employed statistical tests are i) analysis of similarities (ANOSIM) (Clarke 1993), and ii) permutational multivariate analysis of variance using distance matrices (PERMANOVA) (Anderson 2001). The intuition for ANOSIM is explained by the documentation of the `vegan`[48] R package: *"If two groups of sampling units are really different in their species composition, then compositional dissimilarities between the groups ought to be greater than those within the groups."*. As for PERMANOVA, it *"is used to compare groups of objects and test the null hypothesis that the centroids and dispersion of the groups as defined by measure space are equivalent for all groups."* (Anderson 2001). Both tests produce a score ranging from 0 to 1 the closer this value is to 1, the more the sites within a group are similar to each other and dissimilar to sites in other groups. In other words, 1 indicates no similarity between the two groups, where 0 indicates a perfect similarity between the two groups.

We employ the vegan R package, which implements functions to apply both ANOSIM and PERMANOVA. For the generic samples, ANOSIM produced an R score of 0.1867, p-value = 0.001. This score indicates that the dissimilarity between the generic samples from the original and doppelganger datasets is low (the R score is a measure of effect size (Urdan 2016)) and the probability of this result being coincidental is very low ($< 0.1\%$). In a similar vein, PERMANOVA (called `adonis` in R) produces an R^2 score of 0.1602, p-value = 0.001. This means that only 16% of the difference in their variance is explained by the dataset from which the sample was drawn. Thus, it is possible to say that the generic samples should not be considered different, since so little of their difference stems from having two groups. When considering the model-specific samples, for ANOSIM we have $R = 0.001455$ and a p-value of 0.035 and for PERMANOVA $R^2 = 0.0022$ and p-value = 0.023. For the model-specific samples, the produced scores for both tests suggest that the two datasets should also not be considered different. Based on the results of this analysis, we combine the two datasets into a single dataset and make no further distinctions. In order to maintain the integral future assurance of the soundness of our data, we have decided to block any new samples, after V1.1.6, coming from Battery Double.

7 GreenHub Pipeline

This section will briefly describe the GreenHub Pipeline which was used to support the data cleaning, data processing, and the results of the previous sections. For easier manipulation of the data, we created a CSV to Parquet converter[49] for our GreenHub data tables, significantly reducing the size of our datasets. For the analysis, we used the Python Pandas library[50] and Jupyter Notebooks,[51] an open-source web-based interactive computing environment for creating notebook documents. These notebooks are human-readable documents

[48] Vegan: https://cran.r-project.org/web/packages/vegan/index.html
[49] Dataset-Converter Tool: https://github.com/greenhub-project/dataset-converter
[50] Pandas: https://pandas.pydata.org/
[51] Project Jupyter: https://jupyter.org/

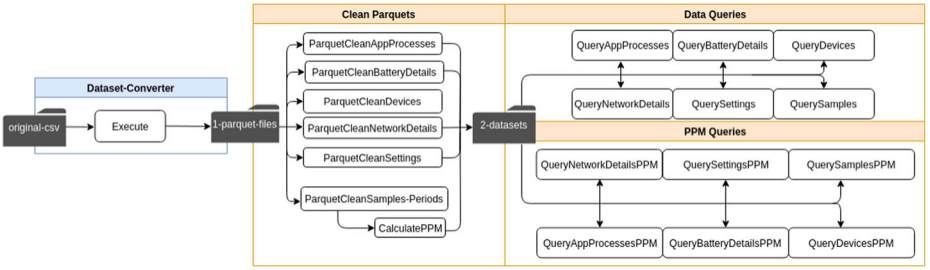


Fig. 18 The GreenHub Analysis Pipeline

containing both computer code and text elements for analysis descriptions and results (such as figures, tables, etc) and are executable to perform data analyses.

