applied sciences

MDPI

Article

# Towards Energy-Efficient Mobile Ad Optimization: An App Developer Perspective

**Ahmad Raza Hameed [1], Saif ul Islam [2,*], Ahmad Almogren [3,*], Hasan Ali Khattak [4], Ikram Ud Din [5] and Abdullah Bin Gani [6,7]**

1 Department of Computer Science, National University of Computer and Emerging Sciences, FAST, Islamabad 44000, Pakistan; i171025@nu.edu.pk
2 Department of Computer Science, KICSIT, Institute of Space Technology, Islamabad 45000, Pakistan
3 Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia
4 School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), H12 Islamabad 45000, Pakistan; hasan.alikhattak@seecs.edu.pk
5 Department of Information Technology, The University of Haripur, Haripur 22620, Pakistan; ikramuddin205@yahoo.com
6 Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia; abdullahgani@ums.edu.my
7 Faculty of Computing and Informatics, University Malaysia Sabah, Labuan 88400, Malaysia
* Correspondence: saiflu2004@gmail.com (S.u.I.); ahalmogren@ksu.edu.sa (A.A.)

Received: 5 September 2020; Accepted: 28 September 2020; Published: 1 October 2020


**Featured Application: The proposed approach can be efficiently used to enhance the overall user experience while using mobile phone applications for consuming content.**

**Abstract:** Advertising over smart devices is one of the growing trends in the information technology domain. Most of the Android application (app) developers generate revenue through the use of ads, but on the other hand, the end users get the free app. However, the excessive number of ads infers hidden costs with respect to energy consumption, network utilization, and user comfort. These factors affect the app rating and reviews. Consequently, developers require a technique to balance app performance through optimized mobile ad usage. Therefore, in this paper, we extend an existing work and propose an energy-efficient method that uses gamma correction to reduce the hidden costs of a mobile app. In this approach, gamma correction efficiently balances the app performance by minimizing the size of ads. The size of the mobile ad is reduced by adjusting its pixels, reducing background color, and illuminating the content of the ad. After several experiments, it is deduced that our proposed approach efficiently saves mobile battery and developers can apply this approach to improve app rank and feedback.

**Keywords:** Android application; energy-efficient; mobile ads; gamma correction

## 1. Introduction

In the modern era, the smartphone has a considerable impact on communications, studying, infotainment, finance, and various other factors of human life. According to Gartner statistics, the number of mobile phone users has been increased from 12% to 116% from 2000 to 2018—and is projected to reach 453.19 billion in 2022 [1,2]. In order to make the usability of mobile gadgets better, various solutions have been proposed. However, due to its user-friendly interface and cost-effectiveness, Android is considered one of the most popular platforms in the smart world. Therefore, with the increase in Android usability, the range of Android applications is growing

exponentially with the passage of time [3,4]. Therefore, Android developers focus on app development to get huge revenue. In this context, advertisements through smartphone apps have gained a great attention. A recent analysis shows that more than $30 billion were spent on marketing through smartphones in 2014. Furthermore, the analysis predicts the amount of mobile advertisements even exceed TV advertisement [5]. Based upon this work, it was felt that overall optimization is required for enhancing the overall experience.

Typically, the Android developer gets ads from Google Ads Services and uses the mobile app to generate revenue; on the other hand, end users get the free app. It is a "win–win" situation for developers and users. Presently, a lot of Android developers develop an app without considering energy efficiency. Therefore, smart devices with limited resources of battery expend energy rapidly. Because of the quick power dissipation of mobile, end users under-rate an app, meaning that the developer needs to redesign an app, which is costly in terms of time and money [6].

Recently, a lot of research works have been conducted on energy-efficient techniques within software, application, and hardware level. More specifically, researchers have focused on application level because the developer is unable to buy and operate specialized hardware while designing a mobile app. On the other hand, a recent survey of the mobile ecosystem shows that the perception of free apps is misleading. In fact, almost half of the apps on the Google Play Store contain an excessive amount of ads [7,8]. Consequently, mobile ads contain hidden costs both for an end user and developer. The hidden ad cost affects end users as they slow down the response (consume an excessive amount of CPU memory), increase network usage, and increase energy cost (an excessive amount of battery usage). For the developer, the hidden costs may result in bad feedback [9]. Hence, the developer wants an approach to estimate hidden ad energy cost on a mobile app. Therefore, Gui et al. [5] have proposed an approach for the estimation of hidden power cost of ads at the application level of a smartphone. The estimation approach works in two phases: the first one is before the app implementation phase, where static modeling is applied to static ad configuration to generate energy consumption of mobile ads. The second one is after app implementation, where a dynamic model is applied on an implemented app to compute the energy consumption of ads. However, this work only estimates the hidden ad energy costs. This motivates us to propose an approach that reduces the hidden energy cost of ads over mobile app.

