# Learning Battery Consumption of Mobile Devices

**Paul Eastham** EASTHAM@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

**Andrés Muñoz Medina** AMMEDINA@GOOGLE.COM
Google, 111 8th Ave, New York, NY 10011

**Ashish Sharma** ASHISHSHARMA@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

**Umar Syed** USYED@GOOGLE.COM
Google, 111 8th Ave, New York, NY 10011

**Sergei Vassilvitskii** SERGEIV@GOOGLE.COM
Google, 111 8th Ave, New York, NY 10011

**Felix Yu** FELIXYU@GOOGLE.COM
Google, 111 8th Ave, New York, NY 10011

## Abstract

We introduce a data-driven model to predict battery consumption of apps. The state-of-the-art models used to blame battery consumption on apps are based on micro-benchmark experiments. These experiments are done under controlled conditions where power measurements of each internal resource (CPU, Bluetooth, WiFi, ...) are readily available. We empirically verify that such models do not capture the power consumption behavior of mobile devices *in the wild* and propose instead to train a regression model using data collected from logs. We show that this learning approach is correct in the sense that under mild assumptions, we can recover the true battery discharge rate of each component. Finally, we present experimental results where we consistently outperform a model trained on micro-benchmarks.

## 1. Introduction

With more than one billion users worldwide, mobile devices have surpassed the popularity of personal computers. Social networking, media streaming, text messaging, and navigation are among the many ways that consumers use these devices. This versatility, however, has come at the expense of deteriorating battery life. To address this problem, mobile device manufactures have invested a lot of effort into improving battery capacity. However, battery improvements develop at a slower pace than the current market requires (Schlachter, 2013).

Another way to improve battery life is through software engineering, minimizing the amount of battery used by apps. Of course, before it can be minimized, an app's battery usage must first be measured. While measuring the battery consumption of a mobile device is typically straightforward, attributing that consumption to the individual apps running on the device is more challenging, and is commonly done in two steps:

1. Tracking how much the app uses each hardware component (such as Bluetooth, CPU, camera, etc).
2. Modeling how much power each hardware component drains when it is in use.

The modeling problem has received the attention of a substantial body of literature (see Section 1.1). However, nearly all previous work has a common caveat: Model parameters were tuned by exercising each hardware component separately in carefully controlled *micro-benchmark* experiments performed in a laboratory. Even in cases where real-world data was used to estimate model parameters, the learned models were evaluated in a laboratory setting. However, laboratory experiments are inadequate for estimating

*Proceedings of the 33$^{rd}$ International Conference on Machine Learning*, New York, NY, USA, 2016. JMLR: W&CP volume 48. 

or validating models that will ultimately be used to predict the battery consumption of apps used by actual consumers for the following reasons:

- Even in an experimental setting, it is hard to completely isolate a particular hardware component, as some components must always run in the background to allow the device to function.
- The discharge rates vary from device to device depending on their hardware and operating systems, and it is impractical to estimate parameters for all components on all devices and all OS versions..
- The battery consumption rates of many hardware components is not the same in a laboratory as during real-world operation. For instance, the radio will transmit at a higher power when it is farther from a tower (Ding et al., 2013). Accurately capturing all possible variations of the power profile of a component is difficult.
- By training each component model separately, one cannot capture the interactions between different components. Evidence of the importance of correlations between metrics is given in (Peltonen et al., 2015).

In view of these problems, we propose a data-driven method to jointly estimate the discharge rates of all hardware components. Our data consists of battery discharge reports from thousands of users over several days of usage. These reports contain information about the components used by each app during a particular period of time as well as the battery discharge, measured in milliampere-hours, over this period. Our model parameters are learned by performing linear regression on these observables. We provide theoretical evidence that the estimated parameters are accurate. We also empirically measure the accuracy of our models by using them to predict the battery usage of apps on specially-instrumented devices. To the best of our knowledge, ours is the first work to both estimate and evaluate battery consumption models on real-world data.