This structure allows easy to update modular processes for data cleaning, transformation, statistical modeling, and data visualization and can be automatized to keep all the results within GreenHub up to date and public. Additionally, this base pipeline can be used to update and increment new notebooks by members within the research community. Our GreenHub Jupyter Pipeline can be found at:

https://github.com/greenhub-project/notebooks

Shown in Fig. 18 is the GreenHub Pipeline. It is divided into 4 different parts: *Dataset-Converter*, *Clean Parquets*, *Data Queries*, and *PPM Queries*.

The dataset converter pipeline tool converts the large CSV files, from GreenHub Farmer, into efficient Apache Parquet[52] binary files. It uses configuration files to describe how the conversion for a given CSV file should be performed, which columns to include, and which plugins to use. The tool returns, for each configuration file, a parquet file and a pickle file containing the pandas data types for later reuse.

Afterwards, *Clean Parquets* is for data cleaning and transformation of each parquet file, including normalization of values, string treatments, etc. Additionally, it contains the notebooks for both identifying the *period slices* and calculating the PPM metrics explained in Section 4.1.

Finally, *Data Queries* and *PPM Queries* contains the queries and results shown in Section 3 and Section 4.2 respectively, including further data and results not presented directly in this paper.

It is worth noticing that the CSV file for the App Processes are a smaller set containing the top 15 most represented apps (excluding system apps, pre-loaded non-play store apps, and GreenHub data collecting apps) as the App Processes data is far too large to be efficiently processed. In this case, the results were provided by directly querying and manually filtering the GreenHub Farmer data, prior to generating the App Processes CSV files.

8 Threats to Validity

We discuss two of the main possible threats to the validity of our study: GreenHub's data integrity and the findings from the data analysis.

[52] Apache Parquet: https://parquet.apache.org/

In regards to the data integrity, GreenHub's open-source data collection infrastructure was inspired by that of the open-source Carat (Oliner et al. 2013) project.

The data collecting system and data models was from an initial fork of the Carat code, which itself was based off accessing the public Android API to gather the data regarding all the various aspects of the smart phone, and also uses the phone specification models which are provided by manufacturers. From this fork, the data model was updated in GreenHub to also consider further modern device characteristics and data aspects (e.g. NFC, Flashlight, etc.), and in such cases, GreenHub thoroughly followed the same spirit, methods, and protocols of data collection as in the original Carat system.

All the collected data is completely through the public Android API, with no manipulation within the GreenHub infrastructure, which is then sent to the GreenHub Farmer database. By design, no data is discarded whatsoever (from what could be collected), as removing data could limit or prevent it from being used in a particular scenario by other researchers, which could not have been initially envisioned.

In some cases, such data are measurements (such as battery level, temperature, brightness, etc.), other cases they are a sensor *boolean* value (Bluetooth on or off), and finally in other cases they are values from a given subset of possibilities (country codes). The automatic constraint check of the data types when inserting a new sample within the database guarantees another level of data assurance and quality. As such, the data present within Farmer is purely that of a direct reporting of a device's sample containing all the attributes shown in Table 1. Thus, there is no corrupt data collected and the Android API guarantees that each such attribute contains an associated value (there are no empty or *null* values).

Nevertheless, as with any data science research or data analysis, the data must undergo a data cleaning step, heavily based on the goals and objectives the data scientist has during the analysis process. The second part of our open-source GreenHub pipeline (as shown in Fig. 18) focuses on the such cleaning of the data, in line with our research goals. In this step, we carefully analyzed the distribution of the data values, detected any inconsistent, incomplete, or non uniform data, treated any detected syntax errors, added complementary information, etc. Such issues occur due to the specification models, which manufacturers provide, not following a normalized approach. In addition, during the data cleaning process, we also discarded all statistically calculated outlier PPM values and their periods.

All such dirty data cleaning is openly documented within our GreenHub pipeline for any researcher to further understand all the actions performed, and to contribute to the initiative and data cleaning with their expertise and the addition of any complementing data which they would wish to add.

