The Impact of Cluttered JavaScript Code in Mobile Web Apps with respect to Energy Consumption

Geoffrey van Driessel
2639310
VU Amsterdam
g.r.van.driessel@student.vu.nl

Abijith Radhakrishnan
2667575
VU Amsterdam
a.radhakrishnan@student.vu.nl

Sophie Vos
2551583
VU Amsterdam
s.o.vos@student.vu.nl

Marc Wiggerman
2653796
VU Amsterdam
m.g.wiggerman@student.vu.nl

ABSTRACT

Context. Mobile phones are a major contributor to global internet traffic. Although, the energy usage of a single smartphone may be considered modest, the overall energy footprint left by mobile devices is far higher. Thiagarajan et al. [1], showed that JavaScript (JS) code mainly contributes to the energy consumption relative to other web elements. Chaqfeh et al.[2] introduced a tool, called *JSCleaner*, that de-clutters mobile web apps by removing non-essential JS code. They showed that this positively influences the Page Load Time.
Goal. Since energy-efficiency contributes to the user experience and the environmental impact of increasingly popular mobile web apps, we aim to contribute by investigating the impact of cluttered JS code on energy consumption. The goal of our study is: "Analyze cluttered JS code for the purpose of evaluation with respect to the energy consumption from the point of view of software developers in the context of mobile web apps".
Method. To accomplish this goal, we conducted an empirical experiment to measure and compare the energy consumption of cluttered and de-cluttered mobile web apps. We randomly selected 27 diverse, currently operational mobile web apps as subjects. Furthermore, we used Android Runner and the Batterystats profiler to measure the energy consumption.
Results. We tested the energy consumption for small, medium, and large mobile web apps. Our results showed that cluttered mobile web apps do not influence the energy consumption.
Conclusions. Hence, based on this experiment, we do not recommend developers who aim to reduce the energy consumption of their mobile web app to apply a de-cluttering engine. It should be noted that we tested the difference in energy consumption using a relatively small dataset and using a single de-cluttering engine. Therefore, further research is required to support this claim.

Green Lab 2020/2021, September–October, 2020, Amsterdam, The Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Geoffrey van Driessel, Abijith Radhakrishnan, Sophie Vos, and Marc Wiggerman. 2020. The Impact of Cluttered JavaScript Code in Mobile Web Apps with respect to Energy Consumption . In *Green Lab 2020/2021 - Vrije Universiteit Amsterdam, September–October, 2020, Amsterdam (The Netherlands).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

CISCO predicted global internet traffic in 2020 to be equivalent to 95x the volume of the entire global internet in 2005 [3]. Since the end of the second quarter of 2020, mobile devices (excluding tablets) have been a major contributor to global internet traffic. Mobile devices account for 51.53% of global web traffic and have an ascending trend [3]. Corresponding to the increased traffic, the complexity of modern mobile web apps has advanced in the previous decade, leading to larger mobile web apps that are computationally intense for mobile devices[1].

The increased complexity in mobile web apps can be mainly attributed to the inception of web and JavaScript (JS) frameworks (such as jQuery, Angular, and React) [4] which provide swift solutions for web development with function-specific modules and libraries that can fast pace the development process. The debut of Single Page Applications (SPAs) further paved the way for more frameworks and standardization of web development. Nevertheless, this also led to JS files in mobile web apps being bloated with unused and unnecessary functions from the frameworks and ergo higher computational complexity[2].

A computationally intensive website leads to higher energy consumption [5] as well as increased Page Load Time (PLT) [2]. It is shown that mobile web apps that have a PLT higher than 3 seconds, have a probability of losing 32% of its visitors [6]. Even if the energy usage of a single smartphone may be considered modest, the overall energy footprint left by mobile devices is far higher[3]. This is further extended by top-end smartphones and the 4G network [7]. Google handles the increasing energy consumption and PLT of mobile web apps by using lighter variants of the mobile web apps through Accelerated Mobile Pages (AMP) [8]. Another approach

[1] https://tomdale.net/2015/11/javascript-frameworks-and-mobile-performance/ accessed:25-09-2020
[2] https://www.toptal.com/javascript/are-big-front-end-frameworks-bad
[3] https://science.time.com/2013/08/14/power-drain-the-digital-cloud-is-using-more-energy-than-you-think accessed: 08-09-2020

for reducing PLT commonly used by web developers is concatenation, minification, and uglification of different JS files in the mobile web apps. Even though this approach provides a slightly faster PLT because of a smaller file size[4], JS computations in the browser remain the same. Furthermore, social media mobile web apps, such as Facebook, offer a lighter version of their service (Facebook Lite[5]) which is better suited for low-power devices.

Another method to reduce PLT by reducing the amount of JS code is introduced by Chaqfeh et al. [2]. Chaqfeh et al. [2] introduced *JSCleaner*: a classification algorithm that classifies critical, replaceable, and non-critical JS code. Accordingly, the replaceable JS code is replaced with HTML counterparts, and non-critical JS code is eliminated. This process is referred to as 'de-cluttering'. The authors studied the relationship between cluttered mobile web apps and the PLT. Their results showed a 50% reduction in requests and a 30% reduction in page size of the de-cluttered mobile web app compared to the original/cluttered mobile web app. They concluded that de-cluttering achieves a 30% reduction in PLT.

A different study, conducted by Thiagarajan et al. [1], showed that JS code has a relatively high energy consumption compared to other web elements. Since energy-efficiency contributes to the user experience and impact of increasingly popular mobile web apps, we aim to contribute by investigating the impact of cluttered JS code on energy consumption. In our study, we use JSCleaner to de-clutter JS code in mobile web apps. Our research question is: "*What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?*". We answer this research question by measuring and comparing the energy consumption of cluttered and de-cluttered mobile web apps. Our results showed that cluttered mobile web apps do not influence the energy consumption.

The remainder of this paper is structured as follows: Section 2 provides an insight into related work. Next, Section 3 defines the experiment using the GQM method. Afterward, we outline the experiment planning in Section 4 and describe the experiment execution in Section 5. Section 6 presents the results and Section 7 interprets and discusses the results. Lastly, we discuss the threats to validity in Section 8 and conclude the work in Section 9.

