Native app vs their web app counterpart, which one consumes more?

Terry Bommels
2614413
VU Amsterdam
t.bommels@student.vu.nl

Daniël Hana
2614426
VU Amsterdam
d.hana@student.vu.nl

Dion David Haneveld
2614399
VU Amsterdam
d.d.haneveld@student.vu.nl

Stijn Meijerink
2639627
VU Amsterdam
s.d.j.meijerink@student.vu.nl

Luciën Martijn
2643464
VU Amsterdam
l.martijn@student.vu.nl

ABSTRACT

Context. Nowadays, many tools and technologies are available to developers to create and publish their mobile apps. Aside from their data limit, mobile phone users are limited by their phone's battery capacity, wanting to get the most out of it. Android developers can use this knowledge to publish their app on a platform that is less energy consuming than its competition, making it more attractive for the user to use the app.
Goal. The purpose of this research is to compare the energy consumption of native Android apps against their web app counterparts, with the use of monkeyrunner and batterystats profiler. This research can help developers decide whether they should offer their mobile app as a native Android app or web app, since both already have their own strength and weaknesses. Energy consumption can be a deal breaker when the mobile app is new and make it unappealing for the end user, where this research can help both end-users and developers make a decision between native or web.
Method. For this research an experiment with two factors has been designed. One is the app type (web or native) and the other is the device tier (low- or high-end). The experiment has been conducted with a full factorial design on a sample of eight subjects. The response variable is the energy consumption of the device. Due to energy consumption inherently having a high fluctuation, every trial has been repeated fifteen times.
Results. The results show that the app type for both high-end and low-end device have a significant impact on the energy consumption. Furthermore, the effect size of the energy consumption is considered large by utilizing the Cohen and Cliff's effect size measures.
Conclusions. The conducted research shows a first empirical study on energy consumption between native and web apps on the Android platform, with the usage of automatized user-behaviour scripts performed on eight apps. The experiment results show a promising difference in energy consumption between native apps and their web counterparts.

1 INTRODUCTION

Nowadays mobile devices are used in an ever increasing manner for all kinds of purposes such as reading news, watching videos and communicating with others. The Comscore's Global State of Mobile reports that mobile apps comprise up to 63% and mobile web comprise up to 7% share of the time spent on digital media in 2019 [1].

Many apps and tools are evolving over time and new features are continuously developed and applied to existing integrated development environments (IDEs) such as Android Studio [1]. Native mobile apps can be developed with different technologies such as Java, Kotlin [2] Flutter [3] and React Native [4] to name a few. Web apps that run in the mobile browser are built in HTML5, CSS3 and JavaScript. Introduced in 2015 by Google, the Progressive Web App [5](PWAs) is a new approach that combines the functions of a native app (e.g. device hardware access, push notifications) and is intended to work on any browser solely by the means of web technologies such as HTML5, CSS3 and JavaScript. This mid way approach gives end users an experience almost identical to the native app counterpart in the browser.

Choosing the right development technologies to launch a mobile app is a critical step in order to launch a successful mobile app to the market. Both web and native have their share of benefits and drawbacks. Choosing the best approach depends largely on the desired requirements and the available resources (time, money, etc.) [2].

Mobile devices have a limited amount of power. Making sure the user gets the most out of this limitation can be essential for an app to be successful and can be regarded by some as the most important aspect of the mobile experience[6]. Knowing which platform consumes the least amount of energy, native or web, can be of great importance when a user's mobile device is low on power.

Several articles on the web have been written which state that Facebook and Twitter native apps for both iOS and Android are the culprits of a quickly depleted battery, due to several running background processes of the native app. The authors offer several tips in order to reduce the battery drainage of these apps, or encourage readers to switch over to the web version of the app which reportedly drains the battery less [3][4][5].

The goal of this paper is to evaluate the difference in the energy consumption between native Android apps and their web versions.

[1]https://developer.android.com/studio
[2]https://kotlinlang.org/
[3]https://flutter.dev/
[4]https://reactnative.dev/
[5]https://web.dev/progressive-web-apps/
[6]https://developer.android.com/topic/performance/power

Similar research has been conducted in the past, however to our knowledge (see section 2) the literature primarily focused on energy consumption in the context of either native- or web apps. This research will compare the energy consumption between the two app types. Two great examples of these empirical studies are [6] on energy consumption of native Android apps and [7] on assessing the impact of service workers on the energy efficiency of progressive web apps (PWAs).

During this experiment, monkeyrunner[7] will be used to automatically simulate user behaviour on a selection of apps. Monkeyrunner emulates app user behaviour programmatically, such as installing an app, making keystrokes and taking screenshots, which alleviates the error introduced by manual behaviour of human subjects.[8]

The **main contributions** of this paper are:

- the results on energy consumption of eight android apps and their web app counterparts;
- a discussion and conclusions on the results of the experiment on energy consumption
- the experiment replication package containing all automated interaction scripts.

As mentioned earlier, mobile developers have multiple techniques to develop their envisioned app. This research can give the mobile developers an extra criteria to select a technique in the form of energy consumption. Furthermore the user also has an option to choose, because many companies offer a native and web version of their app. This way a user can make an informed choice about the energy impact.

This paper presents a more formal definition of our goal and research questions. These are presented in section 3. Furthermore, a detailed definition of the experiment with a planning on the steps of the experiment is given in section 4. Subsequently in section 5 all the tools, devices, metrics and a description of the overall software/hardware infrastructure, are defined in detail with a rationale. Next, we conduct the experiment and show the retrieved results in section 6. In section 7 we discuss the findings of our results. An overview of threads to validity of the experiment is given in section 8. Finally we conclude all our main findings/results to our given research questions and see whether the result can be improved by utilizing new inventions explained with other research in section 9.

2 RELATED WORK

As stated before, previous research has been conducted in the field of energy consumption. This section will describe the various research conducted and how this paper varies in respect to other papers. Possible research gaps and how our research fits in this scenery will be explained as well. To get a short overview of the literature field, the following query was run on Google Scholar: "energy consumption AND (android app OR web app)".

Oliveira *et al.* explores the differences in energy consumption on the android platform with a focus on programming languages and CPU usage. The authors used multiple benchmarks to evaluate the energy consumption of the both widely used languages Java and JavaScript in the context of Android hybrid- and native apps. It was shown that combining the two languages led to a negligible improvement in terms of energy consumption, however preliminary results show that JavaScript operations were less CPU intensive, hence resulting in less energy consumption [9].

Ma *et al.* presents a comparative study on the performance of native apps and Web apps from the perspective of Web services, where performance was measured in terms of number of HTTP requests, response time, data- and energy consumption. It was shown that the performance of web apps was better than native apps in almost a third of the cases. The energy consumption was measured by employing a Monsoon Power Monitor and using a trace consisting of HTTP(S) requests and responses, where the authors start measuring at the first issued network request and stop measuring of the last received response in order to calculate the energy consumption. Furthermore the experiment was conducted by two human users who concurrently interacted with their app [10].