In addition to the data collected through the BatteryHub app, there is data from a clone app called Battery Double. Roughly 75% of the sample data we have collected came from users of Battery Double, which quickly made us question the quality and validity of the data. As detailed in Section 6, we ran statistical tests to understand if this doppelganger app was in fact only a clone made for ad revenue, as it superficially seemed, and there was no tampering with the data collecting process, or if the developer altered something (which would reduce our confidence in the data). Through both statistical tests and manual inspection of source code, we verified that we could in fact combine the two datasets, and make no further distinctions. In addition to the analysis of the data samples from each app, we analyzed the source-code and further guarantee with full confidence that the data collecting processes within Battery Double suffered absolutely no modifications, and the data itself was being correctly gathered up until version V1.1.6 as of July 2020, guaranteeing the past and present assurance of the soundness of the data.

In regards to our presented findings, it is important to understand the interpretation of our presented *percentage-per-minute* metric, or PPM. The actual measuring of the energy (Joules) or power (watts) consumed by the devices is not only difficult, but at the scale of the GreenHub project considering such varied real-world usage, would be impossible. Thus, we defined the PPM metric (detailed in Section 4.1) as a proxy to the battery discharge/charge rate within a given continuous period of time, from one specific user. This metric equates to the *tendency* which various factors, such as the different models, brands, networks, settings, applications, and even countries have on the battery. The principal idea behind PPM is that it is not to be seen as a direct and concrete value of efficiency, but how a single (or group of) characteristic (i.e., device brand, certain settings turned on/off, different apps) *tends* to affect the battery, considering all the wide ranges of (secondary) factors which exist.

For example, let us consider that we were analyzing app processes, and one of the specific social media apps in question has a consistently higher reported PPM value than the others. This would indicate that this app has, across a broad range of samples across varied usage patterns, devices, brands, setting configurations, operation systems, background/foreground apps, etc., (essentially across all the possible characteristics which differ across devices), a tendency to be present when there is a higher charge/discharge pattern in a smartphone. This does not directly mean that the app in question is without a doubt inefficient, but that there is a high possibility of it contributing to such patterns. In other words, the PPM metric represents the tendency one (or a group of) characteristics have on the battery, factoring in all the other characteristics from real-world usage, and points to possible efficiency problems or advantages.

Thus, our findings and results are of a reporting nature of this tendency metric, detailing what our data seems to state and are the first iteration of understanding what factors influence a smartphone's battery. As such, these findings open paths to deeper and more detailed studies which may further answer underlying questions to understand why such results occurred.

9 Related Work

A number of previous initiatives aiming at collecting information about energy usage in a crowdsourced manner have been devised. For example, Carat (Oliner et al. 2013) collects information about smartphone usage to identify energy bugs, situations where an app running on a device consumes much more energy than it usually does under the same circumstances in other devices, and energy hogs, situations where the app consumes much more energy than other apps on the same device. The work of Chon and colleagues (Chon et al. 2016) leverages crowdsourced information to estimate the battery capacity of a smartphone when compared to a new one and provides hints for users to assist them in configuring their phones to save energy. Guo et al. (2017) have performed intra device analysis independently of manufacturer and operating system versions to identify battery discharging patterns, types of mobile apps, and their energy consumption patterns. Compared to these previous initiatives, GreenHub has a more collaborative approach, since it was designed to be used by third parties and all the collected data is made publicly available. Furthermore, it gathers information about modern sensor utilization (e.g., NFC) and concerning the battery itself (e.g., electric current), which are neglected by previous work. The combination of these distinctive factors may potentiate further breadth studies, possibly performed by members of the community other than us, as well as depth studies, since more data is available.

Energy profiling is an active area of research that has received significant attention lately and is also related to GreenHub. A number of energy profilers have arisen in the last few years, including ones within the major mobile computing platforms. Android provides since 2014 tools called BatteryStats and Battery Historian (LLC 2014) to collect information about energy usage from Android devices and support the visualization and analysis of the evolution of these measurements, respectively. Previous work has shown that energy measurements produced by these tools are comparable to those produced by a mid-range physical meter (Di Nucci et al. 2017). Furthermore, since 2018 Android Studio includes a full-fledged battery profiler (LLC 2018). Apple's Xcode also includes an energy profiler as part of its Instruments tool (Inc. 2018). Qualcomm also had a profiler for Android devices running on its Snapdragon family of processors named Trepn (Incorporated 2014). Besides being capable of measuring energy consumption with very high precision, Trepn was one of the few energy profilers we are aware of that is capable of reporting on GPU energy usage. It is, however, no longer supported by Qualcomm.

