Power Consumption Estimation for Laptops a Machine Learning Approach

Carlota Parés Morlans
Department of Electrical Engineering
Stanford University
Palo Alto, CA 94305
cpares@stanford.edu

Ruben Rodriguez Buchillon
Google Inc.
Mountain View, CA 94043
coconutruben@google.com

Udaya Kiran Ammu *
Google Inc.
Mountain View, CA 94043
udaykiran@google.com

Puthikorn Voravootivat
Google Inc.
Mountain View, CA 94043
puthik@google.com

Milad Hashemi
Google Inc.
Mountain View, CA 94043
miladh@google.com

Abstract

Power consumption estimation is key for correctly sizing the battery of portable devices as it directly affects their cost, weight, and lifetime. Existing laptop design flows rely on the know-how of domain-experts, who analytically size laptop batteries based on prior knowledge, heuristics and component data sheets. This method is non-standard, difficult to audit, and fails to capture the non-linear relationships in the interplay between hardware components. In this paper, we leverage Google's existing Chromebook (ChromeOS laptop) development and testing laboratory data to present a machine learning (ML) based framework for data-based power consumption prediction for a given workload. First, we collect a dataset of over 100 Chromebook designs including different workload tests and power-relevant hardware components. Processed features are then fed into an ML model that successfully predicts the mean and standard deviation of the power consumed by a combination of hardware components with a RMSE of 0.39W$\pm$0.06 and 0.15W$\pm$0.007, respectively. Our results demonstrate that a ML model can learn the function mapping a hardware description to total system consumption based on existing data.

1 Introduction

Power consumption prediction and battery sizing go hand-in-hand as one dictates the other: when designers want to ensure a specific runtime on a given workload, they first characterize the power consumption of the workload, and then pick a battery size that meets the desired runtime. Common methodologies for battery size estimation include system domain experts (e.g. system-on-a-chip (SoC) manufacturers like Intel [1], or hardware designers) running a set of workloads on multiple hardware variations to extrapolate the impact of a hardware component on the system's average power consumption. That approach lacks precision, scalability, and generalizability, and for this reason the research community has proposed multiple approaches that tackle this problem by focusing on either single device power consumption breakdown or power consumption modeling.

Carroll et al. [2] provide an in depth analysis of the power consumed by the different hardware components of a smartphone. Similarly, Pramanik et al. [3] detail the various entities and factors

*Corresponding author.

Preprint. Under review.

that contribute to the total power consumption in a smartphone. Focusing on laptops, Mahesri et al. [4] analyze the power consumption of a laptop, resulting in insights on workload power variability as well as the power distribution among the different hardware components. These works, while providing meaningful information of the device under test, lack generalizability to other devices. Additionally, most published power breakdown analyses are based on smartphones, which share some characteristics with laptops, but have very different usage patterns and thus battery requirements.

Prior work has also proposed analytically computing device power consumption by individually modeling each hardware component [2, 5]. These approaches, even if they shed light on methodologies for power estimation, are very restricted to the context where they have proven successful. Recently, Intel has released a tool, Power Calculator [6], which allows users to select a hardware component configuration and generate power consumption estimates. This tool is more similar to our work as it allows customizable hardware configurations and workloads. However, the diversity of components is limited to Intel server configurations and workload definition is restricted to describing subsystem utilization.

In this work, we tackle modeling power consumption from a unique perspective due to the particularities of Chromebooks. Unlike individual laptop manufacturers that must restrict their hardware breadth to a specific set of components, ChromeOS supervises different providers that supply a wide variety of hardware designs that run the same software stack. Therefore, given the 3 main factors that directly affect power consumption: (i) workload, (ii) software, and (iii) hardware, we fix, by definition, (i) and (ii), to focus on the power consumption of hardware components. Then, the power consumption for a certain workload can be modeled as follows:

$$P(x) = z \ [mW] \tag{1}$$

Moreover, we have modeled the power consumption of a set of hardware components as a Gaussian, given that other factors such as room temperature, battery life-cycle, and sensitivity of external measurement tools add variability to power consumption. Thus, the power function can be written as:

$$P(x) = \mathcal{N}_z(\mu, \sigma) \ [mW] \tag{2}$$

The main contributions of this paper are the following:

- We define a laptop hardware parametrization correlated with power consumption.
- We collect a dataset of over 100 Chromebook designs, spanning chip designs, display sizes and technologies, and form-factors, and their total system power consumption for different workload tests and power-relevant hardware components.
- We build a Machine Learning (ML) framework that processes dataset features and successfully predicts the power consumption of a set of hardware components for two different workloads.

