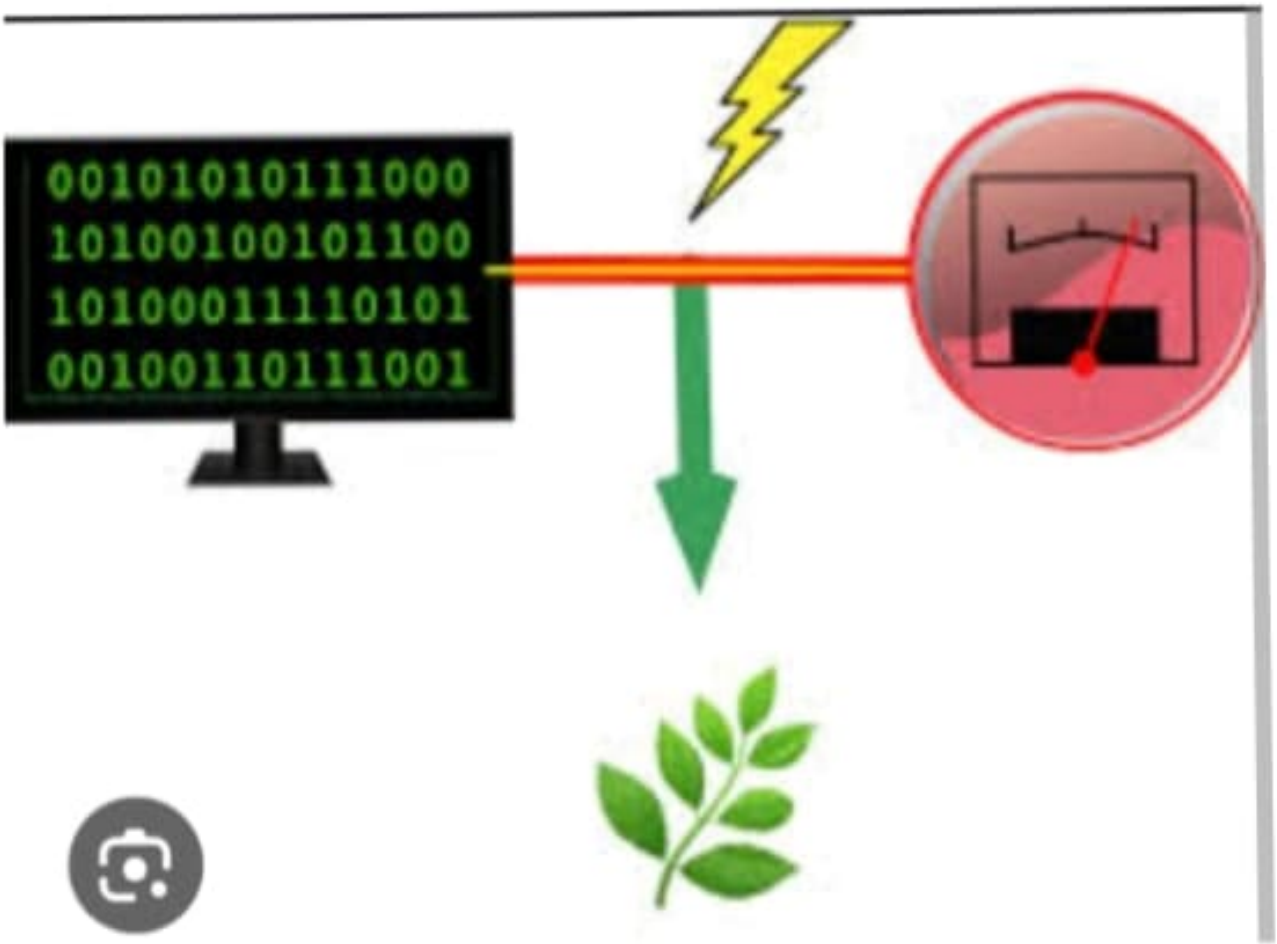
INTRODUCTION:



Energy consumption has been widely studied in the computer architecture field for decades. While the adoption of energy as a metric in machine learning is emerging, the majority of research is still primarily focused on obtaining high levels of accuracy without any computational constraint. We believe that one of the reasons for this lack of interest is due to their lack of familiarity with approaches to evaluate energy consumption. To address this challenge, we present a review of the different approaches to estimate energy consumption in general and machine learning applications in particular. Our goal is to provide useful guidelines to the machine learning community giving them the fundamental knowledge to use and build specific energy estimation methods for machine learning algorithms. We also present the latest software tools that give energy estimation values, together with two use cases that enhance the study of energy consumption in machine learning.

2.1 Energy Consumption of software

Green and sustainable software is a software product that has the smallest possible economic, societal, ecological impact as well as impact on human beings (Ahmed, et al., 2014). This has led to the introduction of various programmes and initiatives that encourages energy efficient software such as green software engineering and Eco-design software (Kaliterre, n.d.).

According to the Greenhouse Gas Protocol (2012), applications are executed with an OS. They affect the power consumption of a device due to data requests and processing. Managing energy requires accurate measurement of the energy available and consumed by a system. This involves monitoring or estimating the resource and energy consumption of hardware and software (Noureddine, et al., 2013). However, a device's power consumption is subjected to the type of application and the task being performed which is evident in our experimental results presented in Section 4 of this paper. In order to reduce the overall power consumption for a web-based or standalone task, it will be necessary to provide users with an insight of the power consumption of the different web-based browser applications (e.g. Google Chrome, Internet Explorer, Mozilla Firefox, Safari, etc...) and also the resource hungry nature of many applications such as movie player and games.

4.1 Web Browser Applications for Windows

Browser	Experiment No	Hardware Energy Consumption (J)					Application (J)	Total Energy Consumption (J)	Total Time (s)	Aggregated Average Energy Consumption (J/sec/s)
		CPU (J)	Monitor (J)	Disk (J)	Base (J)	Hardware				
Google Chrome 1.3.27	1	111.168	154.130	0.000	378.319	643.616	106.652	750.268	140.836	
	2	101.205	157.087	0.000	385.576	643.868	96.657	740.524	142.808	
	3	103.381	155.673	0.000	382.107	641.781	98.958	740.739	141.521	
	4	102.365	152.213	0.000	373.613	628.190	98.770	726.960	138.275	
	5	103.215	156.681	0.000	394.580	650.176	105.205	755.781	142.437	
	6	135.710	155.001	0.000	380.457	671.167	126.990	798.157	140.930	
	7*	Error								
	8	126.437	155.237	0.000	381.183	662.917	119.757	782.674	141.175	
	9*	76.783	97.493	0.000	239.296	413.556	74.544	488.100	88.626	
	10	106.745	155.394	0.000	381.425	703.563	107.086	861.445	144.637	5.442
Internet Explorer 9	1	45.784	154.217	0.000	378.532	578.532	38.276	616.788	140.197	
	2	43.317	152.142	0.000	397.305	609.464	43.736	653.220	147.402	
	3	50.285	152.635	0.000	374.649	577.569	44.340	621.917	138.759	
	4	50.734	156.955	0.000	395.252	601.007	50.227	651.228	142.686	
	5	51.618	156.134	0.000	383.238	590.990	46.809	637.791	141.940	
	6*	Error								
	7*	Error								
	8*	32.282	116.732	0.000	286.524	435.540	17.344	452.892	106.320	
	9	47.094	153.671	0.000	377.684	578.640	34.351	612.990	139.883	
	10	57.187	153.783	0.000	377.468	588.439	49.972	638.411	139.803	4.454
Mozilla Firefox 27.0.1	1	329.149	157.489	0.000	386.564	672.203	118.922	791.125	143.172	
	2	62.580	154.375	0.000	378.921	595.876	75.664	691.540	140.341	
	3	97.265	155.535	0.000	381.777	634.581	88.251	722.832	141.399	
	4	73.310	155.602	0.000	382.423	612.134	66.883	679.017	141.630	
	5	87.325	153.672	0.000	391.921	636.918	78.392	717.310	145.156	
	6	328.675	153.595	0.000	376.982	859.243	95.772	775.015	139.623	
	7	87.172	154.356	0.000	378.875	620.403	80.606	701.011	140.324	
	8	86.152	155.170	0.000	380.873	622.195	78.340	700.495	141.064	
	9	98.039	155.811	0.000	382.450	636.297	89.956	726.237	141.645	
	10	90.892	154.529	0.000	379.299	624.720	83.558	707.879	140.485	5.098
Safari 5.7.1	1	102.221	154.069	0.000	389.845	646.145	98.505	744.650	144.387	
	2	88.365	151.370	0.000	383.798	623.483	83.946	707.407	142.143	
	3	89.013	150.006	0.000	381.194	620.213	92.471	721.684	141.183	
	4	92.004	149.867	0.000	380.047	621.917	89.480	691.397	140.756	
	5	93.047	146.667	0.000	371.272	610.985	80.209	691.195	137.508	
	6	104.067	149.436	0.000	380.489	633.992	146.294	830.286	140.922	
	7	95.606	148.621	0.000	378.723	623.236	90.298	713.533	141.378	
	8	104.739	150.849	0.000	383.824	639.413	98.896	738.307	142.157	
	9	97.310	150.679	0.000	383.846	632.833	92.843	725.276	142.185	
	10	94.616	147.526	0.000	372.743	614.885	80.425	694.306	141.267	5.134