In academia there are also multiple proposals for energy profilers. For example, PETRA (Nucci et al. 2017) is a software-based profiler that leverages Batterystats to estimate method-level energy consumption. Eprof (Pathak et al. 2012) goes in a different direction. It instruments applications and the OS so as to trace system calls and identify components that consume more energy. The work of Hoque and colleagues (Hoque et al. 2015) analyzes in detail a variety of profilers from both industry and academia.

Such energy profilers are typically used to address the issue of not only monitoring the energy consumption of software, but also to properly relate it with specific components of an application. Nevertheless, the information that one could achieve with GreenHub diverges from the one offered by profilers, although both share the same ultimate goal. GreenHub leverages *Batterystats* to collect data about battery usage. However, it combines information about tens of thousands of devices and the apps running on them to support app and platform developers, researchers, and device manufacturers to study how these devices and apps behave in the wild, under real-world usage conditions and on a large scale. In this sense, it complements existing energy profilers.

10 Conclusions

This paper presents the results of the data we gathered that stem from the GreenHub initiative. The GreenHub dataset contains over 23 million samples of real-world usage crowd-sourced over 1.6k+ different brands of Android devices running over 50 Android versions and across 160 countries. The data is both representative, and publicly available, allowing a collaborative approach in the usage of the gathered data for analyzing and identifying opportunities for optimization of battery consumption in Android devices. But, we believe that this dataset also contains a rich set of data that will be valuable for researchers in other areas, other than Android battery consumption. Thus, to further promote the analysis and identification of opportunities for research, we presented a detailed characterization and a high-level overview of the GreenHub dataset, and present various potentially new avenues of research, which we deemed research paths.

Focusing on our own interests, concretely on analyzing battery consumption in smartphones, and to further help us understand how our data on battery discharging and charging relates to usages in the real-world, we defined a metric called *Percentage Per Minute* or PPM. This helped support our analysis on the tendencies of battery consumption in regards

to different aspects and settings of smartphone usage by factoring in real-world human interaction across varied user usage profiles.

During this analysis we were able to understand things such as what are the charge/discharge tendencies across different countries; are there observable battery tendencies across brands and models; is battery usage improving between Android versions; how do popular applications compare in terms of battery consumption tendencies; among others. We also gathered results that defied common intuition, and even identified potential anomalies within a popular Android application. During our analysis and while providing answers to our own research questions, many new research questions and paths (focusing on Android smartphone battery consumption) emerged and deserve their own dedicated studies.

To easy the replication and automation of our work for other researchers, and allow others to further contribute to the GreenHub initiative, we have also presented the GreenHub Pipeline. This processing and analysis pipeline supports the results presented in this paper, and allowed for easier manipulation and visualization of the data. Based on the Pandas library and Jupyter Notebooks, we present human-readable documents containing both computer code and text elements for analysis, descriptions, and results. As this pipeline is modular, it allows any future analyses on the data to be reproduced. Furthermore, it also allows new modules to be introduced and updated by other members of the community to further expand the analysis on the data and perform new queries focusing on new questions.

We aim to preserve a sustained movement towards extracting useful information from the collected data. For this, we invite other researchers and developers to both analyze and contribute to our dataset, analysis pipeline, and the GreenHub initiative.

Continuing with what we have presented in this paper, we intend to further expand the dataset schema with different domains of information, such as data on demographics to better understand the mobile device user profile and respective usage pattern. Upgrading the infrastructure is also an important consideration in order to sustain the scalability of the dataset size, which will continue to be open-source. This process will result in an incremental improvement of the data provided regarding the quality and the availability factors.