2 Data collection

In this section, we explain the setup used to capture our dataset.

2.1 Experiment Setup

Our experimental setup is shown in Figure 1. It includes two units: (i) a Chromebook, also referred as device-under-test (DUT) and (ii) a servo [7], a proprietary debug tool. This hardware configuration is then modulated using autotests [8], a standard Python framework for fully automated testing. The main power information comes from the smart battery compliant battery controller [9], included in all Chromebooks and accessible through a kernel interface [10]. As indicated in the battery controller specification [9], the system provides a $\pm 1\%$ accuracy on voltage, current, and average current. However, in practice, we have seen modern batteries have a significantly higher accuracy.

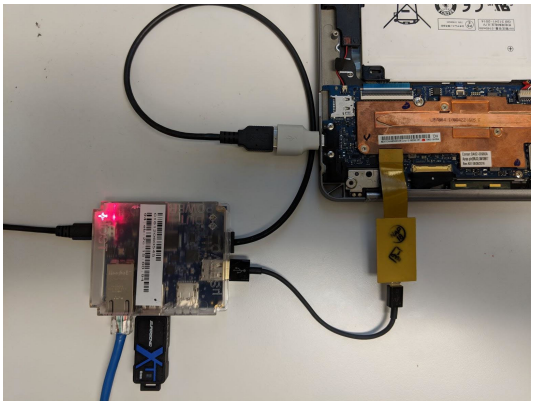

Figure 1: Laboratory Setup Diagram

2.2 Usage scenarios

In this work, we have captured the relationship between hardware components and energy consumption for two different usage scenarios. We chose these two workloads from the ChromeOS power key performance indicators (KPIs) collection given their low power deviation, high relationship with battery sizing, and good representation of user laptop usage.

Video Playback. This usage scenario is designed to measure the power laptops consume when playing a video. A total of 22 videos are played over the span of 50 minutes, including different resolutions (720p, 1080p, 4k) and video codecs (H.264, VP8, VP9, AV1).

Idle. This usage scenario is designed to measure the power laptops consume when they are turned on (white screen) and idling as much as possible. Regular maintenance tasks remain running. The laptop remains idle for a total of 120 seconds.

2.3 Dataset

In total, we measured the power consumption of 153 different Chromebooks with an average of 20 tests per workload. We consider the average and standard deviation of the system's total power consumption over all tests our dependent variables. Then, for each of the devices, we recorded the values of 20 hardware components including display resolution, CPU cores, storage size, and memory size amongst others. These features are our independent variables. For a detailed list of the extracted features see Appendix. While ChromeOS works with various SoC providers, we restricted the dataset to Intel and AMD SoCs (x86 architecture).

3 Methodology

In this section, we describe the different steps implemented to built a Machine Learning (ML) framework that uses the dataset detailed in Section 2 to successfully predict the power consumption of a combination of hardware components.

3.1 Data Preparation

Data preparation, involves different techniques that transform raw data to a form that is suitable for a learning algorithms. In this project, we applied data cleaning and feature engineering to preprocess the data obtained from the experimental setup 2.1.
Data cleaning. This task involves the correction of errors in the data. Domain knowledge is used to identify incorrect observations. Once identified, they are either removed or the issue is fixed in the pipeline and data recollected. Particularly, in this work, we have applied the following cleaning steps:

- Removed network dependent and warming up workloads.
- Removed DUTs with fewer than 5 runs per workload.

- Removed outliers using: x > 5% quantile, x < 95% quantile.
- Removed DUTs that have components which have never been used in a different device.

Feature Engineering. This task refers to the process of deriving new variables from the available data. The transformations applied in this work include scaling of numerical features to have 0 mean and unit variance and label encoding of categorical features.

3.2 Model architecture

The model architecture, as seen in Figure 2 is a FCNN with 5 hidden layers and a 2 dimensional output: the predicted mean and variance of the power consumed by a set of hardware components. We used ReLU as our activation function and added a 10% dropout to avoid overfitting.