Li *et al.* conducted an empirical study on energy consumption of android apps. Based on 405 real-world market applications the authors find that apps on average spend 61% of the energy while being idle state where the network component is the most dominant energy consumer. Additionally, it is found that only a few APIs used dominate non-idle energy consumption. This research helps to find where native android apps consume most energy, however it only compares native android apps and not their web app counterpart [6].

Li and Halfond discuss a technique that brings down the energy consumption of HTTP request in android apps. The work builds up on a previous published work of the same authors in which they showed that 32% of non-idle power consumption is lost on HTTP-requests. The study shows that by bundling a chain of HTTP requests into one HTTP request yields a 50% reduction in energy consumption in two Android apps that were selected for the experiment [11].

Malavolta *et al.* discuss the impact of service workers on the energy efficiency of PWAs by means of an empirical study. The authors show that service workers have a negligible impact on the energy consumption of PWAs on Android devices under different network conditions (2G and Wi-Fi) for seven PWAs [7].

The previously mentioned research shows promising results in the field of energy consumption for both Android native- and web apps. Most previous work focuses on the performance of Android apps compared to each other or to PWAs, while others looked at the deeper workings and comparisons of the programming languages. The goal of this paper is to fill this gap, focusing mainly on the energy consumption of native apps versus web apps on Android. The main difference is that this research is done by simulating android user interaction using an automated interaction tool. This will ensure that there is no human replication error during the experiments, which will produce more reliable and reproducible measurement results.

3 EXPERIMENT DEFINITION

The scope of this experiment has been captured by carefully following the guidelines of the Goal Question Metric (GQM) approach by Basili et al [12]. In the next subsections the specifics of the experiment definition are expressed in the form of a Goal (Section 3.1), Questions (Section 3.2) and Metrics (Section 3.3).

[7]https://developer.android.com/studio/test/monkeyrunner

3.1 Goal

The goal of this experiment is formally defined by applying the GQM template and is as follows:

Analyze	native- vs web apps
For the purpose of	Evaluation
With respect to	Energy consumption
From the point of view of	end users and software developers
In the context of	Android
Result	
Analyze **native- vs web apps** for the purpose of **evaluation** with respect to **energy consumption** from the point of view of the **end users and software developers** in the context of **Android**.	

Table 1: Goal definition

3.2 Questions

The goal of our experiment can be expressed into the following quantifiable questions.

(RQ1) To what extent is the energy consumption of a native app different from their web counterpart during app usage?

(RQ2) To what extent does the tier (e.g. low-end) of the device impact the difference in energy consumption between mobile native- and web apps?

By answering these questions, software developers might better understand the differences in energy consumption between native and web during user usage and its impact on energy consumption in both low- and high end devices. With the new gained insights, software developers can utilize that information to improve the energy efficiency of both native- and web apps. Also end-users that have a low-end device, which tends to consume more energy, are able to choose the right app type which has lower energy consumption.

Furthermore, developing countries' phone users are slowly transitioning from cellphones to low-tier smartphone devices [13]. These devices might run on older Android versions[8], which means that there is a high probability that the user may run into incompatibility issues with respect to the latest native apps.

Therefore, the user might want to make an informed choice on the viability of using the web app version instead of the native one in terms of energy consumption and incompatibility issues.

3.3 Metrics

In order to answer the defined research questions in section 3.2 we have identified the following quantifiable raw metrics. The **energy consumption** E_Δ in a timeframe t_Δ is a derived metric, expressed in joule (J), of the either the native app or the web counterpart.

Each timeframe's start and end is characterized by component states such as CPU frequency, audio and Wi-Fi. The raw metrics which are gathered by AndroidRunner's *batterystats* plugin [9] are summed up below:

- **Current Power Intensity** I_Δ measured in milli-ampere (mA) consumed in one second in one timeframe. Each component's current power intensity is defined in the device's *power_profile.xml*. It is calculated as follows:
$$I_\Delta = \sum_{\forall c \in C} I_{\Delta,c,s} \quad (1)$$
where C is the set of hardware components and $I_{\Delta,c,s}$ is the current power intensity of component c with the state s within one timeframe.
- **Device Voltage** V_Δ measured in volts (V) in one timeframe.
- **Time** t_Δ measured in milliseconds (ms). It represents the amount of time a device component spends in its current state in one timeframe.

The energy consumption E_Δ of the device in one timeframe can then calculated with the following formula[14]:

$$E_\Delta = I_\Delta V_\Delta t_\Delta \quad (2)$$

The total energy consumption metric E of the device is subsequently calculated by summing up all the elapsed timeframes during the start of the profiling by monkeyrunner in one trial:

$$E = \sum_{t_\Delta \in T} E_\Delta \quad (3)$$

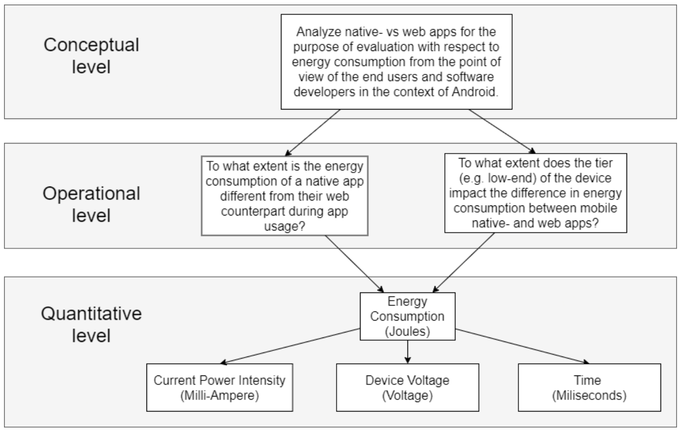
Conceptual level
Analyze native- vs web apps for the purpose of evaluation with respect to energy consumption from the point of view of the end users and software developers in the context of Android.
Operational level
To what extent is the energy consumption of a native app different from their web counterpart during app usage?
To what extent does the tier (e.g. low-end) of the device impact the difference in energy consumption between mobile native- and web apps?
Quantitative level
Energy Consumption (Joules)
Current Power Intensity (Milli-Ampere)
Device Voltage (Voltage)
Time (Miliseconds)

Figure 1: GQM tree of experiment

4 EXPERIMENT PLANNING

4.1 Subjects Selection

Regarding the selection of native Android apps and their web counterparts, due to time constraints of this project only eight apps will be selected. Furthermore the user interface and the feature set between the native app and the web version need to be similar, such that the same interaction steps/use cases can be carried out. The web app should be able to run in Chrome for Android, due to the fact that Chrome is the most used mobile browser [10]. Furthermore

[8]Based on https://gs.statcounter.com/android-version-market-share/all/africa

[9]github.com/bommels/android-runner/tree/master/AndroidRunner/Plugins/batterystats

[10]Based on statistics of August 2020: https://www.w3counter.com/globalstats.php

to have an interaction which is as close as possible to a normal user interaction, the subjects will be selected such that no login is required and a standard usage scenario can be performed. Additionally the selected apps should be available on APKPure. Lastly, the selected native apps should be able to run on both low-end and high-end Android devices, thus supporting both Android versions.