In the proposed work, gamma correction is applied over mobile ads, which reduces the hidden energy cost in terms of power dissipation and network usage. Furthermore, the main contribution of the work is summarized below:

- Gamma correction reduces the size of the mobile ad by adjusting pixels and reducing background color
- After size reduction, there is illumination of the content of the mobile ad

The experiment validates that the proposed approach efficiently reduces power dissipation of the mobile app caused by mobile ads; therefore the developer applies this approach to improve app rating and feedback.

The remainder of this paper is structured in such a way that Section 2 presents the related work. In Section 3, we discuss the problem statement. Section 4 elaborates proposed work. The evaluation and experimentation is presented in Section 5. Finally, Section 6 concludes the paper.

## 2. Related Work

There are several research contributions to developing a technique that efficiently saves power consumption of smart devices. However, the hidden energy cost of mobile ads over apps is overlooked. Therefore, in this context, Gui et al. [5] use a statistical method to compute the energy utilization related to the ads. This technique operates in two ways: First, they built a statistical model which provides the assessment. This model assumes the static values of ads that are SIZE, TYPE, and RRATE before the implementation of an app, where the SIZE is the size of the ad, TYPE is the behavior of the

app either text or video and RRATE is the refresh time of the ad content. Second, for more precise energy measurement, they introduce a run-time energy measurement technique. This technique works after app implementation. This technique captures key run-time metrics such as system energy model, network energy model, and display energy model. This technique provides information related to the power consumption of an app. Therefore, this proposed work predicts 31% energy consumption before the implementation and 14% after the app development.

Performance-based energy-efficient guidelines for Android mobiles are proposed in [8]. In this work, the authors provide the best set of practices to develop an energy-efficient app. This practice includes static analysis of the mobile app. The static analysis includes allocating an object upfront, efficient wake call, recycles, reduce over layout, useless parents, using fewer resources, reducing view call and overdrawing. Through this analysis, the developer reduces a considerable amount of mobile app energy consumption. Furthermore, energy modeling provides the source of energy dissipation; therefore, developers correct a particular part of the code to achieve energy savings. However, this static analysis is unable to reduce the hidden energy cost by mobile ads.

EcoDroid, an energy-based ranking approach, is proposed in [9]. In this work, Jabbarvand et al. efficiently calculate the energy consumption of mobile apps by dynamic and static analysis. In dynamic analysis, a test case is generated by interaction with the application, and converts it to path information. This information is combined with an analyzer for the estimation of consumed power of the dynamic path in the app. However, static analysis extracts the app call graph which contains the different possible invocation sequences of Android application. Through this energy consumption information, EcoDroid ranks the same category applications according to their energy consumption.

Hao et al. [10] propose a lightweight fine-grained power estimation *eLens* approach, which efficiently calculates the power consumed in smart devices at the software level. In order to estimate energy dissipation, *eLens* combines two energy estimation processes, program analysis and per-instruction energy modeling. In this approach, the workload is generated to find the path of the user action, and is incorporated with the energy model to calculate per-instruction energy consumption. Furthermore, *eLens* uses energy annotation to display energy consumption per-instruction graphically; through this, the developer efficiently finds energy consumption of the app and reduces it. However, this approach enables the reduction of the hidden cost of an app such as mobile ads. This hidden cost affects the mobile app in terms of energy dissipation and user rating.

In [11], Hao et al. propose a programmable user interface $(UI)$ automation (PUMA) for mobile applications. PUMA is a programmable framework which efficiently separates the exploring logic of app pages from analyzing the logic of the app. The authors implemented PUMA to perform dynamic analysis of smartphone apps using event handler Monkey (UI automation)—to monitor security, energy consumption, performance, and the correctness. Through this, the developer modifies apps at the run-time and enables the validation of app activities from the security breach. However, the behavior of hidden mobile app activities is unseen, which affects mobile devices in terms of energy consumption and security.

Corral et al. [12] proposes a kernel customization approach for reducing the power consumption in a smartphone application. In this approach, authors customize the kernel by optimizing CPU frequency scale, input–output (I/O) scheduling, under-lacking/under-voting and timer coalescing to adjust the power and workings of an app. The authors perform several tests on the customize kernel. Therefore, the modified kernel efficiently reduces the power consumption of the app.