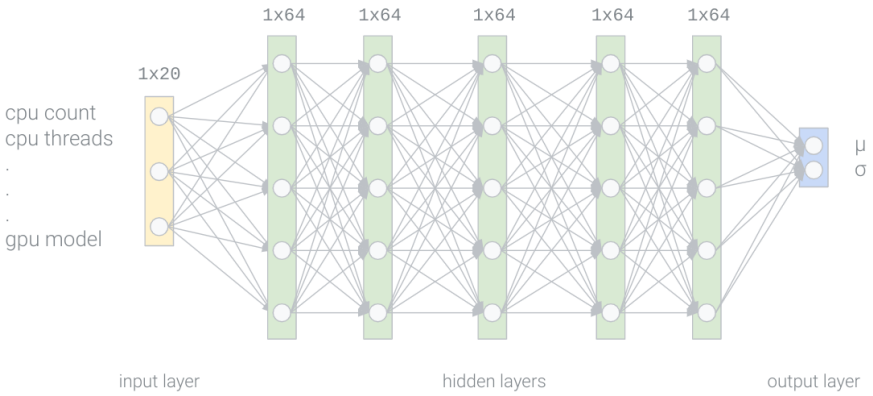

Figure 2: Neural Network Architecture Diagram

3.2.1 Loss Function

To train the previously outlined NN architecture, we have used RMSE as our loss function. We choose to minimize RSME as it has the same units as the target variable and severely penalizes large errors.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2} \tag{3}$$

3.2.2 Evaluation

To train the models (one for every workload), we split the dataset into 80% train and 20% test using a pseudo-random strategy. Specifically, we apply a iterative stratification process [11] that allows us to ensure that categorical features are proportionally distributed between the two sets, avoiding the prediction of an unseen hardware component. We train and evaluate the model on 5 different folds for every workload.

To optimize the aforementioned loss function, we employed Adam algorithm. We trained the model for 150 epochs and evaluated the test set on the last checkpoint. A list of the hyperparameters is included in the Appendix.

4 Results

In Table 1, we summarize the model's performance. Overall, the model appears to successfully generalize predictions to configurations where each individual component has been seen before, but the entire set wasn't known to the model. We can see that even though both tests have a similar RMSE for the average power consumption, we get a higher error for the standard deviation. Moreover, compared to the global values, we get higher errors for the idle test. This difference can be explained

by the variability observed in the power measurements for such test, which we relate with the shortness of the workload, making it very dependent on the CPU state and the background task noise of that particular time.

Table 1: Model results per workload

Test	Global μ	μ RMSE	Global σ	σ RMSE
Video Playback	5.99W $\pm$ 1.15	0.39W $\pm$ 0.06	0.27W $\pm$ 0.177	0.15W $\pm$ 0.007
Idle	3.89W $\pm$ 1.05	0.375W $\pm$ 0.006	0.34W $\pm$ 0.296	0.277W $\pm$ 0.005

Given these results, two direct applications for battery size estimation arise. First, if the goal is to determine which battery size can hold a workload for a certain amount of time, we can build a Gaussian around that lifetime extrapolating the predicted parameters and obtain a lower and upper bound for battery size. Second, if our goal is to keep the battery size at a certain value, we can use the resulting predictions to mix and match different hardware components that bring us to the desired value.

5 Limitations

In this section, we outline the main limitations of this work:

- The testing framework adds an observer cost for setup, teardown, and period polling of power statistics from the kernel.
- The attached servo hardware debugger adds an observer cost remaining physically connected to the DUT during test execution, keeping some Type-C components from idling.
- The power data comes from the battery controller. It can differ from device to device, leading accuracy difference while remaining specification compliant. An external measurement device can mitigate this, increasing cost and complexity of the test setup.
- The current dataset is based on two different workloads and it is restricted to only two SoC providers.

From the aforementioned limitations, we believe that the power overhead added by the measurement setup is minimal and constant, with a restricted influence on the collected data. We plan on addressing the limitations associated with the dataset in future work to potentially lead to a commercially viable product.

6 Conclusions and Future Work

In this work, we have presented an ML approach for power consumption prediction given a set of hardware components. Precisely, we defined a series of parameters that extract the relationship between the hardware components of a laptop and the total system power consumption. We then processed this data and fed it to a machine learning model that learns the relationships between the different hardware components and the total consumption of a given set: $P(x) = \mathcal{N}_z(\mu, \sigma)\ [mW]$. Our results indicate that learning approaches can model this function and provide average power consumption predictions with an RMSE of 0.39W $\pm$0.06.