### 1.1. Related Work

A large body of literature has attempted to model the battery usage of mobile devices, with much of it focused on network activity. For example, Balasubramanian et al. (2009) model the power associated with network activity as a finite state machine. The authors also discuss the existence of high energy consumption states. These states not only consume power while being used but seem to keep draining power after data transfers have finished. Similarly, Rosen et al. (2015) show that several apps persist on their foreground network activity even after moving to the background. In fact, this activity persists sometimes for even a day, unnecessarily wasting data and battery. Ding et al. (2013) propose a way to incorporate signal strength into the network-battery model showing that a poor signal is associated with greater battery consumption. In all these papers, however, the model parameters are tuned and validated in microbenchmark experiments.

| Hardware Component | Model Type |
|---|---|
| CPU | Frequency + Utilization |
| GPU | Frequency + Utilization |
| Screen | Brightness Level |
| WiFi | FSM + Signal Strength |
| 3G/LTE | FSM + Signal Strength |
| WiFi Beacon | WiFi Status |
| Cellular Paging | Cellular Status |
| SOC Suspension | Constant |

*Table 1.* Summary of power model used by (Chen et al., 2015)

Chen et al. (2015) introduce a complex model that can be viewed as the sum of 8 different models, each one accounting for one of the components shown in Table 1. The authors deployed their model via an app to thousands of users to understand their power consumption behavior. Despite analyzing battery consumption "in the wild", the models themselves are estimated via laboratory experiments. On the other hand, Peltonen et al. (2015) collect millions of user reports and estimate the influence of each metric on battery discharge as a function of their mutual information. But their approach is evaluated in a laboratory setting.

## 2. Setup

We will learn a battery consumption model from anonymized reports $r = (\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R}$. Each report spans a contiguous time period (typically 1 day for phone and a few days for tablets) on a particular device. The report consists of a vector $\mathbf{x} = (x_1, \ldots, x_d)$ where each $x_i$ represents how much of a resource was consumed during the span of a report. For instance, the number of bytes transferred via WiFi, number of crashes, gps on time; the full list of features used in our model can be found in Table 2. Each report also contains a scalar $y$ corresponding to the total battery consumed by the device measured in milliampere-hours (mAh). We assume there exists a weight vector $\overline{\mathbf{w}} \in \mathbb{R}^d$ such that $\overline{\mathbf{w}}^\top \mathbf{x} \approx y$. Our goal is to estimate a weight vector $\widehat{\mathbf{w}} \in \mathbb{R}^d$ such that $\widehat{\mathbf{w}} \approx \mathbf{w}$. The estimated weight vector can then be used in two ways:

- **Predicting power consumption at device level.** Given a vector $\mathbf{x}$ representing the resource

usage of a device during a time period, $\widehat{\mathbf{w}}^\top \mathbf{x}$ is an estimate of the battery consumption of the device during the period. Although this consumption can be directly measured without the need for a model, this step can be useful for model evaluation.

- **Predicting power at an app level.** Given a vector $\mathbf{x}$ representing the resource usage of a single app during a time period, $\widehat{\mathbf{w}}^\top \mathbf{x}$ is an estimate of the battery consumption of the app during the period.

## 3. Power Allocation

Using the model introduced in section 2 standard results ensure that the prediction of total power will be accurate on new reports. Can we also guarantee that high accuracy in total prediction power implies a correct allocation of power consumption blame to each app?

In this section we answer the above question affirmatively under certain natural assumptions.

**Assumption 1.** *There exists $\overline{\mathbf{w}} \in \mathbb{R}^d$ such that for any sequence of reports $\mathbf{x}_i$ the total power consumption of report $i$ is given by $y_i = \overline{\mathbf{w}}^\top \mathbf{x}_i + \epsilon_i$, where $\epsilon_i$ are i.i.d. random variables with $\mathbb{E}[\epsilon_i] = 0$ and there exists $R > 0$ such that $\mathbb{E}[e^{\eta \epsilon_i}] \leq e^{\frac{R^2\eta^2}{2}}$ for all $\eta \in \mathbb{R}$.*

The assumption on the exponential moment of the noise is weaker than assuming that the noise falls over the interval $[-R, R]$. (We defer the proof of the Theorem to the full version of the paper.)