In [13], the authors studied the optimal service allocation for a group of mobile applications in mobile cloud computing. They proposed a novel framework named location–time workflows (LTW) that is used to model the mobile applications for handling the service allocation during mobility. Furthermore, the framework optimally partitioned the workflow over mobile applications in 2-tier architecture based on the utility metrics, energy consumption, cost of the services, and delay of the mobile applications. The proposed system is evaluated using varying mobility models include Random

Waypoint and Manhattan models. It achieves 20% less mobile energy consumption and a reduced (30%) network delay.

Rong et al. [14], studied wireless sensor deployment and monitoring problems. They also discussed infrastructures and technologies to support the use of sensors in the smart city. For efficient network deployment configuration, they investigate different aspects including coverage, lifetime, and connectivity. Similarly, in the case of monitoring (mobile and static sensors), they also analyze sensing time, location, devices, and power consumption. Finally, the authors identify some research opportunities and directions to further explore sensor deployment and monitoring.

The authors in [15] proposed a knowledge-aware proactive node selection (KPNS) framework for an IoT environment. In KPNS, the selection of proactive nodes is based on their predicted preceding position. Furthermore, the KPNS system monitors the quality and energy efficiency of nodes. It also reduces the hot spot regions by efficiently utilizing the power of nodes.

Considering energy wastage and ignorance of processing requests by the network, the authors proposed an EAR-ADS algorithm (including energy-aware routing and an adaptive delayed shutdown mechanism) in [16]. This algorithm deployed dynamic service function chains (SFC), which means offline and on–off nodes in the network. Furthermore, this technique saves the power consumption of servers. Due to the adaptive delay shutdown mechanism, the energy wastage is further reduced.

Table 1 presents a comparative analysis of related work. As we see, existing approaches focus either on estimating the hidden ad energy consumption on the mobile app or to reducing the energy consumption of app in different perspectives such as code optimization, kernel optimization, etc. However, the hidden cost of ads over mobile apps is not addressed. Therefore, we propose an energy-efficient mechanism to reduce the hidden cost of ads on the mobile app.

**Table 1.** Comparison of state-of-art work.

| Protocol Name | Features | Achievements | Deficiencies |
| --- | --- | --- | --- |
| Static approach for measuring ad-related energy cost [5] | Static modeling and run-time dynamic modeling | Estimate energy consumption 31% before implementation and 14% after implementation. | Unable to reduce hidden energy cost |
| The hidden cost of mobile ads for software developers [7] | Static mobile ads model | Estimate the energy consumption of ads before app implementation phase. | Focus on identity of hidden energy cost of ads |
| Performance-based energy-efficient [8] | Static analysis include object upfront, efficient wake call, recycles, reduce over layout, useless parents, useless resources, reduce view call and overdrawing | Minimize the energy consumption of app. | Still ad hidden energy cost |
| EcoDroid: an energy-based ranking approach [9] | Dynamic and static analysis | Dynamic analysis estimates the energy consumption of ads by interaction path analyzer while static analysis uses history of mobile data. | Absence of mechanism to reduce hidden ad cost |
| *eLens* app energy estimation [10] | power modeling | power consumption estimation. | Unable to mitigate ads hidden cost |

**Table 1.** *Cont.*

| Protocol Name | Features | Achievements | Deficiencies |
| --- | --- | --- | --- |
| PUMA [11] | Separate the exploring logic of app pages from analyzing logic of app | It verifies security breach, energy consumption and correctness of activities in response. | Absence of hidden cost |
| Software-based kernel customization approach [12] | Customize the kernel and balance between energy and performance | This phenomena reduces the energy consumption of app running on it. | Hidden energy cost |
| An optimal service allocation approach for mobile applications. [13] | A location–time workflow (LTW) model for mobile apps | Services are offloaded during mobility and the workload is partitioned to minimize the energy utilization of apps. | Hidden ads energy cost |
| A survey on wireless sensors for smart city environment [14] | Deployment strategies and monitoring techniques | Analyze scheduling techniques to reduce energy consumption of network and mobile devices. | Illustrate ads energy consumption |
| KPNS [15] | The law of target movement for prediction | Maintain balanced workload to reduce the energy cost of mobile devices. | Performance degradation of mobile devices |

## 3. Problem Statement

For better understanding of the problem, we illustrate problem statement in the design science approach (DSA) way.

### 3.1. Motivation

In recent years, mobile devices become an important part of our daily life. The emergence of mobile devices such as smartphones, tablets, etc. provides us with useful features. In addition, it helps us to acquire information in no time. Android is considered as a ruling platforms in the mobile industry. The number of Android applications has increased, which makes them versatile and necessary for us [3]. Moreover, ads are an integral part of mobile applications. Typically, ads are used a source of income, while the subscribers get the free app in return. Due to the use of inefficient mobile ads, there is an energy depletion problem which directly reduces the performance and battery life of the smartphone. Consequently, the end user underrates the developer app, which directly affects a lot of the information technology (IT) community. Therefore, recent research has focused on this problem to compute the energy utilization of ads in apps.