For the dataset analysis and future research directions, we aim to study which features (or combination of features) of mobile devices may impact the energy usage behavior the most, how the different environments affect energy consumption and how accurately can we predict energy related patterns.

Acknowledgements This research and work is funded: by national funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project CISUC - UID/CEC/00326/2020, project UIDB/50014/2020, and by European Social Fund, through the Regional Operational Program Centro 2020; by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, co-financed by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT; by CNPq/Brazil (304755/2014-1, 406308/2016-0), FACEPE/Brazil (APQ-0839-1.03/14), and INES 2.0, FACEPE grants PRONEX APQ 0388-1.03/14 and APQ-0399-1.03/17, and CNPq grant 465614/2014-0; by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT - Foundation for Science and Technology; The first author was financed by post-doc grant reference C4_SMDS_L1-1_D and the third author financed by FCT grant SFRH/BD/132485/2017; Additionally, this paper acknowledges the support of the Erasmus+ Key Action 2 (Strategic partnership for higher education) project No. 2020-1-PT01-KA203-078646: "SusTrainable - Promoting Sustainability as a Fundamental Driver in Software Development Training and Education". The information and views set out in this paper are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

References

Anderson MJ (2001) A new method for non-parametric multivariate analysis of variance. Austral Ecology 26(1):32–46

Chon Y, Lee G, Ha R, Cha H (2016) Crowdsensing-based smartphone use guide for battery life extension. In: Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing. ACM, pp 958–969

Clarke KR (1993) Non-parametric multivariate analysis of changes in community structure. Aust J Ecol 18:117–143

Couto M, Pereira R, Ribeiro F, Rua R, Saraiva J (2017) Towards a green ranking for programming languages. In: Proceedings of the 21st Brazilian symposium on programming languages, SBLP 2017. ACM, pp 7:1–7:8. Best Paper

Cruz L, Abreu R (2017) Performance-based guidelines for energy efficient mobile applications. In: Proceedings of the 4th international conference on mobile software engineering and systems, MOBILESoft '17. IEEE Press, pp 46–57

Di Nucci D, Palomba F, Prota A, Panichella A, Zaidman A, De Lucia A (2017) Software-based energy profiling of android apps: Simple, efficient and reliable? In: 2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER), pp 103–114

Fu B, Lin J, Li L, Faloutsos C, Hong J, Sadeh N (2013) Why people hate your app: making sense of user feedback in a mobile app store. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1276–1284

Guo Y, Wang C, Chen X (2017) Understanding application-battery interactions on smartphones: a large-scale empirical study. IEEE Access 5:13387–13400

Harris (2018) Our phones and gadgets are now endangering the planet. https://www.theguardian.com/commentisfree/2018/jul/17/internet-climate-carbon-footprint-data-centres. Accessed 24 Jan 2018

Hasan S, King Z, Hafiz M, Sayagh M, Adams B, Hindle A (2016) Energy profiles of java collections classes. In: Proceedings of the 38th international conference on software engineering. ACM, pp 225–236

Hoque MA, Siekkinen M, Khan KN, Xiao Y, Tarkoma S (2015) Modeling, profiling, and debugging the energy consumption of mobile devices. ACM Comput Surv 48(3):39:1–39:40. https://doi.org/10.1145/2840723

Hu Y, Yan J, Yan D, Lu Q, Yan J (2017) Lightweight energy consumption analysis and prediction for android applications. Science of Computer Programming

Inc. A (2018) Instruments overview. https://help.apple.com/instruments/mac/10.0/#/dev7b09c84f5. Accessed 28 Sep 2019

Incorporated Q (2014) Trepn profiler. https://developer.qualcomm.com/forums/software/trepn-power-profiler. Accessed 28 Sep 2019

Jabbarvand R, Sadeghi A, Garcia J, Malek S, Ammann P (2015) Ecodroid: an approach for energy-based ranking of android apps. In: Proceedings of 4th international workshop on green and sustainable software, GREENS '15. IEEE Press, pp 8–14