The apps have been selected by making use of the utility website AndroidRank [11]. The categories were selected by the most popular and subsequently selected the apps of these categories that were viable for both web and native with the most installs. In order to compile a list of heterogeneous group of apps, only one app is selected per category and per company. These constraints have led to the following eight apps, which are listed in table 2.

App	Category	Version
AliExpress	Shopping	8.17.1
TripAdvisor	Travel & Local	38.0
Deliveroo	Food & Drink	3.47.0
Reddit	News & Magazines	2020.37.0
The weather channel	Weather	10.18.0
9GAG	Entertainment	6.89.04r22929
Youtube	Video Player	15.40.37
Zara	Lifestyle	9.4.0

Table 2: Selected apps based on criteria list

4.2 Experimental Variables

In order to answer the two research questions, we consider the following experimental variables as can be seen from Table 3.

The independent variable for the first research question is the **app type**, with two treatments, namely *native app* and *web app* where the tier of the device is constant. For the second question, the first independent variable is **device tier**, which has two treatments, *low-end* and *high-end* and the second is **app type** with two treatments, *native app* and *web app*.

The **dependent variable** for both research questions is the *energy consumption* of the device which is a continuous ratio level of measurement. .

One co-factor that needs to be mitigated is the *caching behaviour* of (web) app data. We do not have any insight in the caching behaviour of the apps during the experiments. Therefore, we empty the native app- and browser cache before each run conducting the experiment. This will result in more reliable experiments.

4.3 Experimental Hypotheses

The following hypotheses have been formulated in order to answer the two research questions.

Given that mean energy consumption τ_{web} where app type is web app and the mean energy consumption τ_{native} where app type is native, then the null and alternative hypotheses for **RQ1** are as follows:

There is no difference in energy consumption between native app and their web app counterpart:

$$H1_0 : \tau_{native} = \tau_{web}$$

There is a difference in energy consumption between native app and their web app counterpart:

$$H1_1 : \tau_{web} \neq \tau_{native}$$

RQ2 considers an additional factor on the energy consumption, namely the tier of the device. Here, τ_i is the first factor (app type) and β_j is the second factor (device tier) and their interaction on each other $\tau\beta_{ij}$. Now, the null and alternative hypothesis are then formulated respectively:

The device tier and the app type do not interact in affecting the energy consumption:

$$H2_0 : (\tau\beta)_{ij} = 0 \forall i, j$$

The device tier and the app type do interact in affecting the energy consumption:

$$H2_1 : \exists(ij)|(\tau\beta)_{ij} \neq 0$$

4.4 Experiment Design

In order to capture all measurement information needed to answer both research questions, a 2-factor 2-treatment (2x2) full factorial design is chosen. With this design, there is no need to do two separate experiments for each research question. Each app will receive the *native app* and *web app* treatment as well as the *low-end* and *high-end* treatment. As energy consumption measurements in mobile devices have a high intrinsic variability [15], this is mitigated by applying the *repeated measures* design for each trial of the experiment. Measurement data for each subject-treatment combination are collected by conducting 15 trials of the experiment. Each treatment is assigned to a subject in random fashion. Finally, *balanced design* is applied, each combination is assigned to an equal number of apps. An example of a 2 level full factorial design table of an experiment run for the Facebook app can been found in Table 3. This can be generalized for all other apps in random order of treatments for each trial run.

Every app will be controlled by an automatic interaction script for a duration of one minute. More information on the exact one minute interaction for each app is explained in Section 5.2. To account for the hardware components not being kept alive, a cool-down period between runs for two minutes is enforced. This ensures at the start of every run the hardware components are in idle mode. The total run-time of the experiment is calculated as follows: $8\ apps \times 2\ app\ types \times 2\ device\ tiers \times 3\ minutes \times 15\ trials = 24\ hours$.

4.5 Data Analysis

The data will be initially analyzed by means of numerical and graphical summaries, such as sample mean, sample variance and boxplots. The normality (i.e. whether it stems from a normal distribution)

[11] https://androidrank.org based on results from 9-13-2020

App	App Type	Device Tier
Reddit	native	low
Reddit	native	high
Reddit	web	low
Reddit	web	high

Table 3: Example 2L full factorial design table for Reddit

of the measurements will be analyzed with visual inspection by means of a Q-Q plot. In terms of normality check completeness, the *Shapiro-Wilk test* will be applied in order to test normality of the data in a quantitative manner.

In the case that the measurement data does follow a normal distribution and is homoscedastic (which is verified with *Levene's test*), the hypotheses of both research questions will be tested by applying the *two-way analysis of variance* (2-way ANOVA) technique on the measurement data, where **RQ1** can be answered by looking at the p-value result of the first factor (app type). **RQ2** can be answered by looking at the p-value result of the effect of interaction between the two factors (app type and device tier).

If the measurement data happens does not follow a normal distribution, transformations may be applied which preserve the underlying properties of the data. Such transformations will potentially lead to a higher statistical power of the hypothesis tests. If the transformed data normality deviates moderately from the normal distribution, then applying 2-way ANOVA on the data is still a suitable option. As using a large number of non-normal distributions show that the false positive rate is not affected much by this violation of the non-normality assumption [16][17][18].

However, it could still be the case that the transformed data deviates too much from a normal distribution. In this case, 2-way ANOVA statistical test will not be applicable anymore. This will result in dropping **RQ2**, as the device type will become a *blocking factor* in this case. The data will be split in two blocks, namely high- and low-end device tier, where each block will be analyzed separately for descriptive statistics, normality checks and statistical tests. This also means that **RQ1** will be answered in two-fold. If the data in a block stems from a normal distribution, then a *Paired t-test* is utilized. When this is not the case, the non-parametric counterpart, which is the *Wilcoxon signed-rank test*, is utilized.

All statistical tests will be executed with a significance level of $\alpha = 0.05$, as we consider this threshold to be small enough (i.e. this experiment does not have any implications on human lives in case of a type II error) in order to test the null hypotheses of our two research questions.

5 EXPERIMENT EXECUTION

We will use a collection of tools available to conduct our experiments. For the collection of data and automation we will be using Android runner. Android runner includes monkeyrunner, a tool that can be used to automate input and read data from (simulated) android devices, and Android Debug Bridge (adb), which is a toolkit that makes it possible to communicate with a device.