Note: * - proportion of the remaining cleansed data > 50% of the raw data and ** is for otherwise and thus considered an error.
White coloured records – results of experiments for Day 1; Blue coloured records – results of experiments for Day 2.

Table 2: The hardware and application energy consumption for running several web browser applications

Table 3 depicts the aggregated data for all the experiments conducted for each web browser: Google Chrome, Internet Explorer, Mozilla Firefox, and Safari. However, in order to provide a fair comparison among the web browsers, the time for running the application will have to be set to 1s (i.e. t = 1s) Consequently, the corresponding energy consumption for each component will have to be normalised for t = 1s (see Table 4). Based on the results shown in Table 4, it seems that Internet Explorer 9 consumes the least energy on laptops, followed by Mozilla Firefox and Safari while Google Chrome seems to be the highest energy consumer. These results are consistent with experiments

Figure 3 depicts the energy consumption by the hardware and application for each web browser. It can be seen that the energy consumed by the application is very much less compared to the energy consumed by the hardware that runs the application. In order to reduce the overall energy consumption, a further investigation on the interface as well as processes that occur between the software and hardware will have to be conducted and optimized. The energy consumption patterns for the various web browsers are consistent with that for the hardware.

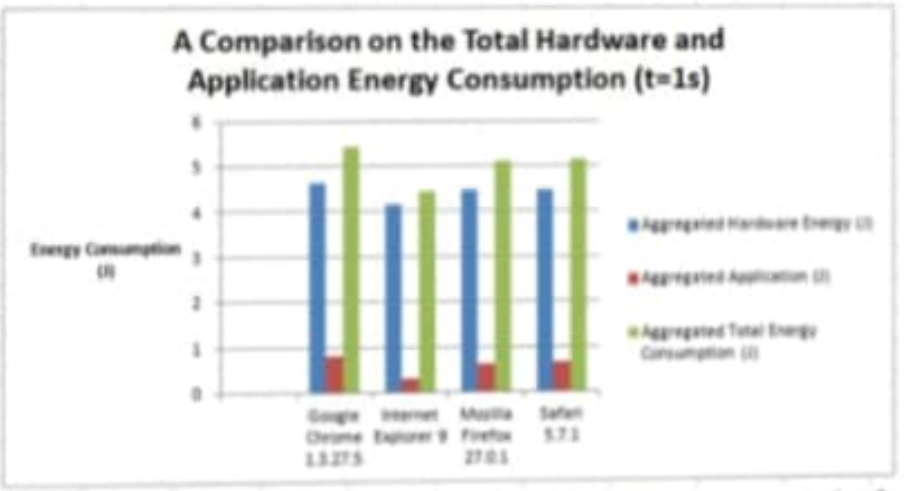


Figure 3: A comparison of the normalised hardware and application energy consumption for the web browsers