Khalid H, Shihab E, Nagappan M, Hassan AE (2015) What do mobile app users complain about? IEEE Softw 32(3):70–77

Li D, Halfond WGJ (2014) An investigation into energy-saving programming practices for android smartphone app development. In: Proceedings of the 3rd international workshop on green and sustainable software, GREENS 2014. ACM, New York, pp 46–53. https://doi.org/10.1145/2593743.2593750

Li D, Hao S, Halfond WG, Govindan R (2013) Calculating source line level energy information for android applications. In: Proceedings of the 2013 international symposium on software testing and analysis. ACM, pp 78–89

Li D, Lyu Y, Gui J, Halfond WGJ (2016) Automated energy optimization of http requests for mobile applications. In: Proceedings of the 38th international conference on software engineering, ICSE '16. ACM, pp 249–260

Lima LG, Melfe G, Soares-Neto F, Lieuthier P, Fernandes JP, Castor F (2016) Haskell in green land: analyzing the energy behavior of a purely functional language. In: Proceedings of the 23rd IEEE international conference on software analysis, evolution, and reengineering (SANER'2016). IEEE, pp 517–528

Lin K, Kansal A, Lymberopoulos D, Zhao F (2010) Energy-accuracy trade-off for continuous mobile device location. In: Proceedings of the 8th international conference on mobile systems, applications, and services. ACM, pp 285–298

Linares-Vásquez M, Bavota G, Bernal-Cárdenas C, Oliveto R, Di Penta M, Poshyvanyk D (2014) Mining energy-greedy api usage patterns in android apps: an empirical study. In: Proceedings of the 11th working conference on mining software repositories. ACM, pp 2–11

Linares-Vásquez M, Bavota G, Cárdenas CEB, Oliveto R, Di Penta M, Poshyvanyk D (2015) Optimizing energy consumption of guis in android apps: a multi-objective approach. In: Proceedings of the 2015 10th joint meeting on foundations of software engineering. ACM, pp 143–154

LLC G (2014) Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/studio/profile/battery-historian. Accessed 28 Sep 2019

LLC G (2018) Inspect energy use with energy profiler. https://developer.android.com/studio/profile/energy-profiler. Accessed 28 Sep 2019

Manotas I, Bird C, Zhang R, Shepherd D, Jaspan C, Sadowski C, Pollock L, Clause J (2016) An empirical study of practitioners' perspectives on green software engineering. In: International conference on software engineering (ICSE), 2016 IEEE/ACM 38th, IEEE, pp 237–248

Matalonga H, Cabral B, Castor F, Couto M, Pereira R, de Sousa SM, Fernandes JP (2019) Greenhub farmer: real-world data for android energy mining. In: 2019 IEEE/ACM 16th international conference on mining software repositories (MSR). IEEE, pp 171–175

Mickle T (2018) Your phone is almost out of battery. remain calm. call a doctor. https://www.wsj.com/articles/your-phone-is-almost-out-of-battery-remain-calm-call-a-doctor-1525449283. Last visit: 2019-02-05

Nucci DD, Palomba F, Prota A, Panichella A, Zaidman A, Lucia AD (2017) Petra: a software-based tool for estimating the energy profile of android applications. In: 2017 IEEE/ACM 39th international conference on software engineering companion (ICSE-c), pp 3–6

Oliner AJ, Iyer AP, Stoica I, Lagerspetz E, Tarkoma S (2013) Carat: collaborative energy diagnosis for mobile devices. In: Proceedings of the 11th ACM conference on embedded networked sensor systems, SenSys '13, Roma, Italy, November 11-15, 2013. ACM, pp 10:1–10:14

Oliveira W, Oliveira R, Castor F (2017) A study on the energy consumption of android app development approaches. In: Proceedings of the 14th international conference on mining software repositories. IEEE Press, pp 42–52

Oliveira WJr, Oliveira R, Castor F, Fernandes B, Pinto G (2019) Recommending energy-efficient java collections. In: Proceedings of the 16th international conference on mining software repositories, MSR 2019, pp 160–170. Montreal, Canada