We will be using a forked version of Android runner, release tag 'gz-exp-1' [12]. This is a snapshot of the state at which we conducted the experiments, and can always be retrieved later to recover this state.

Android runner supports multiple plugins to profile device usage. We will be using the 'batterystats' plugin, to measure energy consumption and battery drain. Figure 2 shows the overall hardware and software infrastructure.

Regarding the web apps, the low-end will use Google Chrome version 86.0.4240.75 and the high-end device will use Google Chrome version 86.0.4240.110.

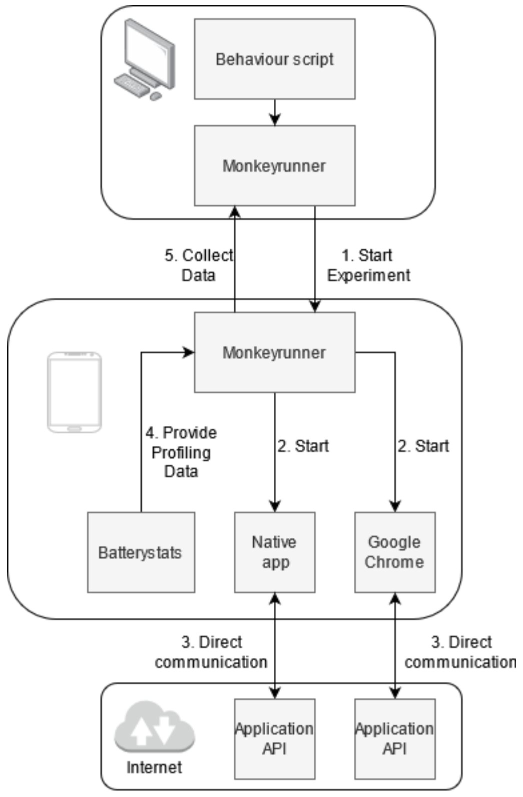

Figure 2: Experimental setup

All experiments will be executed on both a high-end Android device and a low-end Android device. The technical specifications are listed in table 4. The controlling machine is running on a Ubuntu 20.04 virtual machine (guest), which runs on a windows 10 host.

For the collection of data we will be exporting the results in appropriate format to R (v4.0.2) and plot data graphically and apply statistical analyses in R Studio.

All experiments will be run in the same network environment. The Java version used is AdoptOpenJDK8, the Python version 3.7.0.

5.1 Preparation

Each device is initialized with a clean install of the latest supported Android and set to not have any background activity for the duration of the experiment. The brightness is turned down to the

[12] https://github.com/bommels/android-runner/releases/tag/gz-exp-1

Device	**OnePlus 7 Pro**	**Samsung Galaxy J7 Duo**
Tier	High-end	Low-end
Released	May 2019	April 2018
CPU	Octa-core (1x2.84 GHz Kryo 485 & 3x2.42 GHz Kryo 485 & 4x1.78 GHz Kryo 485)	Octa-core (2x2.2 GHz Cortex-A73 & 6x1.6 GHz Cortex-A53)
OS	Android 9.0 OxygenOS 10.0.5	Android 8.0.0
RAM	8 GB	4 GB
Cores	8	8

Table 4: Specifications of Android devices.

lowest setting the phone supports for battery saving and since the apps will be driven by monkeyrunner, no manual input from the user is needed. Any setting that's irrelevant to the experiment will be turned off or disabled for as long as it's authorized to do so. Furthermore, push notifications and automatic OS updates are disabled and the working memory is cleared with no apps/services running. Prior to each trial each device will have 100% battery and will be charging. Lastly, each device is connected to the same Wi-Fi network and has roughly the same distance to the network router.

The experiment will run in four phases, namely the four combinations of app type and device tier depicted in Table 3. First the high-end device will run two experiments with native and web app types. Next, the low-end device will run also the web and native app types. Each app type and device tier combination has fifteen trials, where the order will be decided randomly a priori by the framework.

Before every trial, the device is reset to its default state, namely both the native app cache and Google Chrome's cache will be cleared. Also, after each trial a new baseline of the device's power consumption is determined and the batterystats' log is cleared. The devices will be connected through USB with the controlling device. The apps will be automatically pre-installed on the device, directly by installing the from APKPure[13] downloaded APK or xAPK file on the device.

5.2 Execution

In order to perform a consistent experiment for every app, the script will execute a series of commands that form a *usage scenario* which is looped for a duration of one minute followed with a duration of two minutes of no commands and tracing. Deka et al. has collected thousands of user interaction traces and analyzed the most common usage scenarios by end users for each app category[19]. A *user interaction trace* captures the views visited and interactions performed by the end-user. The *usage scenario* for each app from Table 2 will be designed based on the most common interactions performed by end-users from the analyzed results of [19]. The monkeyrunner scripts, dataset and results can be found in the GitHub repository mentioned in Footnote 9. The markdown file REPLICATION.md[14] contains an overview of the replication package and describes each folder and file in detail.

The usage scenarios for each app is detailed in the list below:

(1) **AliExpress:** a search for a certain product will be performed. It will scroll through he product overview page.
(2) **Deliveroo:** a search for a zipcode will be performed it will then scroll through the overview.
(3) **Reddit:** the hot feed is selected, then it scrolls through hot, then goes back to the top of the feed and switches to popular, where it repeats the same scroll action.
(4) **The Weather Channel:** a search for Amsterdam will be performed. Lastly, a scroll action on the page will be performed.
(5) **Tripadvisor:** a search for hotels in Amsterdam will be executed, where a scroll action on the hotels view will be performed.
(6) **Youtube:** the script first scrolls back and forward through the Home feed. Then it selects the Trending feed and opens the first video, which it will wait at for five seconds. Lastly it navigates back to the Home feed.
(7) **Zara:** a similar execution as for AliExpress will be performed. Searching for a pair of shoes and scrolling through the search results. Opening a product and going back to the main overview.
(8) **9GAG:** the script will do a similar scenario as for Reddit. It will scroll down and up, then changing between Hot and Trending feed.

6 RESULTS

The descriptive statistics of the obtained measured dataset and the results of the hypothesis tests are presented in Section 6.1 and in Section 6.2 respectively. Subsequently, in the sections thereafter the research questions are answered in more detail using the results of the hypothesis testing.

6.1 Descriptive Statistics

device	min	max	median	mean	SD	CV	IQR
both	90.5	778	184	246	147	0.596	148
high	90.5	279	152	153	35.2	0.231	47.9
low	113	778	293	339	156	0.462	216

Table 5: Summarization of energy consumption (in Joules) - *SD = Standard Deviation, CV = Coefficient of Variation, IQR = Interquartile range*

The boxplot in Figure 3 shows the distribution of the full dataset. As can be observed, there are many outliers. Furthermore the distribution is heavily skewed towards the right tail.