**Theorem 1.** *Let $\mathbf{X}_m = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be the matrix whose columns are given by the reports, $\mathbf{y}_m = (y_1, \ldots, y_n)$ be the vector formed by the observed power consumptions and let $\mathbf{V}_m = \lambda \mathbf{I} + \mathbf{X}_m \mathbf{X}_m^\top$. Let also $\widehat{\mathbf{w}} = \mathbf{V}_m^{-1} \mathbf{X}_m \mathbf{y}_m$ denote the solution to our optimization problem. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the randomness of the noise we have*

$$\|\widehat{\mathbf{w}} - \overline{\mathbf{w}}\| \;\leq\; R\sqrt{\frac{2dn \log\left(\frac{1+\frac{mL^2}{\lambda}}{\delta}\right)}{\lambda + \sigma_{\min}}} + \frac{\lambda \|\overline{\mathbf{w}}\|}{\lambda + \sigma_{\min}},$$

*where $\sigma_{\min}$ denotes the minimum eigenvalue of $\mathbf{X}_m \mathbf{X}_m^\top$ and $L = \max_{i \in \{1,\ldots,m\}} \|\mathbf{x}_i\|$.*

**Corollary 1.** *If $\sigma_{min} \geq m\sigma$ for some $\sigma > 0$ then*

$$\begin{aligned}\|\widehat{\mathbf{w}} - \overline{\mathbf{w}}\| &\leq M\sqrt{\frac{2dn\log(1+mL^2) + \log(1/\delta)}{1+m\sigma}} \\ &\quad + \frac{1}{1+m\sigma}\|\overline{\mathbf{w}}\| \\ &= O\Big(M\sqrt{\frac{\log(m) + \log(1/\delta)}{\sigma m}} + \frac{\|\overline{\mathbf{w}}\|}{\sigma m}\Big)\end{aligned}$$

The previous corollary gives us conditions for which, given enough data, our algorithm will find the true discharge rate for each component. In particular, we require the eigenvalues of the covariance matrix $\frac{1}{m}\mathbf{X}_m \mathbf{X}_m^\top$ to be bounded away from zero. It is easy to show that with high probability this property of the covariance matrix holds when there is no set of components that is perfectly correlated. Moreover, this condition can be empirically verified and Figure 1(a) shows that the smallest eigenvalue of the covariance matrix is roughly $0.2 \gg 0$. This implies we can correctly allocate *blame* to each component. Furthermore, if the power drainage of an app corresponds to the sum of the power drainage of its components, we can then accurately allocate blame to apps.

## 4. Experiments

Here we report the performance of our model and compare it against a model trained on micro-benchmarks. The micro-benchmark (MB) model was trained by using controlled laboratory experiments where each component was ran on isolation and the power used by the phone was regressed against that particular component usage. Our data consists of a sample from 19 days of reports or approximately $300,000$ reports covering a varied set of devices. A more detailed description of the types of devices found in our training set can be found in Figure 1(b). Our model was trained using stochastic gradient descent on the square loss. Our test set consists of $m = 19,000$ unseen examples. Figure 2 reports the distribution of errors, for both models, between the predicted battery consumption and the true battery consumption reported by the device.

Notice that our model consistently outperforms the model trained on micro-benchmarks. And in fact the mean prediction of our model is 8x better than the MB model.

To validate the way our model allocates battery blame to different components we conducted various experiments with a portable power monitor on a Nexus 5X. The experiments were designed to run certain apps while measuring the amount of battery drained. Given that we can control the apps being used we are able to come up with a ground truth on how much power each app was consuming. In Table 4 we report the errors on the predictions of our model and the MB model. In this experimental setup we are again achieving better accuracy across the board. This is remarkable since the MB model was trained under laboratory conditions and we therefore expect it to be better, yet only on a couple of examples does it beat our learned model.

| | | | |
|---|---|---|---|
| mobile active time | full wakelock time | partial wakelock | phone on time |
| gps on time | bluetooth on | low power mode enabled time | mobile bytes rx |
| mobile bytes | wifi bytes rx | wifi bytes tx | wifi on |
| wifi running time | wifi scan time | wifi scan | full wifi lock time |
| system kernel overhead time | wakeup | kernel wakeup count | app wakeup count |
| idle | screen brightness dark time | screen brightness dim time | screen brightness medium |
| screen brightness light time | screen brightness time | wifi strength none time | wifi signal strength poor time |
| wifi signal strength moderate time | wifi signal strength good | wifi signal strength great time | wifi signal strength none count |
| wifi signal strength poor | wifi signal strength moderate count | wifi signal strength good count | wifi signal strength great |
| signal strength none or unknown time | signal strength poor time | signal strength moderate time | signal strength good time |
| signal strength great time | signal strength none or unknown | signal strength poor count | signal strength moderate count |
| signal strength good | signal strength great count | process crashes | process anrs |
| cpu user time | cpu system time | cpu power | flashlight count |
| audio time | audio | video time | video count |