### 3.2. Problem

In [5], the authors calculate the energy utilization of mobile ads in two phases, as portrayed in Figure 1—pre- and post-implementation. In the pre-implementation phase, static modeling is applied on the static configuration of the ads which estimates its energy consumption on a mobile app. Due to this, the developer gets early feedback before application implementation. Thus, the developer designs an application according to the feedback solution. In the post-implementation phase, the developer provides a workload as an input to calculate the power consumed through ads. To find the precise information of ads, the authors make a duplicate copy of the app and remove an ad from it then apply the workload on both apps with or without ads. The run-time energy calculation model includes system, network and display model, which works parallel to the workload to find precise mobile ad energy consumption after application implementation. However, existing work only focuses on

estimating mobile ads energy consumption. This motivates us to propose a solution for the app developer which reduces the size of ads by applying gamma correction on mobile ads.

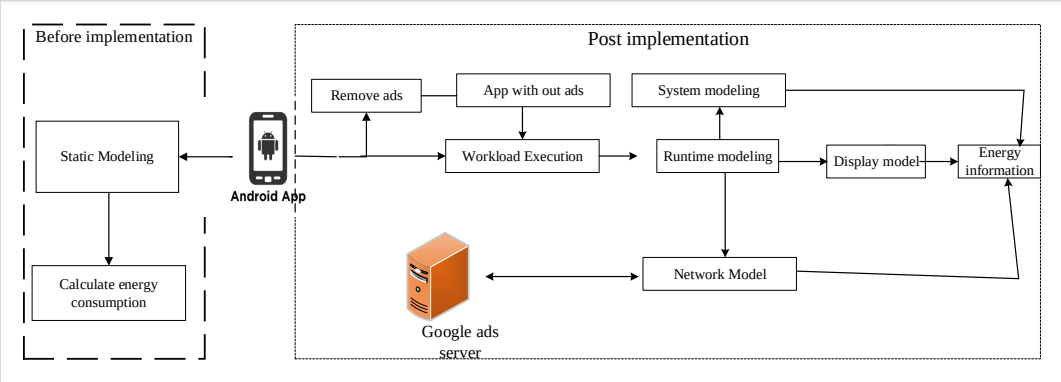


**Figure 1.** Existing work.

### *3.3. Evaluation*

For fair comparison, we used state of the art tools and devices for experimentation [5]. Furthermore, cam scan, nature photo editor, blind traveler app, and karvan card applications are used for the proposed work evaluation.

- **RERAN tool**: Recording the manual generated workload. Through this we use the same workload at any time.
- **Android profiler**: Recording energy consumption of applications.
- **Trapen profiler**: Recording energy consumption of applications.
- **Android studio**: For Code.
- **Device for experiment**: Q mobile, Samsung core prime and Huawei Mate 10 lite.
- **Matlab tool**: Pictorial representation of findings.
- **Android applications for experimentation**: Cam Scan, Nature Photo Editor, Blind traveler app, and Karvan Card.

### *3.4. Hypothesis*

The proposed mechanism is expected to reduce mobile energy consumption by 50% by optimizing ads over the mobile application. Furthermore, our work has an impact on a considerable part of the Android community, in particular app developers who optimize applications to improve rank and feedback.

## 4. Overview of the Proposed Work

The aim of the proposed work is to reduce energy use along with estimating the energy cost of ads all around the life cycle of app development. The proposed study uses the same techniques as presented in [5], which estimates the power cost of ads before and after application development as shown in Figure 1. In order to balance battery depletion, we applied gamma correction on mobile ads. Gamma correction reduces the size of mobile ads by adjusting the pixels and reduces the background color. For better visibility of mobile ads, we illuminate its content. The process is applied both before the implementation phase and after the app development phase.

The proposed work is shown in Figure 2, which operates in two ways:

- Before the implementation phase
- post-implementation phase

In the first phase, our approach takes the static model value as an input to gamma correction, which further reduces the quality and size of the ad over the mobile app. This process provides

early feedback to the developer to optimize mobile ads before app development. Similarly, in the post-implementation phase, the proposed work takes the display model as an input to gamma correction, which customizes mobile ads to make it energy-efficient for an app. We validate our approach by comparing the modified ad application with other existing application.