2 RELATED WORK

To the best of our knowledge, this paper is the first one to analyze and compare the energy consumption of mobile web apps using (1) the cluttered JS code base, and (2) the de-cluttered JS code base. Most of the research up until now focused on improving and analyzing the performance increase with relation to the PLT. First, we describe the work of Obbink et al. [9], who introduced a tool called *Lacuna*. Lacuna automatically eliminates dead JS code from mobile web apps, which in effect improves the PLT and energy consumption by reducing the required network and browser interpretation time. Obbink et al. [9] apply different analysis techniques to discover and eliminate dead code from the code base without changing the features of the mobile web app. After testing Lacuna on 29 mobile web apps, the results showed that Lacuna could be integrated into real-world build systems due to its reasonable execution time.

Second, Chaqfeh et al. [2] present a way to automatically de-clutter JS code using a tool called JSCleaner. The proposed JS de-cluttering tool helps to reduce PLT in mobile web apps without significantly jeopardizing its content and functionality. It follows a rule-based classification algorithm that divides JS code into three classes: non-critical, replaceable, and critical. It then deletes non-critical JS code from the mobile web apps, interprets replaceable JS components with their corresponding HTML code, and maintains critical JS code without any modification. The study then quantitatively analyzed 500 popular mobile web apps and reports an empirical evaluation in which a 30% reduction in PLT and a 50% reduction in the number of requests and size using the JSCleaner tool is presented. A qualitative evaluation followed in which the authors recruited 103 users to analyze the difference in functionality between cluttered and de-cluttered mobile web apps. The de-cluttered mobile web apps maintained the appearance of the cluttered mobile web apps with a median value of over 93%. Both papers [2, 9] introduce a different way to improve the PLT. However, they do not analyze the change in energy consumption of the cluttered versus the de-cluttered code base.

Other studies perform a similar research methodology, only to accomplish a different goal. For example, Malavolta et al. [10] performed an experiment to test the performance and energy efficiency of Progressive Web Applications (PWAs) while tuning the browser's cache behavior. The study involved measuring the energy consumption in joules and the PLT in milliseconds during the initial load of nine PWAs on a single Android device running in the Firefox browser. The authors concluded that the current results do not show evidence that adding caching to a website improves its energy efficiency. Despite the goal being different from the one presented in our study, the experimental setup and execution are similar to the ones presented in our study.

Another study that has a similar research methodology is the work of Chan-Jong-Chu [11]. The authors conducted an empirical experiment using mobile web apps with the aim to find a correlation between performance and energy consumption. The study has a clear research setup in which the correlation is proved using statistical tests together with a thorough description of the experiment execution. We provide a similar setup to ensure that the measured difference in energy consumption of cluttered and de-cluttered mobile web apps is statistically significant and the study can be precisely replicated.

Lastly, Thiagarajan et al. [1] studied the energy consumption of downloading and rendering mobile web apps on mobile devices by comparing the different energy usage (in joules) of web elements. This is relevant to our research as we aim to understand the energy consumption of a specific modification in a mobile web app. Thiagarajan et al. [1] aim to advise developers in designing energy-efficient mobile web apps. They concluded that the usage of JS code in mobile web apps has a negative effect on energy consumption compared to HTML elements. This is relevant to our study as we measure (a potential) energy gain by eliminating non-critical JS code or replacing the JS elements with HTML code. The main difference between the study of Thiagarajan et al. [1] and our study is that the work of Thiagarajan et al. is more descriptive (they are

[4]https://www.webfx.com/blog/web-design/improve-website-speed/ accessed: 14-09-2020

[5]https://www.facebook.com/lite

trying to understand a phenomenon), whereas we aim at discovering a specific relationship (i.e., the one between being cluttered and consuming different amounts of energy).

3 EXPERIMENT DEFINITION

As described in the related work section, JSCleaner reduces the size of JS files significantly and, therefore, reduces the PLT. However, it has not been shown that this actually reduces the energy consumption. The **goal** of our study is: "*Analyze cluttered JavaScript code for the purpose of evaluation with respect to the energy consumption from the point of view of software developers in the context of mobile web apps*". This goal has been established using the GQM method [12], the steps are presented in Table 1.

Analyze	cluttered JavaScript code
For the purpose of	evaluating
With respect to the	energy consumption
From point of view of	software developers
In the context of	mobile web apps

Table 1: Goal

Following our goal, we have identified the **research question (RQ)**: "*What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?*". We answer this research question by comparing the energy consumption of popular mobile web pages compared to their de-cluttered version. The mobile web apps are de-cluttered by JSCleaner.

To answer our research question the energy consumption (in joules) is used as **metric**. The energy consumption is calculated using the measured *power consumption (in watts)* times the *PLT (in seconds)*. Furthermore, we measured the page size (in MBs) to block a potential effect of page size on the energy consumption.

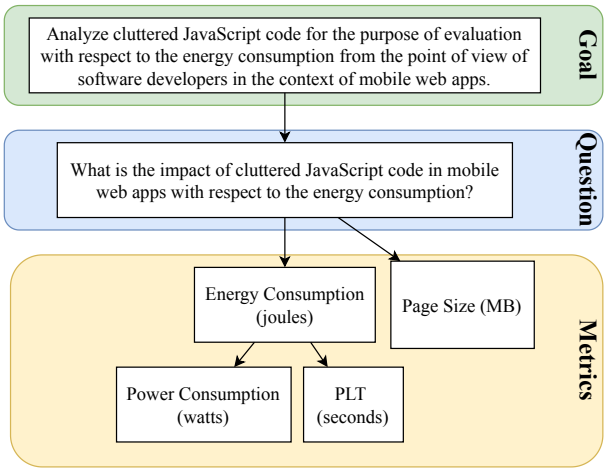

Figure 1: The GQM Tree

Figure 1 presents a visual representation of the GQM. It hierarchically shows how the goal is obtained using the research question, and how the research question is answered using the metrics.

4 EXPERIMENT PLANNING

In this section, we explain how the experiment is laid out. This is done in the following steps: context, subjects, experimental variables, hypothesis, experiment design, and data analysis.

4.1 Context Selection

To characterize the experiments of our study, the four dimension method presented by Wohlin et al. [13] is used.