*Table 2.* Features included in the model

| App | Relative error (learned) | Relative error (MB) |
|---|---|---|
| freewarmer | **14.7%** | 58.9% |
| music | 54.5% | **39.7%** |
| mytracks | **0.35%** | 87.6% |
| screen | **7.5%** | 61.2% |
| voltair | **15.8%** | 71.9% |
| youtube | **43.4%** | 86.2% |
| chrome | 69% | **59.9%** |

*Table 3.* Results of battery prediction in reports labeled by portable power monitor. The columns correspond respectively to the relative error (absolute mAh error / true mAh consumption) of the learned model and the micro-benchmark (MB) model.

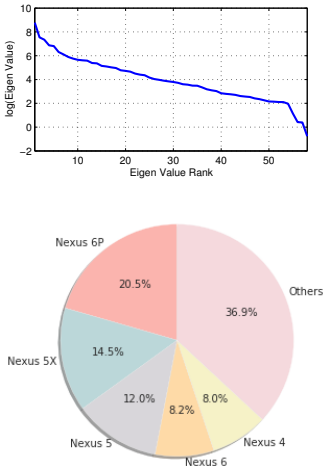


*Figure 1.* (a) Distribution of eigenvalues of empirical covariance matrix. Notice that the smallest eigenvalue is bounded away from zero as required by Theorem 1. (b) Distribution of devices in our training data.

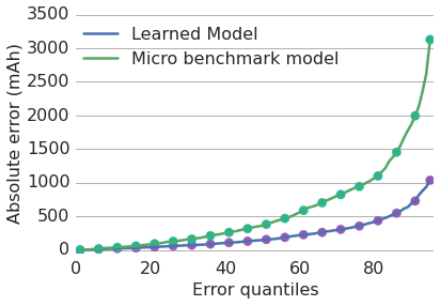


*Figure 2.* Absolute prediction error for our learned model (in blue) and the micro-benchmark model (in green).

## 5. Conclusion and Open Problems

We introduced a data-driven model for battery prediction. Unlike previous work, our model does not require the use of controlled experiments. Our model is therefore easier to train, adaptive to different usage conditions and more robust to changes in software. Moreover, we demonstrated that this learned model consistently outperforms a model trained on micro-benchmark experiments across different tasks.

A natural extension of this work is that of non-linear models. Whereas we believe the assumptions made in this paper are reasonable, preliminary experiments have shown that non-linear models can achieve better accuracy than linear ones. However, if a device's battery consumption cannot be decomposed as a sum of each component's consumption, it is non-trivial to solve the battery attribution problem. An interesting question is whether a set of *natural* assumptions for this problem exist that would induce a unique allocation rule. More importantly, if such allocation exists, would there be an analog to Theorem 1 for it?

## References

Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM*, pages 280–293, 2009.

Xiaomeng Chen, Ning Ding, Abhilash Jindal, Y. Charlie Hu, Maruti Gupta, and Rath Vannithamby. Smartphone energy drain in the wild: Analysis and implications. In *Proceedings of the 2015 ACM SIGMETRICS June 15-19, 2015*, pages 151–164, 2015.

Ning Ding, Daniel T. Wagner, Xiaomeng Chen, Y. Charlie Hu, and Andrew C. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In *Proceedings of ACM SIGMETRICS*, pages 29–40, 2013.

Ella Peltonen, Eemil Lagerspetz, Petteri Nurmi, and Sasu Tarkoma. Energy modeling of system settings: A crowdsourced approach. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015, St. Louis, MO, USA, 23-27 March, 2015*, pages 37–45, 2015.

Sanae Rosen, Ashkan Nikravesh, Yihua Guo, Zhuoqing Morley Mao, Feng Qian, and Subhabrata Sen. Revisiting network energy efficiency of mobile apps: Performance in the wild. In *Proceedings of the 2015 ACM IMC*, pages 339–345, 2015.

Fred Schlachter. No Moore's law for batteries. *Proceedings of the National Academy of Sciences*, 110 (14):5273, 2013.