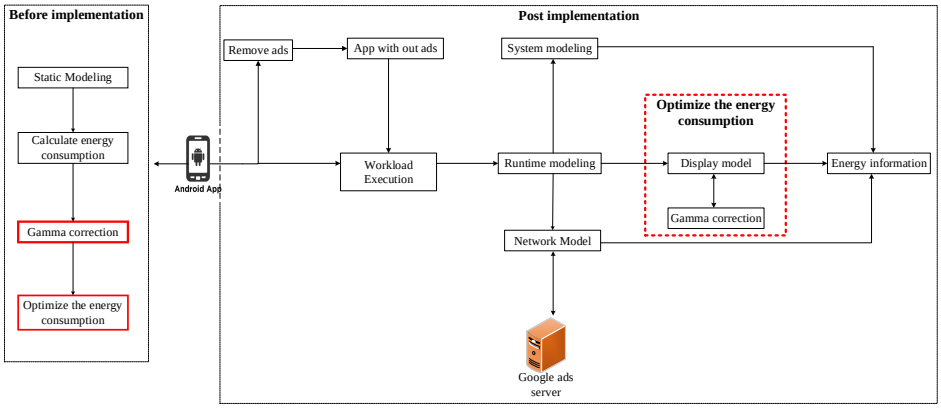


**Figure 2.** Proposed work.

### *4.1. Static Model*

To acquire the energy consumption of mobile ads in app, existing work [5] uses static modeling. The model provides information regarding the ads energy utilization before the development phase. Before app implementation, the developer includes some static variables of ads such as size $s$, type $t$, and refresh rate $r$. Therefore, these values of a variable are taken as an input to the static model result in high-level energy estimation of ads in the mobile app. In order to build efficient estimation through the static model, we conduct several numerical experiments. In these experiments, we considered static ad configuration, varying one configuration and keeping the others the same, then calculating the average energy consumption of mobile ads over application. In order to find accurate average energy consumption of ads, we repeat this numerical experiment several times. Thus, finally we conclude that energy consumption of ads is linear when $r$ when $s$ and $t$ are kept fixed. We formulate the relation as:

$$E = \beta - \alpha * (r - 30) \tag{1}$$

In the above Equation (1), $\alpha$ is a coefficient, $\beta$ presents the power consumption (average) and the lowest refresh rate $r$ is 30.

### *4.2. Dynamic Model*

The approach proposed in [5] provides developers with energy consumption information about mobile ads. The proposed approach takes the implemented apps and generated workload by the developer as input and then computes the energy cost of a mobile ad at run-time. For manual workload generation, the RERAN tool is used [5]. This tool keeps a record of user activities which are repeated later on. Furthermore, for clear differences between the app and mobile ad energy consumption, the proposed approach compares both the with-and without-ads application and computes the utilization of energy by the ads. This model considers the utilization of CPU at different frequencies, network usage, and screenshots with the time-stamp. The energy model is formulated in Equation (2):

$$\bar{E}_{total} = \bar{E}_{system} + \bar{E}_{network} + \bar{E}_{display} \tag{2}$$

#### 4.2.1. System Model

The system model computes the energy consumption of the CPU at a different frequencies. This model generates information regarding the power consumption of ads. This model finds a linear relationship between CPU time and frequency, which is calculated as:

$$\bar{E}_{system} = \sum_{f=1}^{n} \left( \bar{E}_f \times \bar{T}_f \right) + \left( \bar{E}_m \times \bar{M} \right) \tag{3}$$

In Equation (3), $f$ presents the CPU frequency and $n$ shows the total number of variations in CPU frequency. $\bar{E}_f$ is the energy usage while mobile ad running on CPU and $\bar{T}_f$ is the total time at which mobile ad running on the CPU. The mobile ad also use memory while running; therefore, this model computes memory energy usage and total memory usage respectively, $\bar{E}_m$ and $\bar{M}$.

#### 4.2.2. Network Model

The network model computes the energy utilization of mobile ads by considering the total number of bytes sent over the network. The model formulates a linear relationship in Equation (4):

$$\bar{E}_{network} = \bar{C}_n \times \bar{B}_{total-bytes} \tag{4}$$

where $C_n$ is the coefficient that shows the energy utilization per unit byte transferred over the network and $B_{total-bytes}$ is a total number of bytes that the ads have sent.