The first dimension regards the choice between on-line and off-line experiments. As the experiments conducted in our study use a local Android device loading pre-developed mobile web apps, the off-line dimension is selected.

The second dimension represents the choice of using students versus professionals as subjects. This dimension is not suitable for our study as we use mobile web apps as subjects. No humans are the subject of the performed experiments.

The third dimension determines if the nature of the experiments are real or toy problems. As the mobile web apps used in this study are collected from real mobile web apps, the experiments and results presented in this paper can be regarded as covering real problems. Moreover, the outcome of our study could also directly impact real problems, i.e. developers could immediately tackle the energy efficiency of their real mobile web apps using a de-cluttering tool.

The fourth and final dimension defines whether the experiments are specific versus general. Our study can be regarded as general as it uses diverse and frequently visited mobile web apps to conduct the experiments. In other words, no specific type of mobile web app is selected.

4.2 Subjects Selection

The subject selection is done by a Python script (available through https://github.com/TheEnergyEngineers/ClutteredJS) that reads a CSV file containing 93 mobile web apps and their corresponding size. The mobile web apps are a subset of the mobile web apps used by Chaqfeh et al. [2] containing 93 popular mobile web apps. The set is obtained from The Majestic Million[6] and includes a wide variety of mobile web apps such as: news sites, education, sports, entertainment, travel, and government web apps. This ensures the generalizability of our sample to the real-world population of mobile web apps. The mobile web apps are available on a proxy server provided by Chaqfeh et al. [2]. The proxy server contains both the cluttered mobile web apps and the mobile web apps de-cluttered by JSCleaner.

To perform the analysis required for the subject selection, each mobile web app is downloaded locally using *Resources Saver Extension* for *Google Chrome*[7] after which its size is measured and a *Lines Of Code* analysis is performed using *CLOC*[8]. Each page load and download is given one minute for each stage to complete.

Before the subjects are selected, some mobile web apps have to be removed from the data set. First, pages with a size of 0 bytes are removed. This size occurs when the page failed to load on the proxy. Second, pages that occur more than once in the dataset are removed. The third group being removed are the pages which –

[6] https://majestic.com/reports/majestic-million
[7] https://github.com/up209d/ResourcesSaverExt
[8] https://github.com/AlDanial/cloc

according to CLOC – do not contain JS, HTML, or CSS code. For all these mobile web apps, CLOC did not report a value for any one of these three programming languages. After visually analyzing the deleted mobile web apps using Google Chrome, we observe that most mobile web apps do not show any visible changes after de-cluttering the mobile web apps. The complete deletion process results in 74 mobile web apps that are suitable for analysis.

After this cleaning step, the mobile web apps are divided into three groups based on the size of the cluttered mobile web app. These groups are created to control the effect of size on the energy consumption. The groups are created using three quantiles and contain the following size ranges:

Small	0.0 MB - 4.6 MB	(24 mobile web apps)
Medium	4.7 MB - 7.5 MB	(24 mobile web apps)
Large	8.1 MB - 46 MB	(26 mobile web apps)

After this division, the script randomly selects 10 mobile web apps from each group using a pseudo-random number generator[9] with a seed of 42. This process completes with a total of 30 mobile web apps (each mobile web app is used twice during the experiments, once as the cluttered version and once as the de-cluttered version). The selected mobile web apps and their information is presented in Table 2.

From Table 2, we observe that some de-cluttered mobile web apps are larger compared to the cluttered version. After a visual inspection, we found that most cluttered and de-cluttered mobile web apps do not differ in functionality. Some mobile web apps do show a difference. For example, the cluttered version of 'store.steampowered.com' shows a list of games and their corresponding images. In the de-cluttered version, these games are not loaded. This causes a difference in size between the two versions due to JSCleaner. In the case when the cluttered and de-cluttered versions do not differ in functionality, we often see that the total lines of code (JS, HTML, and CSS) increases in the de-cluttered version compared to the cluttered version. This again could be a side-effect of JSCleaner which alters the code of the cluttered version.

4.3 Experimental Variables

In our study, the *cluttering state* is regarded as the independent variable. The independent variable is both cluttered and de-cluttered once for each selected subject. The *energy consumption (in joules)* is regarded as the dependent variable which is measured in the performed experiments. The energy consumption is calculated by multiplying the *power consumption (in watts)* and the *PLT (in seconds)*.

To reduce the effect of the page size on the results, the size categories introduced in Section 4.2 are used as a blocking variable. Next, to reduce the effect of co-factors such as the browser type, network type, and device on the results, these co-factors are fixed to a single value.

4.4 Experimental Hypotheses

To test the research question, we have defined the following hypotheses. The null hypothesis states that the average energy consumption for cluttered mobile web apps is equal to the average energy consumption of de-cluttered mobile web apps:

$$H_0 : \mu_{E_{cluttered}} = \mu_{E_{de-cluttered}}$$

where E is the measured energy consumption and the subscript specifies the treatment. The alternative hypothesis states that the average energy consumption for cluttered mobile web apps is not equal to the average energy consumption of de-cluttered mobile web apps:

$$H_a : \mu_{E_{cluttered}} \neq \mu_{E_{de-cluttered}}$$

4.5 Experiment Design

The experiment is performed according to the following method: for each selected subject (which are described in Section 4.2) we have each treatment applied. This entails that there is a cluttered and de-cluttered version of each subject. In this way, each block of the mobile web app size is automatically balanced (crossover design). Both versions (cluttered and de-cluttered) of the mobile web app are loaded (in random order) on a smartphone to measure the power consumption and PLT. Our experiment design has the structure: 1 factor - 2 treatments [13]. The cluttering state is the factor and <*cluttered, de-cluttered*> are the treatments. The energy consumption is the measured outcome of the experiment. We repeat each trial (a combination of subject and treatment) 15 times to mitigate the effect of measurement errors. To summarize, the energy consumption of each mobile web app is measured 15 times for the cluttered version and 15 times for the de-cluttered version. Hence, as there are 10 subjects in each of the three blocks <*small, medium, large*>, there are: 3 (blocks) x (10 (cluttered mobile web apps) + 10 (de-cluttered mobile web apps)) x 15 (trial repetitions) = 900 measurements.