Pang C, Hindle A, Adams B, Hassan AE (2016) What do programmers know about software energy consumption? IEEE Softw 33(3):83–89

Pathak A, Hu YC, Zhang M (2012) Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In: Proceedings of the 7th ACM European conference on computer systems. ACM, pp 29–42

Pereira R, Couto M, Ribeiro F, Rua R, Cunha J, Fernandes JP, Saraiva J (2017) Energy efficiency across programming languages: how do energy, time, and memory relate? In: Proceedings of the 10th ACM SIGPLAN international conference on software language engineering, SLE 2017. ACM, pp 256–267

Pereira R, Couto M, Saraiva J, Cunha J, Fernandes JP (2016) The influence of the java collection framework on overall energy consumption. In: Proceedings of the 5th international workshop on green and sustainable software, GREENS '16. ACM, pp 15–21

Pereira R, Simão P, Cunha J, Saraiva J (2018) jStanley: placing a green thumb on java collections. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, ASE 2018. ACM, pp 856–859

Pinto G, Castor F (2017) Energy efficiency: a new concern for application software developers. Commun ACM 60(12):68–75

Pinto G, Castor F, Liu YD (2014) Mining questions about software energy consumption. In: Proceedings of the 11th working conference on mining software repositories. ACM, pp 22–31

Pinto G, Liu K, Castor F, Liu YD (2016) A comprehensive study on the energy efficiency of java's thread-safe collections. In: 2016 IEEE international conference on software maintenance and evolution, ICSME 2016, Raleigh, NC, USA, October 2-7, 2016, pp 20–31

Richter (2018). The most wanted smartphone features. https://www.statista.com/chart/5995/the-most-wanted-smartphone-features. Accessed 24 Jan 2018

Shiffler RE (1988) Maximum z scores and outliers. The American Statistician 42(1):79–80

Thorwart A, O'Neill D (2017) Camera and battery features continue to drive consumer satisfaction of smartphones in US. https://www.prnewswire.com/news-releases/camera-and-battery-features-continue-to-drive-consumer-satisfaction-of-smartphones-in-us-300466220.html. Last visit: 2019-02-06

Tung L (2015) Android fragmentation: there are now 24,000 devices from 1,300 brands. https://www.zdnet.com/article/android-fragmentation-there-are-now-24000-devices-from-1300-brands/. Accessed 19 Sep 2019

Urdan TC (2016) Statistics in plain english, 4th edn. Routledge

Wan M, Jin Y, Li D, Gui J, Mahajan S, Halfond WG (2017) Detecting display energy hotspots in android apps. Software Testing, Verification and Reliability 27(6):e1635

Affiliations

Rui Pereira[1,2] · Hugo Matalonga[3] · Marco Couto[2,3] · Fernando Castor[4] · Bruno Cabral[5] · Pedro Carvalho[5] · Simão Melo de Sousa[6] · João Paulo Fernandes[5]

Hugo Matalonga
hugo@hmatalonga.com

Marco Couto
marco.l.couto@inesctec.pt

Fernando Castor
castor@cin.ufpe.br

Bruno Cabral
bcabral@dei.uc.pt

Pedro Carvalho
pfcarvalho@student.dei.uc.pt

Simão Melo de Sousa
desousa@di.ubi.pt

João Paulo Fernandes
jpf@dei.uc.pt

1 C4 - Centro de Competências em Cloud Computing (C4-UBI), Universidade da Beira Interior, Rua Marquês d'Ávila e Bolama, Covilhã 6201-001, Portugal

2 HASLab/INESC Tec, Porto, Portugal

3 Universidade do Minho, Braga, Portugal

4 Universidade Federal de Pernambuco, Recife, Brasil

5 CISUC, DEI, Universidade de Coimbra, Coimbra, Portugal

6 Universidade da Beira Interior, C4, Lab. Release, NOVA-LINCS, Covilhã, Portugal