Further analyses have been conducted to investigate the ratio of energy consumption between the various web browsers and Internet Explorer (used as a base because it is the lowest energy consumer). Results in Table 5 reveal that the factor for application energy consumption by Google Chrome is almost 3 times that of the Internet Explorer while it is a factor of 2 for Mozilla Firefox and Safari. However, the corresponding factors for the hardware ares lower compared to the factor for the application. Additionally, running a youtube application on Google Chrome (in a laptop) seems to consume approximately 22% more energy than Internet Explorer (note: this is consistent with the 18% finding provided by Center for Sustainable Energy Systems at Fraunhofer USA)[5]. However, the increase for Mozilla Firefox and Safari seems to be in the region of 14%-15%.

Table 5: Energy consumption ratio for the various web browsers

As previously mentioned in Section 3, an experimental error may arise due to human error. The expected total uninterrupted application running time is 2 minutes and 14 seconds (134 seconds). The aggregated expected total application time has been calculated by taking into consideration the proportion of the cleansed data and also the number experiments that have been rendered as errors. The percentages of experimental error for the 4 web browsers are shown in Table 6 and it seems that the experimental error for the Google Chrome is the lowest while the rest is approximately 5 times of Google Chrome. In order to re-affirm the validity of the findings previously discussed, it will be essential to in crease the number of experiments for each browser and measures taken to reduce human inconsistency.

	Time in Joulemeter (s)	Expected Total application time (s)	Experimental error (%)
Google Chrome 1.3.27.5	1221.10	1206.00	1.25
Internet Explorer 9*	1096.79	1035.84	5.88
Mozilla Firefox 27.0.1	1414.85	1340.00	5.59
Safari 5.7.1	1413.87	1340.00	5.51

Table 6: Experimental Error

4.2 Web Browser Applications for IOS

Table 7 depicts the results of running experiments on an Apple ipad Air2 with and IOS operating system. A youtube video has been chosen for this set of experiments and the running time is 30 minutes. Additionally, the experiments are run forthree different times of the day: morning; noon; and night.

Watching YouTube video at (noon)

Browser	Battery % (Before)	Video Duration	Battery % (after)	Battery Consumption %	Brightness	Volume	Date
Chrome	100%	30 (min)	97%	3%	Auto	Full	05/05/2015
Safari	96%	30 (min)	87%	9%	Auto	Full	05/05/2015
Opera mini	87%	30 (min)	78%	9%	Auto	Full	05/05/2015
Puffin	77%	30 (min)	66%	11%	Auto	Full	05/05/2015

Watching YouTube video at (night)

Browser	Battery % (Before)	Video Duration	Battery % (after)	Battery Consumption %	Brightness	Volume	Date
Chrome	66%	30 (min)	62%	4%	Auto	Full	05/05/2015
Safari	62%	30 (min)	58%	4%	Auto	Full	05/05/2015
Opera mini	58%	30 (min)	54%	4%	Auto	Full	05/05/2015
Puffin	54%	30 (min)	48%	6%	Auto	Full	05/05/2015

Watching YouTube video at (morning)

Browser	Battery % (Before)	Video Duration	Battery % (after)	Battery Consumption %	Brightness	Volume	Date
Chrome	47%	30 (min)	40%	7%	Auto	Full	06/05/2015
Safari	39%	30 (min)	33%	6%	Auto	Full	06/05/2015
Opera mini	33%	30 (min)	28%	5%	Auto	Full	06/05/2015
Puffin	28%	30 (min)	18%	10%	Auto	Full	06/05/2015

Table 7: Results of battery consumption for the various web browsers on an IOS device