4.6 Data Analysis

First of all, the resulting energy consumption data is visualized and explored using multiple scatter and box plots to visually analyze the distribution of the data. Afterward, we check for normality using a Q-Q plot and the Shapiro-Wilk test with $\alpha = 0.05$ to decide which statistical test can be applied to test our hypothesis. Given the results provided by the two previous steps, we apply a paired t-test if the data is normally distributed. The paired t-test is done using the energy consumption measurements as data and the two treatments (cluttered and de-cluttered) as the two populations that have to be compared. In the case that the data is not normally distributed, we would use the Wilcoxon signed-rank test instead of the paired t-test. Following that we have three blocks, we split the data into three parts and execute the paired t-test per page size block: small, medium, and large. We then use the Bonferroni correction to keep the total p-value at 0.05. The effect size is the quantitative measure of the strength of a phenomenon. In our study, this refers to the actual difference in energy consumption if the cluttered and de-cluttered energy consumption differed significantly. As this is not the case in our study, the effect size is irrelevant.

5 EXPERIMENT EXECUTION

In our experiment, we use an Android smartphone to load the mobile web apps. The smartphone used for the experiment is the Google Pixel 3 and has an octa-core (4x2.5 GHz Kryo 385 Gold

[9] https://docs.python.org/3/library/random.html

ID	URL	Cluttered Size	De-cluttered Size	Category	OG LOC JS	OG LOC HTML	OG LOC CSS	DC LOC JS	DC LOC HTML	DC LOC CSS
0	fc2.com	572KB	564KB	Small	4521	326	2210	4505	326	2210
1	telegram.org	920KB	828KB	Small	2794	321	6494	550	321	6494
2	www.nih.gov	1.3MB	2.1MB	Small	22457	4	555	29155	866	12125
3	www.office.com	1.7MB	18MB	Small	13458	1256	8119	238496	1338	21820
4	duckduckgo.com	1.8MB	360KB	Small	35433	109	10031	4	98	10031
5	www.rt.com	3.0MB	2.9MB	Small	38487	3425	21329	36237	3425	21329
6	www.ucsc.edu	3.3MB	2.0MB	Small	25922	4631	464	11441	410	4976
7	www.colorado.edu	3.6MB	264KB	Small	36375	4635	265	1630	4	80
8	pixabay.com	4.0MB	1.3MB	Small	34116	5417	4724	9582	786	4724
9	utk.edu	4.2MB	1.5MB	Small	36161	5545	8506	11624	921	8506
10	www.sindonews.com	4.7MB	2.2MB	Medium	60680	4189	4823	23064	204	616
11	www.osha.gov	4.7MB	4.5MB	Medium	34327	3144	19679	32073	3145	19679
12	missouri.edu	5.6MB	2.7MB	Medium	35247	5403	15862	8301	772	15862
13	www.booking.com	6.0MB	4.1MB	Medium	61556	10902	54391	35001	10396	53983
14	www.mdpi.com	6.2MB	3.5MB	Medium	41608	9578	16062	17075	4947	16062
15	www.raspberrypi.org	6.2MB	5.9MB	Medium	46908	16545	464	44657	15820	464
16	www.homedepot.com	6.6MB	5.5MB	Medium	16739	3067	3	14715	3067	3
17	ae.godaddy.com	6.8MB	6.8MB	Medium	92350	8111	31031	92352	8112	31031
18	store.steampowered.com	7.1MB	6.9MB	Medium	17373	13846	11567	15129	13846	11567
19	www.indeed.ae	7.3MB	6.3MB	Medium	61220	4714	20372	42071	2464	19859
20	www.typeform.com	8.1MB	3.7MB	Large	102582	8770	1044	43485	4138	1044
21	www.researchgate.net	8.1MB	4.7MB	Large	116086	5196	27014	77066	560	27014
22	www.yelp.com	8.7MB	8.4MB	Large	56665	17865	43263	47658	17865	43263
23	kakaku.com	9.0MB	6.3MB	Large	101395	13735	6548	78069	9104	6548
24	steamcommunity.com	9.1MB	1.5MB	Large	22514	1940	8093	20193	538	8093
25	www.tistory.com	9.6MB	9.6MB	Large	24184	495	7023	24082	495	7023
26	www.netflix.com	11MB	11MB	Large	1083	23732	5781	1083	23732	5781
27	www.worldbank.org	11MB	12MB	Large	60677	62704	63298	57052	64838	63298
28	www.ltn.com.tw	28MB	28MB	Large	225749	6517	25981	223069	6102	25981
29	www.thetimes.co.uk	31MB	26MB	Large	354557	25924	787	338192	25500	768

Table 2: The subjects used during the experiments (OG = Original, DC = De-Cluttered, LOC = Lines of Code).

and 4x1.6 GHz Kryo 385 Silver) processor with 4 GB of RAM. This smartphone runs Android version 8.0.0. To mitigate the effects of background processes, we disable services we do not use such as Bluetooth, Location Services (GPS), and NFC. To ensure that the experiment stays running while minimizing the impact of the display on the measured energy consumption, the screen brightness is minimized while ensuring that the phone cannot go to sleep mode. Furthermore, we disabled all push notifications and non-critical apps.