Future work of interest would include the integration of component datasheet power consumption values as features to guide the model predictions. A similar step further would be to take an approach like the one proposed by Zou et al. [12] and measure the component power directly, avoiding potential datasheet inaccuracies. Another future step would involve an improvement on data quality and dataset extension. We believe that the workloads and testbeds available for ChromeOS produce significant noise, even for seemingly simple workloads like idle consumption. While the proposed machine learning framework needs to be able to handle variations, especially for live network tests, it would be helpful to have some tests with minimal variance to validate that the network predicts within the source data variation. Lastly, it would be interesting to study how the addition of the workload description into the prediction input affects the model estimates. This would allow the network to share information across workloads and turn the output into a prediction for *all* workloads.

Authors' Contributions

The authors confirm contribution to the paper as follows: U.K., R.R.B., and O.V. conceived the idea. C.P.M. collected the data, designed and performed the experiments, derived the models and analysed the data. C.P.M wrote the manuscript with the support of R.R.B. All authors reviewed the results and approved the final version of the manuscript.

Acknowledgments and Disclosure of Funding

We sincerely thank Stefan Reinauer, Chithra Annegowda, Prajakta Gudadhe for their help and insight for making our work possible at Google. We thank Mengqi Guo for her discussions and insights. We thank Swapna Iyer, Summer Wang, and Subhajit Dasgupta for their continued support through out the project. We also thank David Lo, Parthasarathy Ranganathan, Huan Ren for reviewing and giving valuable suggestions.

References

[1] "Early power estimator overview," Jul 2021, accessed 2022-09-25. [Online]. Available: https://www.intel.com/content/www/us/en/docs/programmable/683272/current/early-power-estimator-overview.html

[2] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[3] P. K. D. Pramanik, N. Sinhababu, B. Mukherjee, S. Padmanaban, A. Maity, B. K. Upadhyaya, J. B. Holm-Nielsen, and P. Choudhury, "Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage," *IEEE Access*, vol. 7, pp. 182 113–182 172, 2019.

[4] A. Mahesri and V. Vardhan, "Power consumption breakdown on a modern laptop," in *International Workshop on Power-Aware Computer Systems*. Springer, 2004, pp. 165–180.

[5] M. Kim, J. Kong, and S. W. Chung, "Enhancing online power estimation accuracy for smartphones," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 333–339, 2012.

[6] "Power calculator," accessed 2022-09-25. [Online]. Available: https://servertools.intel.com/tools/power-calculator/

[7] "hdctools: Chrome os hardware debug control tools," Sept 2022, accessed 2022-09-25. [Online]. Available: https://chromium.googlesource.com/chromiumos/third_party/hdctools/+/HEAD/README.md

[8] "Autotest for chromium os developers," May 2022, accessed 2022-09-25. [Online]. Available: https://chromium.googlesource.com/chromiumos/third_party/autotest/+/HEAD/docs/user-doc.md

[9] "Smart battery data specification," Dec 1998, accessed 2022-09-25. [Online]. Available: http://sbs-forum.org/specs/sbdat110.pdf

[10] "sysfs-class-power," Nov 2021, accessed 2022-09-25. [Online]. Available: https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-class-power

[11] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the stratification of multi-label data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 145–158.

[12] L. Zou, A. Javed, and G.-M. Muntean, "Smart mobile device power consumption measurement for video streaming in wireless environments: Wifi vs. lte," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017, pp. 1–6.

Appendix

DUT Feature List

Table 2: DUT Feature List

Component	Feature	Data Type
SoC	Number of CPUs	Integer
	Number of threads	Integer
	Number of cores	Integer
	L3 cache size	Integer
	Vendor	String
GPU	Name	String
Memory	Size	Integer
	Type	String
	Frequency	Integer
Storage	Size	Integer
	Type	String
Display	Resolution	Integer
	Size	Integer
	Refresh Rate	Integer
NIC	Name	String
Embedded Controller	Name	String
Touchscreen	Exists	Boolean
Fingerprint	Exists	Boolean
Cellular Connectivity	Exists	Boolean
Stylus	Exists	Boolean

Model Hyperparameters

Table 3: NN Hyperparameters

Hyperparameter	Value
learning rate	0.001
epochs	150
momentum	0.9
weight decay	1e-06
batch size	8
activation function	ReLU
dropout	0.1