The non-normality can be explained due to the fact there is a large difference in total energy consumption between the low-end and high-end mobile device, as indicated in Figure 4. Therefore we decide to follow the rest of the analysis with device type as

[13]https://apkpure.com

[14]https://github.com/bommels/android-runner/blob/master/REPLICATION.md

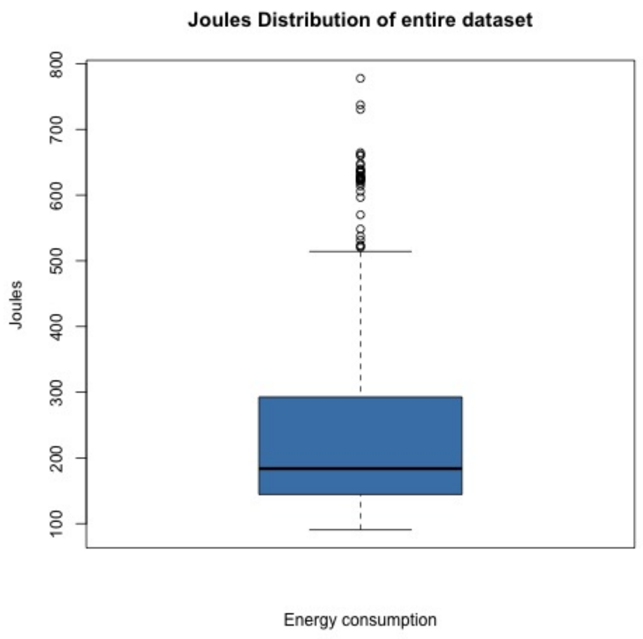

Figure 3: Measured energy consumption

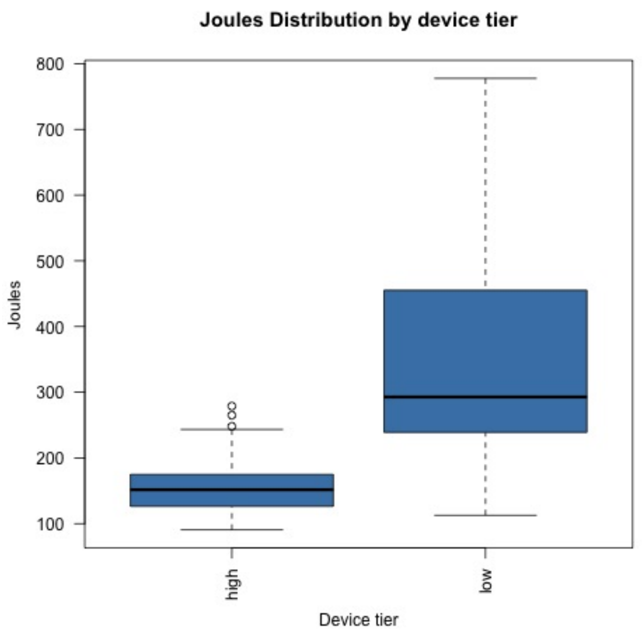

Figure 4: Measured energy consumption per device type

a *blocking factor*. However, this results in dropping **RQ2**, as the interaction between app type and device tier can no longer be assessed. This means **RQ1** will be independently answered in two fold, namely for the low-tier device block (**RQ1.1**) and the high-tier device block (**RQ1.2**) as described in Section 4.5. Additionally, Table 5 shows the summary statistics per device. It's evident that the standard deviation of the high-end device is much less than the low-end device.

In order to test the hypotheses of **RQ1.1** and **RQ1.2** with a *paired t-test*, the assumptions have to be verified. Namely:

- the subjects are randomly selected from the population.
- there are no extreme outliers in the differences between the treatment pairs.
- the differences between the pairs stem approximately from a normal distribution.

For both device blocks, the apps were selected randomly based on the criteria listed in Section 4.1. The verification of the second and third assumption will be discussed in the two following subsections.

6.1.1 *Low-end device block (RQ1.1).*

The QQ-plot in Figure 5 of the low-end device block depicts that the normality of the differences between the treatments is doubtful. Here, the Shapiro-Wilk test will be utilized to confirm its non-normality. Indeed, this yields a p-value of: $5.81e{-3}$, therefore there is evidence that the differences of treatments do not stem from a normally distributed population. Investigating further, by transforming the data by taking the natural logarithm, square root and reciprocal separately. Following the result of each transformation, the tests show no evidence of the data stemming from a normal distribution when utilizing the Shapiro-Wilk test (p-values: natural logarithm (1.50e−3), square root (5.06e−3) and reciprocal (1.53e−4). Concluding, for the low-end device block the *Wilcoxon signed-rank test* will be utilized to test the null hypothesis of **RQ1.1**.

No extreme outliers have been observed by visual inspection, which can be observed in Figure 7.

6.1.2 *High-end device block (RQ1.2).*

The measurement data of the high-end device the QQ-plot in Figure 6 indicates that it stems from a normal distribution. Utilizing the Shapiro-Wilk test produces the following p-value: 0.153, verifying that the null-hypothesis is not rejected, which indicates the differences data stems from a normal distribution. Concluding, for the high-end device block the *Paired t-test* is utilized to test the null hypothesis of **RQ1.2**.

Similarly as the first block, no extreme outliers have been observed by visual inspection, which can be seen in Figure 8.

6.2 Hypothesis Testing

As **RQ1** has been split up in two new research questions, the amount of tests that are performed are two. Therefore, the p-values of both tests have to be adjusted with *Benjamini-Hochberg's correction*. The corrected p-values are reported in the following subsections.

6.2.1 *Low-end device block(RQ1.1).*

For the low-end device we showed previously in Section 6.1 that the data does not follow a normal distribution, therefore we utilize the Wilcoxon signed-rank test. Utilizing this test yields a significant p-value of 1.27e−10. Therefore the null-hypothesis of equal means ($\mu_{web} = \mu_{native}$) is rejected and the alternative hypothesis of non equal means ($\mu_{web} \neq \mu_{native}$) is accepted.