#### 4.2.3. Display Model

The display model estimates the energy consumption of ads by taking screenshots of mobile ads as an input when the workload is manually generated then efficiently estimates the energy utilization of mobile ads. The energy of mobile ads is computed for a specific interval of time as per Equation (5):

$$\bar{E}_{display} = \sum_{s=0}^{n} \left( \bar{P}_{Screen-shot}(s) \times \bar{T}_{screen-shot}(s) \right) \tag{5}$$

where $\bar{P}_{Screen-shots}(s)$ is the total power consumed by the ads at specific interval $\bar{T}_{screen-shots}(s)$, herewith $s$ is the number of screenshots.

$$\bar{P}_{Screen-shot}(s) = \sum_{K \epsilon s} \left( \bar{C}(R_k G_k B_k) \right) \tag{6}$$

Furthermore, the power consumption of each screenshot is the sum of the cost of pixel values as in Equation (6). The prior pixel values are found by RGB (red, green, blue) values of the screenshot.

$$\bar{C}(R_k G_k B_k) = rR + gG + bB + c \tag{7}$$

### *4.3. Gamma Correction*

Gamma correction is a nonlinear operation used to encode and decode luminance in images and videos. The digital camera captures an image; its intensity is not balanced as human perception [17]; therefore, gamma correction is applied to it. The purpose of this correction is to balance the pixels of the image according to human perception (eye intensity) by applying the gamma correction formula:

$$\bar{V}_c = \bar{A} \times \bar{V}_{uc}^{\gamma} \tag{8}$$

here $\bar{A}$ is the arbitrary constant, $\bar{V}_{uc}$ is the uncompressed image obtained from the Google Play Store and $\gamma$ is the gamma correction value. In order to compress and reduce the pixels of an image, $\gamma$ value is

considered to be less than 1 [18]. However, in mobile apps we apply gamma correction on mobile ads to reduce the hidden energy consumption of mobile devices. In order to apply gamma correction, first, the mobile ad converts the image into a bitmap image (it is a vector drawing in Android), then this image is passed to the gamma correction function. In this function, we set the threshold value of 250 and reduce the width and height of the image up to less than equal to the threshold value. After the image-reduction step, image decoding starts in which the compressed matrix applies to each image pixel. Through this process, illumination of pixels is adjusted, which reduces the size and quality of the image. The modified image is encoded and converted into a bitmap object which passes to the mobile app. Due to this process, a considerable amount of hidden energy is saved, which directly increases the battery lifetime of a mobile device.

## 5. Evaluation and Experiments

In this section, we describe and evaluate the experimentation of the proposed gamma correction algorithm as presented in Algorithm 1. In particular, we categorize the experiment to address the following research questions.

- **RQ1:** Can image-compression technique (gamma correction) reduce the energy consumption of the mobile app?
- **RQ2:** Does gamma correction efficiently increase battery lifetime and performance the of a mobile device?

**Algorithm 1:** Gamma correction on mobile ads.

```
Initially;
Image ← get-image(URL);
Bit-image ← Bitmap-covert(Image);
Gamma correction (Bit-image)
W ← Bit-image.Width;
H ← Bit-image.Height;
if W ≥ threshold value & H ≥ threshold value then
    SW ← W/half;
    SH ← H/half;
    Goto IF(W&H ≥ threshold − value)
D-image ← decode(Bit-image);
foreach Pixels ∈ Bit − image do
    C-Image ← Compress-matrix(Bit-image);
E-image ← encode(C-Image);
B-image ← Bitmap-covert(E-image)
return Gamma correction(B − image);
```

### 5.1. Experiment Setup

To examine the performance of the proposed approach, the experiments are performed on the Samsung Core Prime, Huawei Mate 10 lite and Q mobiles, as listed in Table 2. The Android application along with ad (considered as an image) builds on the Android studio. The gamma correction (image compression) technique is applied on the image to reduce the hidden energy cost of mobile ads in the application. For the sake of fare energy comparison, we use the same application with or without an image-compression technique. Furthermore, both applications run for 7 min on a mobile device and we compute the hidden energy cost of mobile ads using the Android profiler. Furthermore, this energy is also computed for verification by trapen energy profiler. For pictorial representation of result, Matlab is used. Therefore, calculation shows that the proposed mechanism is suitable for the app developer while developing an app in the presence of mobile ads.

**Table 2.** Tools and subjected apps for experiment.

| Name | Purpose |
|---|---|
| Android profiler | Recording energy consumption |
| Trapen profiler | Recording energy consumption |
| Android studio | For Code |
| Device for experiment | Q mobile, Samsung core prime, Huawei mate 10 lite |
| Matlab | Pictorial representation of findings |
| RERAN tool | Recording the manual generated workload |
| Android applications for experimentation | Cam Scan, Nature Photo Editor, Blind traveler app, Karvan Card |

### 5.2. *Rq1: Gamma Correction Reduces the Energy Consumption of the Mobile App*

Traditionally, gamma correction is used to manipulate the pixels of a digital camera image, due to which the human eye can view a fully illuminated image [18]. However, in this work, we use correction on mobile ads to reduce the size and adjust the pixels of the image to reduce the quality of mobile ads. Moreover, the quality is maintained up to the point where the content can read easily, hence, it is a "win–win" situation for the developer, end user, and third party Google Store. Figure 3 shows that the energy consumption of gamma-corrected applications is low compared to other non-corrected versions of the applications.