Firstly, it is noted that the battery consumption by the Puffin web browser is consistently the highest at any part of the day. However, the findings for the rest of the three web browsers (in Table 7) are rather inconclusive. Consequently, the average battery consumption for each web browser has been calculated and shown in Table 8. Once again, the average value for Puffin seems to be the highest while it is the lowest for Google Chrome. However, this finding does not seem to be aligned to the findings in the previous section where the energy consumption for Google Chrome is higher than

Electrical Consumer Measurement

The electrical active power measurement in three-phase current systems is based on the
following general formula (Formula 1):

P-Ug cos(p) Formula

Due to the power network quality with the resulting asynchrony of each phase, it is needed to
measure voltage U(t) and the current I(t) of each phase separately. The share of inductivity
and capacitance of each electric consumer leads to an asymmetric dispersion of the phase
specific sinus waves and high range of cos . Harmonic waves have an influence on the
Zero-Crossing-Detection within a measurement device and therefore on the identification of
the periodic time T of the measured sinus waves [14]. This is a critical point in energy calculation.

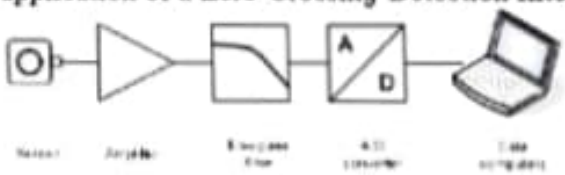
Simplified to p(t)= p = const, T_1, c_p = const.:

$$P(t) = c_p \cdot T_1 \cdot \left[\left(\frac{p(t)}{p_1} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right] \cdot \rho \cdot \dot{V}(t) = C_1 \cdot \dot{V}(t) \qquad \text{Formula 4}$$

The compression work within the compressed air system can be described by the adial change of state. Assuming an ideal gas, the compression work can be calculated by specific heat capacity, the intake air temperature, the in- and outtake pressure and mass flc

The simplification leads to an infrastructural dependent transformation that is seen in figure C_1. Based on the defined system border with the focus on the machine, the common transformation figure in C_1 with the value of 6.5 – 7.5 kW/(m³/min)⁴ for compressed air transformation is reasonable. Depending on the applied compressor and/or the compressed air distribution infrastructure, losses of up to 50% are possible[5]. The conversion factor C_1 represents a benchmark value in efficiency within the industry sector. The technical specification of the applied air flow measurement can be seen in Table 2.

Data acquisition with an n-fold sampling rate according to the signal of interest and the application of a Zero-Crossing-Detection filter can minimize the effects of harmonic waves.



$$f_{sample} \geq 2 \cdot f_{signal}$$ Formula 2

Fig. 1.Measuring chain setup

In the practical implementation (Fig.1), the measurements within the 50Hz network show that low pass filtering of 2000Hz fulfill the requirements of harmonic waves influence identification. The applied analog low-pass filters cut off all signals above 2000Hz. To avoid aliasing, the filtered signals of U(t) and I(t) are sampled by a sampling rate of 4000Hz by the A/D converter. This fulfills the Theorem of Shannon (Formula 2).

The signal can be acquired by a voltmeter for U(t) and by a hall effect sensor for I(t).
Such devices are commercially available, can be used to reach the mentioned requirements [15]. The data acquisition and the Zero-Detection are combined in the measurement data computing within the measuring device.

The applied device offers pulse and analog output for signal acquisition and control reasons (Fig.2/Table1). With a refreshing rate of 0.2 sec of the computed effective power value, it guarantees a near-real-time data acquisition.

Table 1
Technical specification of a sample measuring device.