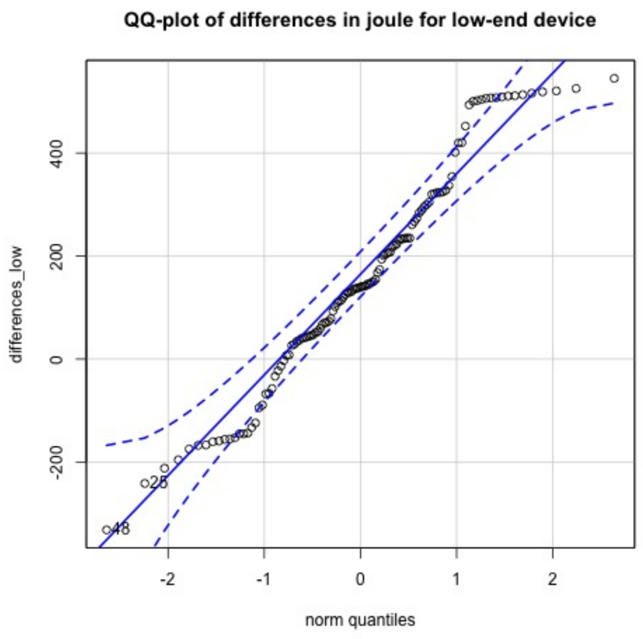

Figure 5: QQ plot for low-end device

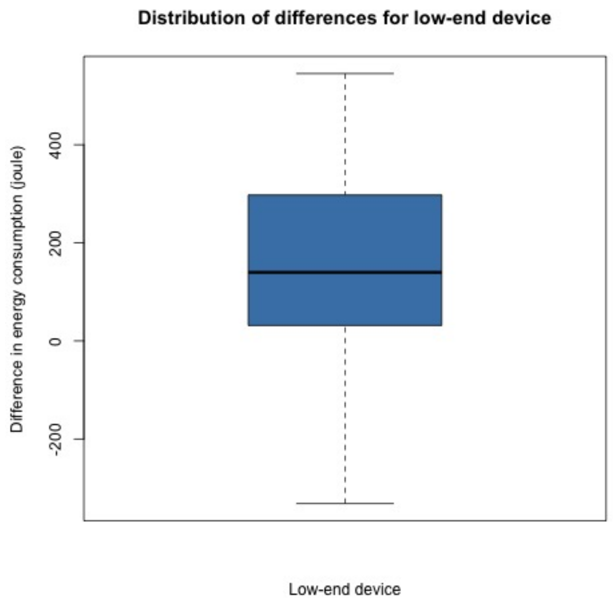

Figure 7: Low-end differences

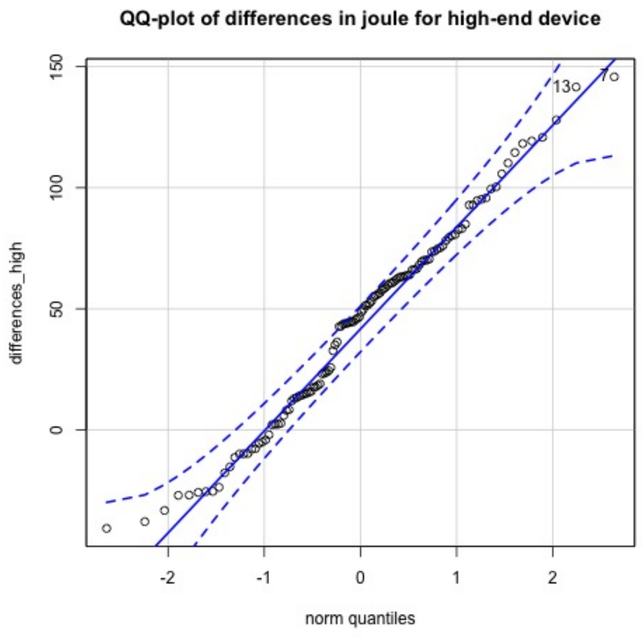

Figure 6: QQ plot for high-end device

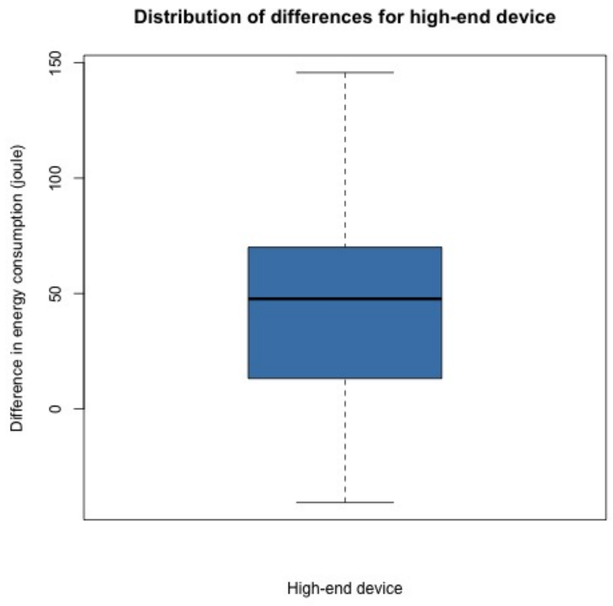

Figure 8: High-end differences

6.2.2 ***High-end device block (RQ1.2)***. In Section 6.1 we showed that the high-end data stems from a normal distribution. Therefore we utilize the paired t-test. Performing this test gives the following p-value: 1.41e−20. Concluding, we reject also in this case the null-hypothesis of equal means. Meaning, the null-hypothesis of equal means ($\mu_{web} = \mu_{native}$) is rejected and the alternative hypothesis of non equal means ($\mu_{web} \neq \mu_{native}$) is accepted.

6.3 Energy consumption difference between native- and web app (RQ1)

As the results of the previous sections indicate that overall (on both a low-end and high-end device) the usage of native or web app does have a statistical significance on the difference in energy consumption. However, it has not been assessed how large this phenomenon's effect size is expressed in the difference in joule. Table 6 details the energy consumption across each device and app type.

device	app type	min	max	median	mean	SD
high	native	90.51	195.07	126.88	131.00	25.58
high	web	122.50	278.88	164.10	174.55	29.67
low	native	112.80	624.61	246.50	261.12	117.48
low	web	230.81	777.64	417.65	416.15	152.30

Table 6: Energy consumption per treatment combination (device & app type)

6.3.1 ***Low-end device block(RQ1.1).*** The effect size for this block is determined by using *Cliff's delta* for non normally distributed data. This results in a score of 0.59, which means that the phenomenon's effect size is large. More specifically, it means that the difference in energy consumption is large. In terms of difference in joule, by consulting Table 6 the median (as it is less susceptible to outliers) can be used to estimate the average energy consumption difference. This estimation yields: 417.65 − 246.50 = 171.15 joule.

6.3.2 ***High-end device block (RQ1.2).*** The effect size of this block where the data stems from a normal distribution, *Cohen's D*'s method is utilized. This yields a score of 1.57, which also means that the effect size of the phenomenon is characterized as large. The same calculation as the low-end block is used to calculate the difference for the high-end device which yields: 164.10 − 126.88 = 37.22 joule.

It can be observed that there is a bigger difference for the low-end device compared with the high-end device.

6.4 App-specific Results

This section provides additional insights on previously discussed results by diving deeper on a per-app basis. As the experiment was designed as a full factorial design (2-factor, 2-treatment), each treatment combination (app type and device type) had been applied on each subject (app) repeated fifteen times for randomization of the measurement runs. The energy consumption of each factor-treatment combination is further analyzed.