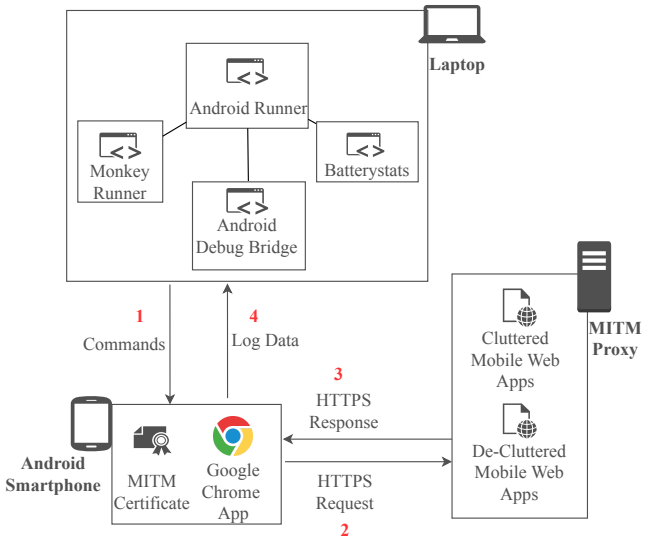

Figure 2: Experiment Execution

The smartphone is connected to a laptop using Android Debug Bridge (adb)[10]. The computer that we use to manage the experiment has an Intel i7 CPU (8th generation, 8 cores, and 4.2 GHz maximum clock speed) with 16 GB of RAM. Adb is a tool that allows communication between a development machine and an Android device. This communication entails information of the experiment configurations from the laptop to the smartphone and measurement data from the smartphone to the laptop. The Android smartphone and laptop are both connected to the same LAN. All the devices are in close proximity (less than a meter) to the WiFi router (802.11ac standard and 5 GHz frequency band) to ensure minimum latency. The Android smartphone uses the Google Chrome web browser version 66.0.3359.158 to access the internet and load the cluttered and de-cluttered mobile web apps. The researchers of the JSCleaner paper [2] cached the cluttered and de-cluttered mobile web apps, which they used for their research, on a proxy server and shared this proxy with us. In order to access the proxy server, the mitmproxy[11] certificate is installed on the smartphone. Mitmproxy is an HTTPS proxy that allows (among other features) interactive HTTPS requests.

In order to execute the experiment, we use Android Runner (AR)[12]. AR is a tool that facilitates the execution of measurement-based experiments on native apps and mobile web apps running on Android devices [14]. AR allows us to fill in the experiment configurations and setup after which it communicates this to the smartphone. After the experiment is conducted, AR stores the recorded measurement data on the laptop. AR utilizes Monkey Runner[13] to implement an API that grants control over the Android smartphone. We use Monkey Runner to construct a log of smartphone input commands. This log can be used to automate the interactions with

[10] https://developer.android.com/studio/command-line/adb
[11] https://docs.mitmproxy.org/
[12] https://github.com/S2-group/android-runner
[13] https://developer.android.com/studio/test/monkeyrunner

the smartphone. Lastly, we use the Batterystats[14] utility to measure the energy consumption. We chose this profiler as it was compatible with the used smartphone. Furthermore, Tan et al. [15] found an indication of Batterystats providing more accurate results of the power consumption compared to Trepn[15]. Batterystats is a plugin of AR.

A high-level view of our experiment execution is visualized in Figure 2. The experiment includes three main components: the *Laptop*, *Android Smartphone*, and *MITM Proxy*. The experiment starts with sending commands from the *Laptop* to the *Android Smartphone* (step 1). The connection between the *Laptop* and the *Android Smartphone* is established using *Android Debug Bridge*. *Monkey Runner* handles the automation of the external control of the smartphone (e.g. Terms and Conditions in browser app) and *Batterystats* takes care of measuring the power consumption. These processes are orchestrated and managed by *Android Runner*. Once the *Android Smartphone* receives the commands, it requests (step 2) the relevant *Cluttered or De-Cluttered Mobile Web Apps* from the *MITM Proxy*. The *MITM Proxy* sends (step 3), after checking the *MITM Certificate*, the *Cluttered or De-Cluttered Mobile Web Apps* to the *Android Smartphone*. Lastly, the log data and measurement results are collected by *Android Runner* (step 4).

After each run, the Google Chrome browser is cleared and closed. Then, the smartphone remains idle for 2 minutes to ensure that all processes are finished. To guarantee the intrinsic variability of the experiment, each trial is repeated 15 times. We repeat this measurement for 10 randomly selected cluttered mobile web apps and their de-cluttered version.

6 RESULTS

This section presents the results obtained from running the experiments described in Section 5. The Android Runner code, Python scripts for the execution, raw data, and R scripts of the data analysis are publicly available on: https://github.com/TheEnergyEngineers/ClutteredJS. Before the results are presented, we want to note that during the execution, the smartphone was unable to load typeform.com from the 'Large' size category, and rt.com from the 'Small' size category. To re-balance the data we randomly selected and removed ae.godaddy.com from the 'Medium' size category. This resulted in 9 subjects per category resulting in 3 (blocks) x (9 (cluttered mobile web apps) + 9 (de-cluttered mobile web apps)) x 15 (trial repetitions) = 810 measurements.

6.1 Intrinsic Variability

We aggregated the raw data by calculating the average energy consumption and standard deviation over the 15 repetitions of each trial. In the remaining graphs and results, we use the average energy consumption for each (cluttered and de-cluttered) subject over these 15 repetitions. First, we discuss the standard deviations (SDs) of the trials to understand if the data is influenced by external factors. Ideally, the SD for each trial should be low as each experiment run with the same subject and cluttering state should have a similar energy consumption in a non-corrupted experiment setting.

[14]https://github.com/S2-group/android-runner/tree/master/AndroidRunner/Plugins/batterystats
[15]https://github.com/S2-group/android-runner/tree/master/AndroidRunner/Plugins/trepn

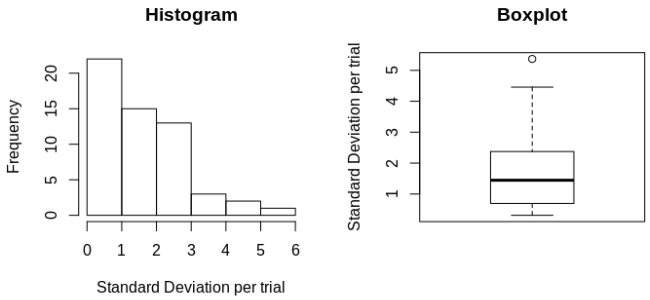

Figure 3: Variance in the repeated trials

The variability of the trials is visualized in Figure 3. In the histogram, we observe that most trials have an SD between 0 and 1. The secondlargest groups have an SD between 1 and 3. The remaining subjects have an SD between 4 and 6. From the box plot, we observe that most subjects have indeed small SDs. However, there are some outliers and subjects with a larger SD.