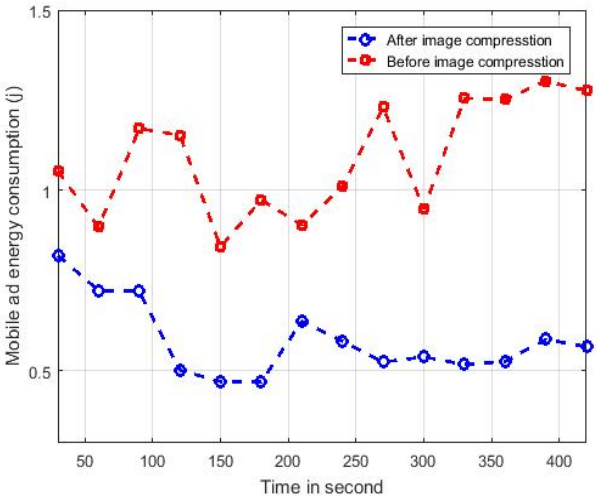


**Figure 3.** Mobile ad energy consumption.

### 5.3. *Rq2: Gamma Correction Efficiently Increases the Battery Lifetime and Performance of the Mobile Device*

According to existing research on the mobile ecosystem [3–7], we can say that the energy consumption of the mobile device has a nonlinear relationship with the lifetime of battery and performance of the mobile device. Similarly, Figure 4 illustrates, after gamma correction on mobile applications, that it consumed less mobile battery. Hence, the gamma correction increases the lifetime of mobile battery and the performance of the mobile device. Thus, our technique provides a way for a developer to reduce the hidden energy cost of the mobile application.

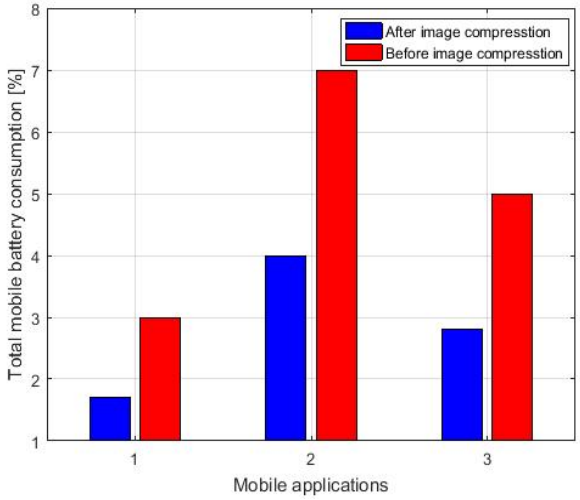


**Figure 4.** The impact of gamma correction on mobile applications.

## 6. Conclusions and Future Work

Developing an energy-efficient mobile application in the presence of mobile ads is a daunting task for the software developer. Unfortunately, much existing work focuses either on finding and estimation the energy consumption of a mobile app. Furthermore, other online resources focus on the performance of the mobile application. Thus, due to lack of guidance, the developer is unable to design an energy-efficient mobile application. Therefore, in this paper, we proposed a way to optimize mobile ads in applications. In this work, we apply gamma correction to a mobile application to reduce the size of mobile ads. Furthermore, in order to reduce the quality of mobile ads, gamma correction adjusts every pixel of the mobile ad. Due to this, our proposed work minimizes the hidden energy cost of mobile ads, which directly increases the lifetime and performance of the mobile app. The simulation result validates that our proposed work efficiently reduces the energy consumption of a mobile application. In the future, we plan to integrate other image-compression techniques with gamma to further optimize the energy depletion of mobile applications and extend our research for the iOS platform.

**Author Contributions:** Conceptualization, A.R.H., S.u.I. and H.A.K.; Formal analysis, I.U.D. and A.B.G.; Funding acquisition, A.A.; Investigation, A.R.H.; Methodology, A.R.H., S.u.I. and H.A.K.; Software, A.R.H., Project administration, S.u.I. and A.A.; Resources, S.u.I.; Supervision, S.u.I. and H.A.K.; Validation, I.U.D. and A.B.G.; Writing—Original draft, A.R.H., H.A.K. and S.u.I.; Writing—Review & editing, A.B.G., I.U.D. and A.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by King Saud University, Riyadh, Saudi Arabia, through Researchers Supporting Project number RSP-2020/184.