Figure 9 depicts a clear trend of web apps consuming more energy than their native counterpart, except for the apps AliExpress and Reddit. However, the native app of AliExpress deviates quite heavily, suggesting a possible measurement error.

In Figure 10 the data seems to have more variance. The trend therefore is less clearly visible at first glance. However, when one looks more closely it can be seen that again Aliexpress and Reddit do not follow the trend based on the mean values, whereas the other apps closely follow the trend.

It can also be observed that on the low-end device there are more extreme outliers. For example The Weather native app has

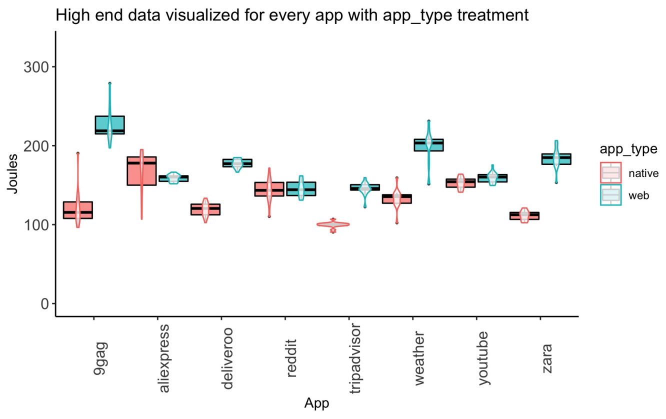

Figure 9: Dataset for high end device

a much lower energy consumption than its web counterpart, but an extreme outlier can be seen. The same holds for the Zara native app and Youtube web app. This can be attributed to the fact that there might have been some measurement errors, due to possible inconsistent executions of the *automatic user scenario* of these apps.

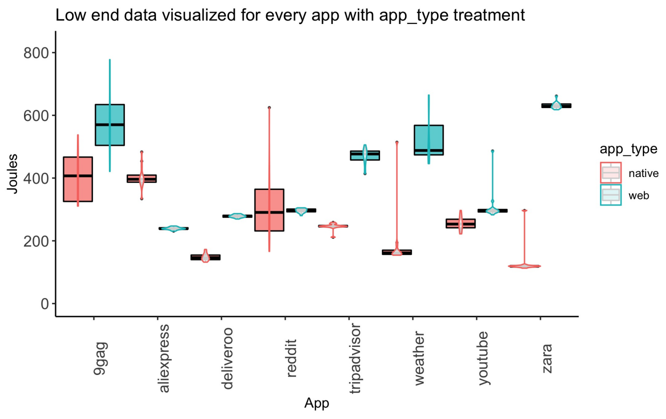

Figure 10: Dataset for low end device

7 DISCUSSION

We will elaborate more on the results of the tests, as seen in section 6, to answer our research questions. As we mentioned earlier, we dropped **RQ2** because of the significant difference in energy consumption between the low- and high-end devices. This resulted in **RQ1** being split up into two, one answering **RQ1** for low-end and the other high-end devices, as the device type became a blocking factor.

As mentioned in subsection 6.3, the results show that there is a significant difference in energy consumption for both the low- and high-end devices when comparing native apps versus their web counterparts. Web apps consume more energy than their native counterparts according to the results we got through different methods. Even though both the statistical tests for both the low-end and high-end block showed that native apps consume less than web apps, it was also shown that the effect size was characterized as large. Meaning the difference in Joules is meaningful in the context of battery usage. It was observed that there is a larger effect size for the low-end device compared with the high-end device.

These results allow for companies and developers to decide whether they should start their business through the creation of a web app or a native app. Companies need to make their apps as attractive as possible for the consumers to make them download their app. One of the many factors a user is interested in, is to get the most out of their battery. Our result shows that native apps have lower energy consumption compared to web apps on Android. This means that users can benefit a lot by downloading a native app, especially if the app requires a lot of resources such as computational power.

The experiment is mainly focused on black-box testing which means that we can not know what the architecture of the source code is and upon what technologies the selected applications are built. This means that we miss certain factors for the energy consumption of our selected apps. Furthermore, the specifications and energy usage of device(s) differ a lot. Companies often have new cutting-edge technology (software and hardware) that can reduce the energy consumption of their mobile phone products overall. A great example is Samsung's "power saving mode", which switches automatically to optimally use the remaining power of the device [20].

Therefore, to get a more in depth experiment, the energy consumption should be tested on all the popular devices. This introduces new independent variables for example: "energy efficiency technology", which can hold the values of these new cutting edge technologies. For example, the mentioned power saving mode of Samsung.

8 THREATS TO VALIDITY

In this section we describe the threats to validity of our experiment. The experiment has been analyzed based on four types of classification of threats to validity. Namely, internal, external, construct and conclusion validity [21]. These four different types of threats to validity of our experiment are described in the following subsections:

8.1 Internal Validity

Internal validity dives deeper into the reason why the results cannot be explained by external factors of the experiment.

8.1.1 ***History.*** As mentioned in 5.1, the approach of the experiment to obtain the energy consumption data was an iteration of 15 trials for each app type and device tier, with each trial lasting for 3 minutes. The results showed that the low tier devices introduced noise, which we defied from the result to give a more logically explained conclusion. To mitigate this threat, the tests were as close to each other as possible to ensure that a balanced data set was produced. This allows us to prevent sudden performance changes of the device.

8.1.2 ***Maturation.*** For our experiment we made sure that before each test execution, the cache of the device(s) were cleared, a new baseline of the device's power consumption is determined and the batterystats log is cleared as mentioned in section 5.1. Furthermore, we set a fixed time-frame between test executions to make sure all tests had the same test scenario. Otherwise maturation might have been of an influence, which means that the energy consumption over time may result in a higher consumption due to these certain conditions.

8.1.3 ***Selection.*** If an unequal number of tests have similar subject-related variables there is a threat to the internal validity of the experiment. To mitigate this threat we made sure we had an equal amount of tests.

8.1.4 ***Reliability of measures.*** An experiment can be influenced by many different aspects. Internet stability, screen brightness and background processes can all take a toll on the energy consumption of the test device. As discussed in 5.1 section, we've made sure to disable as many settings of the phone that has nothing to do with the experiment. The tests were conducted from the same distance to the router to make sure that moving around the router was not an issue.

8.2 External Validity

"External validity is the extent to which you can generalize the findings of a study to other measures, settings or groups. In other words, can you apply the findings of your study to a broader context?" [22]. In other words, it stands for the generalizability of the findings of the experiment.

8.2.1 ***Interaction of selection and treatment.*** As we discussed in section 4.1, we used the website AndroidRank to select our apps. We did this by selecting the eight most popular categories from AndroidRank and choosing their most popular app, as long as the company has not made any of the other apps. Choosing one app from the most popular categories and made by different companies eliminates the possibility for the apps to be representative of one another.