6.2 Data Distribution

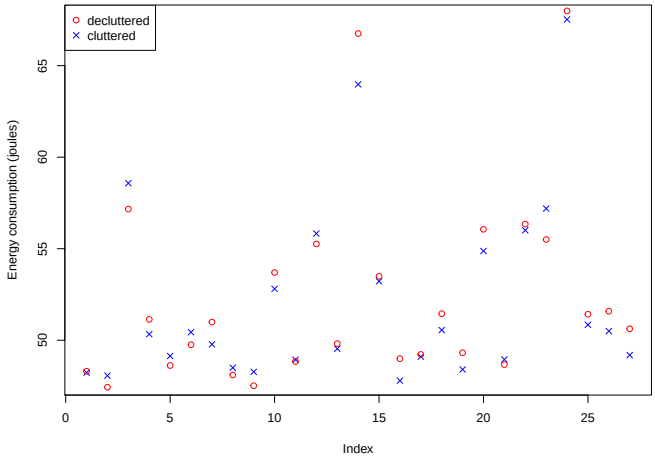

Figure 4: Cluttered and de-cluttered joule measurements

Second, we present the data distributions of the average (over the 15 repetitions) energy consumption for each cluttered and de-cluttered mobile web app. Figure 4 presents a scatter plot containing the energy measurement for all subjects. From this figure, we observe that the measurements of the de-cluttered mobile web apps are close to the measurements of the cluttered mobile web apps.

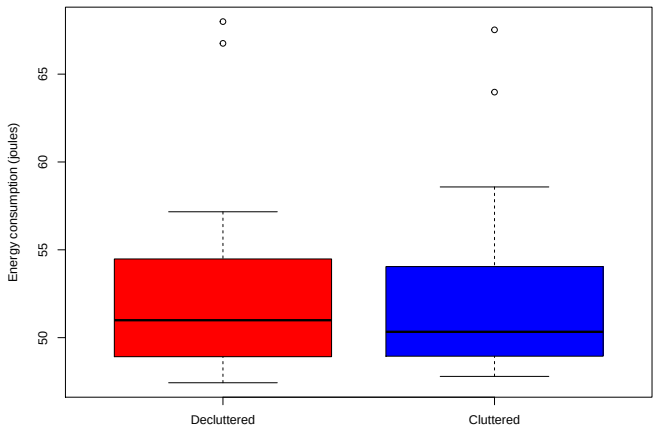

Figure 5: Boxplot of the energy consumption for cluttered and de-cluttered mobile web apps

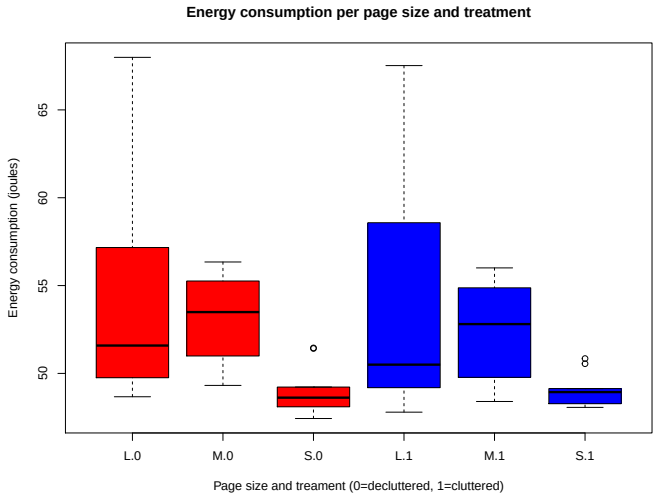

Figure 6: Boxplot of the energy consumption for cluttered and de-cluttered mobile web apps, blocked into three page size classes

Next, from the box plot in Figure 5 we observe that the mean energy consumption of the cluttered treatment is lower than the mean energy consumption of the de-cluttered treatment. This gives an indication that the de-cluttered mobile web apps do not have a smaller energy consumption, without taking into account the page size. To further visually analyze the difference between the two treatments, we take into account the page size and visualize this in Figure 6. We observe that the smallest page size has indeed the smallest mean energy consumption for both treatments. This is expected as a small mobile web app takes fewer system resources to load compared to larger mobile web apps. However, medium-sized mobile web pages break this pattern as, for both treatments, we observe that the medium page size has a higher mean energy consumption compared to the pages in the 'Large' category.

6.3 Testing Normality

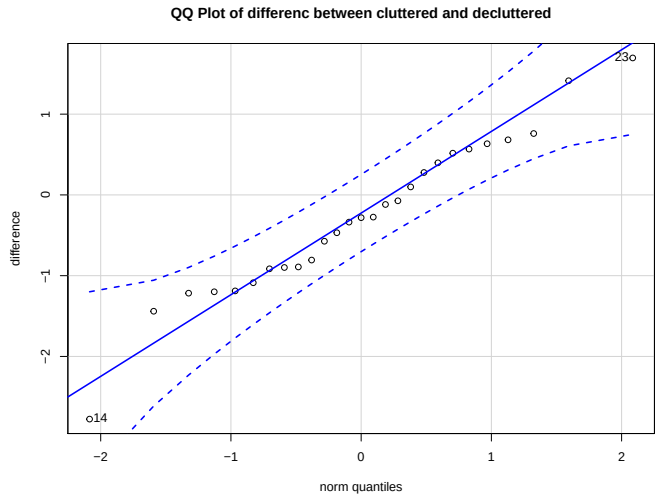

Figure 7: Q-Q Plot of the difference between the cluttered and de-cluttered results

Following the visualizations we test whether the data is normally distributed to determine which test is appropriate. In addition, we check if the difference between the treatments is normally distributed. This is required as a paired t-test is only applicable to data coming from a normal distribution. To execute these normality tests, we used the Shapiro-Wilk normality test. From this, we found a p-value of 0.746, from which we conclude that the data is normally distributed. However, because the sample is quite small (27) and the Shapiro-Wilk test has a limited accuracy on a small sample size, we also perform a visual inspection using a Q-Q Plot. This plot is shown in Figure 7. From this plot, we observe that the data is indeed approximately normally distributed as the data follows the diagonal line. This allows us to perform the paired t-tests.