Measuring principle	3-phase direct voltage measurement with hall effect current transformer
Sampling rate	4000 Hz
Output sampling rate	max. 5 Hz
Measuring error	U,I ≤ ± 1,0 % of measuring range
Output signal	bidirectional serial interface RS232

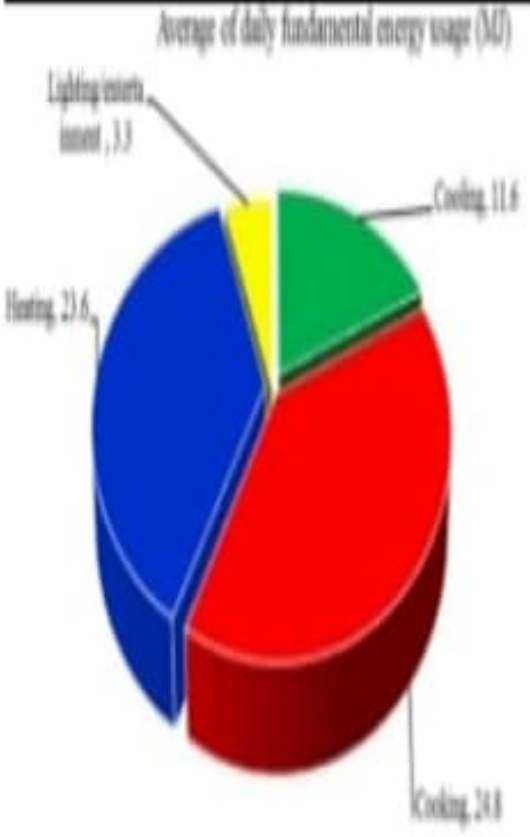
Compressed Air Measurement

For compressed air, as well as other gases, two separate and independent sensors (Fig. 2) were installed: a calorimetric flow sensor and a pressure sensor based on a DMS ceramic sensor for monitoring reasons. Several technical units for compressed air can be utilized for measurements. The applied technical unit for compressed air usage is Nm^3 (standard cubic meter), according to DIN 1343[3]. In correspondence with the defined system boundary, the data acquisition is gathered independently from the compressed air generator. Compressed air is to be measured based on the following assumptions (Formula 3).

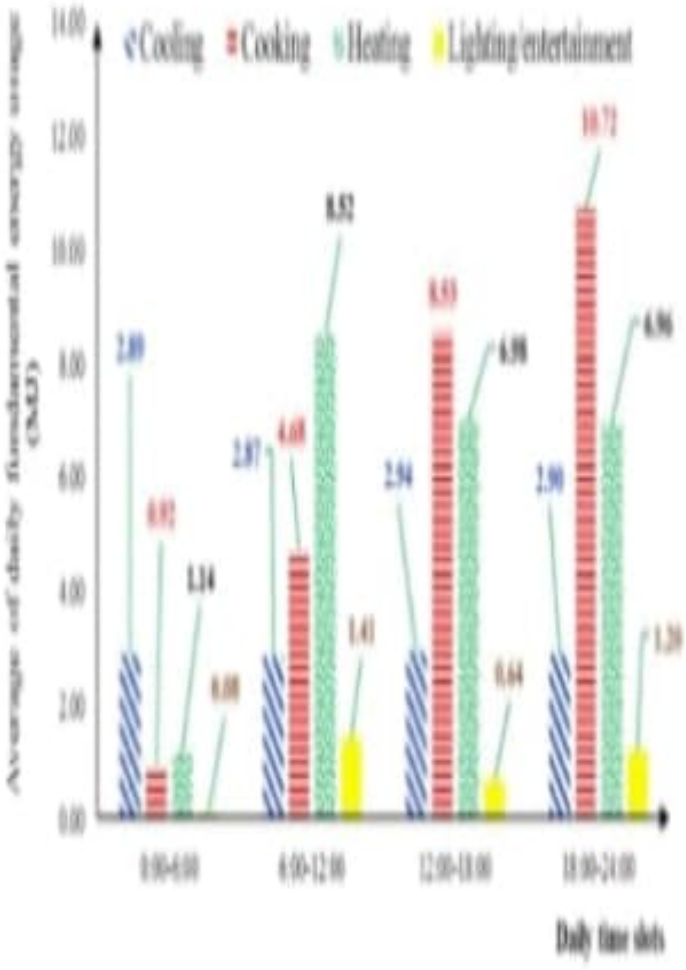
Assuming an adiabatic change of state:

$$P(t) = W(t) \cdot \dot{m}(t) = c_p \cdot T_1 \cdot \left[\left(\frac{p_1}{p_2} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right] \cdot \rho \cdot \dot{V}(t)$$ Formula 3