**Acknowledgments:** Authors acknowledge the support by COMSATS University Islamabad and National University of Computer and Emerging Sciences, FAST, Islamabad for providing means to conduct the study.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Capponi, A.; Fiandrino, C.; Kantarci, B.; Foschini, L.; Kliazovich, D.; Bouvry, P. A Survey on Mobile Crowdsensing Systems: Challenges, Solutions, and Opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2419–2465. [CrossRef]
2. Wang, X.; Athanasios, V.; Vasilakos, M.C.; Yunhao, L.; Ted, T.K. A survey of green mobile networks: Opportunities and challenges. *Mob. Netw. Appl.* **2012**, *17*, 4–20. [CrossRef]
3. Lee, S.; Go, M.; Ha, R.; Cha, H. Provisioning of energy consumption information for mobile ads. *Pervasive Mob. Comput.* **2019**, *53*, 49–61. [CrossRef]

4. Cai, H.; Gu, Y.; Vasilakos, A.V.; Xu, B.; Zhou, J. Model-driven development patterns for mobile services in cloud of things. *IEEE Trans. Cloud Comput.* **2016**, *6*, 771–784. [CrossRef]
5. Gui, J.; Li, D.; Wan, M.; Halfond, W.G. Lightweight measurement and estimation of mobile ad energy consumption. In Proceedings of the 2016 IEEE/ACM 5th International Workshop on Green and Sustainable Software (GREENS), Austin, TX, USA, 16 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–7.
6. Corral, L.; Georgiev, A.B.; Sillitti, A.; Succi, G. Can execution time describe accurately the energy consumption of mobile apps? an experiment in Android. In Proceedings of the 3rd International Workshop on Green and Sustainable Software, Hyderabad, India, 1 June 2014; ACM: New York, NY, USA, 2014; pp. 31–37.
7. Gui, J.; Mcilroy, S.; Nagappan, M.; Halfond, W.G. Truth in advertising: The hidden cost of mobile ads for software developers. In Proceedings of the 37th International Conference on Software Engineering, Florence, Italy, 16–24 May 2015; IEEE Press: Piscataway, NJ, USA, 2015; pp. 100–110.
8. Cruz, L.; Abreu, R. Performance-based guidelines for energy-efficient mobile applications. In Proceedings of the 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), IEEE, Buenos Aires, Argentina, 22–23 May 2017; pp. 46–57.
9. Behrouz, R.J.; Sadeghi, A.; Garcia, J.; Malek, S.; Ammann, P. Ecodroid: An approach for energy-based ranking of Android apps. In Proceedings of the Fourth International Workshop on Green and Sustainable Software, Florence, Italy, 18 May 2015; IEEE Press: Piscataway, NJ, USA, 2015; pp. 8–14.
10. Hao, S.; Li, D.; Halfond, W.G.; Govindan, R. Estimating mobile application energy consumption using program analysis. In Proceedings of the 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 18–26 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 92–101.
11. Hao, S.; Liu, B.; Nath, S.; Halfond, W.G.; Govindan, R. Puma: Programmable ui-automation for large-scale dynamic analysis of mobile apps. In Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, Bretton Woods, NH, USA, 16–19 June 2014; ACM: New York, NY, USA, 2014; pp. 204–217.
12. Corral, L.; Georgiev, A.B.; Janes, A.; Kofler, S. Energy-aware performance evaluation of Android custom kernels. In Proceedings of the 2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software (GREENS), Florence, Italy, 18 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–7.
13. Rahimi, M.R.; Venkatasubramanian, N.; Mehrotra, S.; Vasilakos, A.V. On optimal and fair service allocation in mobile cloud computing. *IEEE Trans. Cloud Comput.* **2015**, *6*, 815–828. [CrossRef]
14. Du, R.; Santi, P.; Xiao, M.; Vasilakos, A.V.; Fischione, C. The sensable city: A survey on the deployment and management for smart city monitoring. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1533–1560. [CrossRef]
15. Liu, X.; Zhao, S.; Liu, A.; Xiong, N.; Vasilakos, A.V. Knowledge-aware proactive nodes selection approach for energy management in Internet of Things. *Future Gener. Comput. Syst.* **2019**, *92*, 1142–1156. [CrossRef]
16. Sun, G.; Zhou, R.; Sun, J.; Yu, H.; Vasilakos, A.V. Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization. *IEEE Internet Things J.* **2020**, *7*, 6116–6131. [CrossRef]
17. Jobson, D.J.; Rahman, Z.U.; Woodell, G.A. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Trans. Image Process.* **1997**, *6*, 965–976. [CrossRef] [PubMed]
18. Gupta, B.; Tiwari, M. Minimum mean brightness error contrast enhancement of color images using adaptive gamma correction with color preserving framework. *Optik* **2016**, *127*, 1671–1676. [CrossRef]


**Sample Availability:** Data can be made available upon reasonable request to the corresponding author.