6.4 Paired t-tests

Finally, we evaluated our hypothesis by executing paired t-tests on the observed data. First, we test whether the cluttered treatment has a significantly different mean compared to the de-cluttered treatment, without taking the page size into account. This test result is a p-value of 0.1472. Using the significance level α of 0.05 we conclude that without taking the page size into account, there is no significant difference in the energy consumption between the two treatments. Next, to test whether the treatment matters for the different page sizes, we run a paired t-test for each page individual page size category. From this, we find the following p-values: 0.6699, 0.01402, and 0.5268 for the small, medium, and large page size categories respectively. Using the p threshold of $\frac{0.05}{3} = 0.016$, which

is determined by the Bonferroni correction, we conclude that all three are insignificant. As the results are insignificant the effect size is irrelevant.

7 DISCUSSION

In this section, we elaborate on the obtained results and state the implications of these results. First of all, we studied the variance level of the repeated trials. A trial is a combination of a subject (mobile web app) and a treatment (cluttered or de-cluttered). We measured the energy consumption of each trial 15 times to understand the external effects on the measurements. From these results, we learned that most trials were relatively stable (the outputs of the different runs had a low SD). However, few trials did result in a relatively high (between 2 and 6) SD. This can be due to fluctuations in the network quality, internal processes from the phone interfering with the experiment, or measurement errors. The variance in the measurements harms the reliability.

Second, we investigated the data distribution. We observed that the energy consumption of the cluttered and de-cluttered mobile web apps was relatively similar. Furthermore, we compared the energy consumption for the small, medium, and large mobile web apps. It is striking that the mean energy consumption of the mobile web apps of a medium page size have a higher mean compared to the mobile web apps of a large size. This could be due to the relatively small sample size. Next, we tested the normality, from both the Shapiro-Wilk test and the Q-Q plot we concluded that the data is normally distributed.

The main research question of this research is "*What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?*". The null hypothesis states that the mean energy consumption of cluttered mobile web apps is equal to the mean energy consumption of the de-cluttered mobile web apps. In other words, we aim to find out whether cluttered JS code impacts the energy consumption. Using the results presented in Section 6, we are unable to reject the null hypothesis as the executed paired t-tests did not find p-values lower than the described values for α. The research question is thus answered by claiming that de-cluttering does not impact the energy efficiency of mobile web apps. This claim holds for small, medium, and large mobile web apps.

Our results show that cluttered mobile web apps do not influence the energy consumption. In the study presenting JSCleaner [2], the researchers did find that de-cluttering mobile web apps using JSCleaner reduces the PLT by 30% and the number of requests and page size by 50%. It thus seems that the choice of using a de-cluttering tool like JSCleaner depends on the goal. If the goal is reducing the PLT, it is a good decision to apply a de-cluttering tool. However, if the goal is to reduce the energy consumption, developers should not (with the current results in mind) apply such a de-cluttering tool.

Hence, the lessons learned of our study are that applying JSCleaner has no impact on the energy consumption. However, we do want to emphasize that more research is required to support this claim and generalize this for all de-cluttering methods. A major implication on the validity of the results, and thus the conclusion, is that a relatively small sample size is used and solely one de-cluttering method is applied.

8 THREATS TO VALIDITY

In this section, we discuss the threats to validity and mitigation measures. Validity is defined as the extent to which results are sound and applicable to the real world [13]. The used validity classification is proposed by Cook and Campbell [16].

8.1 Internal Validity

Internal validity considers the causality between treatment and outcome [13].

History To mitigate the effects of history, we ensured that both the cluttered and de-cluttered mobile web apps were cached at the same time. This ensures that the subjects did not change and can be properly compared. Next, the experiment trials are performed within the same time-frame. If this would not be a single type of time-frame, this could harm the validity as executing the experiment in a different time-frame could lead to different results. We mitigated this threat by performing the experiment on a regular day and night (Wednesday to Thursday and no special holiday). This ensured a representative influence of e.g. data-traffic and connectivity. Furthermore, we chose this setup to ensure that the experiment setting is similar in each trial.

Maturation To mitigate the effect of maturation, we ensured that after each run possible effects on the smartphone and experiment setup of the previous run are eliminated. This entails that we removed the caches of the web browser and paused the smartphone for 2 minutes after each run to ensure that activated background processes are terminated.

Selection A few mobile web apps were not available. To ensure that this is not some specific type of subject we checked the mobile web pages that were unavailable manually to understand the issue. The reason was that some mobile web apps did not properly load. This could be due to network issues or internal issues of the proxy server. The omitted mobile web apps were of different topics. However, we cannot be completely certain that this is not a specific type of subject.

Reliability of Measures In order to reduce the effects of co-factors and noise, we ensured that the experiment execution was similar for each experiment run. For instance, we requested the mobile web pages from the same proxy, through the same LAN, by the same device, using the same settings (e.g. browser, profiler). Moreover, we used the page size as a blocking factor to mitigate its effect on the results. However, not all experiment factors could be controlled. A threat to the internal validity is the fact that we used an external proxy to retrieve the subjects. Due to time constraints, we did not de-clutter the mobile web apps ourselves. This caused a level of uncertainty and lack of control over the web requests. Moreover, factors such as network instability and background processes within the device were out of our control. We mitigated these effects by turning off irrelevant applications, notifications, network and location services. Furthermore, the phone settings that could impact the energy consumption such as screen brightness remained constant throughout the experiment. To ensure the correctness of our measurements, we used a well-tested framework and profiler to manage our experiment. Finally, to guarantee the reliability of our measurements, we repeated each trial multiple times. The outcomes of the trial are relatively stable (most SDs are between 0 and 1).

However, there are outliers with a higher SD. This means that there are other processes interfering with the measurements.

8.2 External Validity

External validity concerns the generalizability of the experiment [13].

Interaction Selection and Treatment To guarantee the generalizability of the results, we selected diverse, real-world mobile web apps. The mobile web apps originate from The Majestic Million[16]. This selection contains operational mobile web apps from a wide variety of themes such as e-commerce, news, and sports. From this selection, we generated a random sample of subjects. A threat to the external validity is the relatively small sample size. The random selection mitigates this threat.

Interaction Setting and Treatment Another threat to the external validity is the controlled research setting. In the real-world, background processes such as push-notifications and location services are enabled. Hence, our experiment setting is not representative of a real-life setting. Furthermore, by ensuring that the research setup remains constant (e.g. using the same device and network connection), we threaten the external validity as, for instance, other devices and network conditions would perform differently. This is an inevitable trade-off between external vs. internal validity. We mitigated this effect by using a new device (Google Pixel 3) in a regular household setting.