8.2.2 ***Interaction of setting and treatment.*** We have designed the experiment with knowledge we got from [19], which covers the most common usage of apps by the end users. This allows us to create scenarios based on real feedback from end users and fitting for the app. Furthermore, what our tests make more realistic is that we always use the Chrome browser of the device for each web-version test of our selected apps. At last we made sure that the device(s) used the internet connection of the connected computer. All these points reduces this specific threat to validity.

8.3 Construct Validity

Construct Validity is the relation between theory and observation. Construct validity is more the question to how are other things related to the subject. Whenever other traits related to the experiment are highly correlated with the experiment, it indicates a high construct validity.

8.3.1 ***Definition of constructs.*** By providing a well defined explanation of the constructs early in this project, we mitigated this threat. From the GQM-tree 1 we derived metrics, a formulated hypothesis and identified independent and dependent variables, that all are important for our experiment to be more reliable.

8.3.2 ***Mono-operation bias.*** As discussed under section 4.4, we have two independent variables with two treatments each. Our experiment consisted of 15 trials with 4 treatments in total, which helps in reducing the mono-operation bias from our measurements and results.

8.4 Conclusion Validity

Conclusion validity stands for statistical correctness and significance. This generally means that statistical methods are used to give an more accurate answer to our defined research questions rather than a logically explained answer.

8.4.1 ***Low statistical power.*** For our experiment we had $8\ apps \times 2\ app\ types \times 2\ device\ tiers \times 15\ trials = 450$ different tests. By having more tests, more data is obtained. Furthermore, we made sure to have the right amount of trials to avoid incorrectly rejecting the null hypothesis. This way better data analysis is performed to reduce the threat of low statistical power. Furthermore, we use certain transformations as explained in **??**, which also leads to a higher statistical power of the hypothesis tests.

8.4.2 ***Violated assumptions of statistical tests.*** To make sure we chose the correct test for our data, we checked the normality through a *Shapiro-Wilk test*, Q-Q plots and boxplots. Our data did not show a normal distribution, which meant we had to use the the Wilcoxon signed-rank test instead of the paired t-test for **RQ1.1**. We used the paired t-test for **RQ1.2** because the data showed a normal distribution.

8.4.3 ***Fishing and error rate.*** To mitigate this threat to validity we applied the Benjamini-Hochberg's correction technique as described in 6.2 to the retrieved p-values to obtain statistical correctness of our experiment.

9 CONCLUSION

This report studied the energy consumption of native Android apps versus web apps on Google Chrome, which is also done on a mobile device. Our results have shown, with the use of statistical data analysis, that web apps consume more energy than native apps, for both low- and high-end devices.

To answer our research questions, which are defined in 3.2. We came to the conclusion that web-apps consume more energy and sometimes this is even double the energy consumption. This dramatic difference shows on both devices. Because of a large difference in total energy consumption between the low- and high-end devices, we found that it was not possible to answer the second research question.

To extent this research, more in depth research can be done. For example by considering app code architecture, energy efficiency technologies, app patterns and so on. By adding those independent variables, there might be more data that can conclude our **RQ2**.

REFERENCES

[1] Comscore, "Comscore's 2019 "global state of mobile report"," https://www.amic.media/media/files/file_352_2183.pdf, Comscore white paper, Tech. Rep., 2019.
[2] M. Warcholinski, "App vs website – which to develop first?" https://brainhub.eu/blog/app-vs-website-which-to-develop-first/, 2020.
[3] K. Purdy, "Twitter and Facebook are killing your phone battery," https://web.archive.org/web/20181223193427/http://itworld.com/article/2833090/mobile/twitter-and-facebook-are-killing-your-phone-battery.html, 2013, [Online; accessed 11-September-2020].
[4] Z. Epstein, "Facebook is probably killing your iPhone's battery – here's how to fix it," https://bgr.com/2014/04/08/how-to-improve-iphone-battery-life-facebook/, 2014, [Online; accessed 11-September-2020].
[5] J. Evans, "Why is Facebook killing iPhone battery life?" https://www.computerworld.com/article/2476124/why-is-facebook-killing-iphone-battery-life-.html, 2014, [Online; accessed 11-September-2020].
[6] D. Li, S. Hao, J. Gui, and W. G. Halfond, "An empirical study of the energy consumption of android applications," in *2014 IEEE International Conference on Software Maintenance and Evolution.* IEEE, 2014, pp. 121–130.
[7] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic, "Assessing the impact of service workers on the energy efficiency of progressive web apps," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft).* IEEE, 2017, pp. 35–45.
[8] G. Inc, "monkeyrunner," https://developer.android.com/studio/test/monkeyrunner, 2020.
[9] W. Oliveira, W. Torres, F. Castor, and B. H. Ximenes, "Native or web? a preliminary study on the energy consumption of android development models," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 589–593.
[10] Y. Ma, X. Liu, Y. Liu, Y. Liu, and G. Huang, "A tale of two fashions: An empirical study on the performance of native apps and web apps on android," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 990–1003, 2017.
[11] D. Li and W. G. Halfond, "Optimizing energy of http requests in android applications," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, 2015, pp. 25–28.
[12] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.
[13] J. Poushter *et al.*, "Smartphone ownership and internet usage continues to climb in emerging economies," *Pew research center*, vol. 22, no. 1, pp. 1–44, 2016.
[14] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Software-based energy profiling of android apps: Simple, efficient and reliable?" in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER).* IEEE, 2017, pp. 103–114.
[15] S. Hao, D. Li, W. G. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *2013 35th international conference on software engineering (ICSE).* IEEE, 2013, pp. 92–101.
[16] G. V. Glass, P. D. Peckham, and J. R. Sanders, "Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance," *Review of educational research*, vol. 42, no. 3, pp. 237–288, 1972.
[17] M. R. Harwell, E. N. Rubinstein, W. S. Hayes, and C. C. Olds, "Summarizing monte carlo results in methodological research: The one-and two-factor fixed effects anova cases," *Journal of educational statistics*, vol. 17, no. 4, pp. 315–339, 1992.
[18] L. M. Lix, J. C. Keselman, and H. Keselman, "Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance f test," *Review of educational research*, vol. 66, no. 4, pp. 579–619, 1996.
[19] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar, "Rico: A mobile app dataset for building data-driven design applications," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, pp. 845–854.
[20] G. Joling, "Use power saving modes on your galaxy phone," https://www.samsung.com/us/support/answer/ANS00079037/, 2020.
[21] R. Hyman, "Quasi-experimentation: Design and analysis issues for field settings (book)," *Journal of Personality Assessment*, vol. 46, no. 1, pp. 96–97, 1982.
[22] P. Bhandari, "Understanding internal validity," https://www.scribbr.com/methodology/internal-validity/, 2020.