Interaction History and Treatment To mitigate the effects of the interaction between history and treatment, we performed the experiment on a regular work-day. As the experiment took a relatively long time to execute, we did run it partly over night. This is a potential threat to the validity.

8.3 Construct Validity

Construct validity concerns the relation between theory and observation [13].

Inadequate Preoperational Explicitation of Constructs To ensure that the constructs are sufficiently defined, we applied the GQM method *a priori*. This method allowed us to accurately define the goal, research question, and metrics.

Mono-Operation Bias To ensure that we properly measured the effect of cluttered JS code on energy consumption, we measured the energy consumption of both the cluttered and de-cluttered version of the mobile web apps. Furthermore, we executed the experiment on different subjects and repeated each trial 15 times. However, the fact that the JS code is solely de-cluttered by one de-cluttering algorithm (i.e. JSCleaner) harms the validity of the experiment. Further research is required to test the effect of cluttered JS code on energy consumption using other de-cluttering algorithms.

Mono-Method Bias A threat to the construct validity is the fact that we used one metric (i.e. power consumption) to measure the energy consumption. This effect is mitigated by considering that it is an objective metric rather than a subjective metric.

8.4 Conclusion Validity

Conclusion validity concerns whether the relationship between the treatment and outcome is statistically correct and significant [13].

[16]https://majestic.com/reports/majestic-million

Low Statistical Power A threat to conclusion validity is our relatively small number of subjects. Due to time constraints, we tested 9 subjects within each category. The outcome of the statistical tests may be erroneous due to the small sample size. To mitigate this effect, future experiments are advised to test a larger sample.

Violated Assumptions of Statistical Tests A threat to the validity is that normality tests typically have a low power in small sample sizes. Considering our small sample size, it could be that the normality assumption is unjust. We mitigated this effect by detecting normality using two methods, namely, the Shapiro-Wilk test and the Q-Q plot.

Fishing and The Error Rate In our experiment, we conducted multiple statistical tests. Namely, for mobile web apps with a small, medium, and large page size. To mitigate the error rate of conducting multiple statistical tests, we adjusted the p-values accordingly.

9 CONCLUSIONS

In this study, we analyzed the energy consumption of cluttered JS code in mobile web apps. To do so, we randomly selected 27 subjects from the majestic million set. The subjects were de-cluttered using JSCleaner [2]. We utilized Android Runner [14] to orchestrate and manage the experiment, and the Batterystats profiler to measure the energy consumption. Furthermore, we used statistical tests to conclude whether there is a significant difference in the energy consumption of the cluttered and de-cluttered mobile web apps. From these tests, no significant difference was found between the energy consumption of the cluttered and de-cluttered mobile web apps. This implies that developers do not need to take the energy consumption into account when applying a de-cluttering engine to their code. Considering that we used a small sample size to perform our experiments and tested solely one de-cluttering algorithm (JSCleaner), future studies are required to support this claim.

Future work could analyze a more extensive set of mobile web apps. Furthermore, the experiments could be performed on more devices (e.g. low- and high-end), operating systems (e.g. both iOS and Android), and browsers (e.g. Chrome, Firefox, Safari). Experiments done using these additions could substantiate the generalizability of the results and conclusions. In addition, more de-cluttering methods should be applied to the mobile web apps to test the difference between the effectiveness of the de-cluttering methods.

REFERENCES

[1] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery? analyzing mobile browser energy consumption," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 41–50. [Online]. Available: https://doi.org/10.1145/2187836.2187843

[2] M. Chaqfeh, Y. Zaki, J. Hu, and L. Subramanian, "Jscleaner: De-cluttering mobile webpages through javascript cleanup," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 763–773. [Online]. Available: https://doi.org/10.1145/3366423.3380157

[3] CISCO. Global - 2020 forecast highlights. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf

[4] M. Persson, "Javascript dom manipulation performance: Comparing vanilla javascript and leading javascript front-end frameworks," 2020.

[5] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down," in *International Workshop on Power-Aware Computer Systems*. Springer, 2003, pp. 26–40.

[6] Google. Find out how you stack up to new industry benchmarks for mobile page speed. [Online]. Available: https://think.storage.googleapis.com/docs/mobile-page-speed-new-industry-benchmarks.pdf

[7] X. Chen, Y. Chen, M. Dong, and C. Zhang, "Demystifying energy usage in smartphones," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–5.
[8] Google. Amp on google. [Online]. Available: https://developers.google.com/amp
[9] N. G. Obbink, I. Malavolta, G. L. Scoccia, and P. Lago, "An extensible approach for taming the challenges of javascript dead code elimination," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 291–401.
[10] I. Malavolta, K. Chinnappan, L. Jasmontas, S. Gupta, and K. Ali Karam Soltany, "Evaluating the impact of caching on the energy consumption and performance of progressive web apps," 10 2020.
[11] K. Chan-Jong-Chu, T. Islam, M. M. Exposito, S. Sheombar, C. Valladares, O. Philippot, E. M. Grua, and I. Malavolta, "Investigating the correlation between performance scores and energy consumption of mobile web apps," ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 190–199. [Online]. Available: https://doi.org/10.1145/3383219.3383239
[12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in software engineering." Springer US, 2012.
[13] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
[14] I. Malavolta, E. M. Grua, C.-Y. Lam, R. de Vries, F. Tan, E. Zielinski, M. Peters, and L. Kaandorp, "A Framework for the Automatic Execution of Measurement-based Experiments on Android Devices," in *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '20)*. ACM, 2020. [Online]. Available: https://github.com/S2-group/android-runner/blob/master/documentation/A_Mobile_2020.pdf
[15] K. H. Tan and I. Malavolta, "Integration of Android battery statistics into a Python framework for Green experiments," in *International Workshop on Power-Aware Computer Systems*. Vrije Universiteit Amsterdam.
[16] T. D. Cook, D. T. Campbell, and A. Day, *Quasi-experimentation: Design & analysis issues for field settings*. Houghton Mifflin Boston, 1979, vol. 351.