κ: isotropic exponent, 1.4
T_1: intake air temperature
p_1: intake pressure
p_2: outtake pressure
c_p: specific heat capacity
ρ: medium density



Average of daily fundamental energy usage (MJ)



Fundamental energy services in terms of daily hours

The measured data consists of one response variable power), and 165 explanatory variables. The task of model uilding is to explain the response as a function of the xplanatory variables. In order to cope with higher or- ler dependencies we also include the squared explanatory ariables (marked by a "SQ" at the variable name end), ielding a total of 330 explanatory variables. We did not nclude interactions between the variables, since this would ave lead to an explosion of the number of variables being ncomputable.

The task was now to identify a small number of variables hat would explain each individual data set, and additionally model that would explain all datasets in parallel. The de- ired models should be parsimonious (number of explanatory ariables should be low) and, in relation to this number, he model quality should be as good as possible. Model uality is mainly measured by R^2, the squared correlation oefficient, which explains how much better our model is vith respect to the simple data average. In more detail, R^2 neasures how much of the total variance (when using the verage of the data) is explained when exchanging the over- ll average with the respective model. Since $0 \leq R^2 \leq 1$, nd a larger value is better, as an initial goal we wanted to at east achieve $R^2 \geq 0.9$, and consider a model to be "good" f $R^2 \geq 0.95$.

. *Model Quality*

Model quality is also measured as average error σ, i.e., the quare root of the sum of the squared differences between redicted and measured values. This σ, being the average rediction error, is also set in relation to the average power, ielding the average error in %. Here it must be noted that imilar approaches usually report an average error of 5% or igher. Our task thus was to derived models that clearly esult in much smaller average errors. However, error in is of course very much influenced by the mean power onsumption of the computers, i.e., higher average power


USE CASE
AVEVA

Data was preprocessed by removing data items with at least one N/A value in any variable. The number of removed data lines however is quite small, in the order of below one per mill.

From data analysis we saw that there were only very few extremely influential points which can be identified using Cook's distance [5]. When removing them, the model quality quickly increases. We therefore also designed a simple outlier test removing those points which are at least 4σ away from the mean. This limit must be regarded as being extremely conservative, since usually either 2σ or 3σ are used. The result is a small number of outliers being removed, and we also state R^2 and σ for these cases, as well as the number of removed outliers.

Another test was done to see whether the residuals are normally distributed. However, since the dataset sizes are very high, and there are certain regularities left, using the Shapiro-Wilk test we cannot conclude that the residuals are indeed normally distributed. However, we plotted the data also on quantile-quantile-plots (Q-Q-Plots). On such a plot we put the quantiles of the theoretical normal distribution on the x-axis, and the empirically measured quantiles of the data on the y-axis. If the resulting plot is nearly linear (a line) then we can conclude that the data follows a normal distribution. The Q-Q-plots of our residuals indeed mainly follow a straight line, but show deviances at the tails, which explains why the Shapiro-Wilk test failed.

B. Additive Model Update

In our analysis we start with an empty model in which we only use the average to describe the response. We then use exhaustive search to find the best combination of N variables explaining the response. Since, when having K explanatory variables, the effort for doing this is

$$O(K(K-1)\cdots(K-N+1)),$$

this is not possible for larger values of both K and N due to an exponential runtime explosion.

To further reduce complexity we actually chose a mixture approach between additive and subtractive, and first start with a full model containing all variables. Then we remove all those variables that do not have any influence onto the result, by demanding that the p-value (of a t-test testing whether the coefficient is different from zero) of a variable should be larger than 0.5. This reduced K, thus speeding up the following additive approach significantly. This reduction of course is not possible if a full model cannot be created due to a lack of data. In these cases, we had to run the search for $K = 330$, which resulted in a drastically increased runtime. Still, combinations of more than $N = 4$ hardly make sense since they demand run times of days or months. However, there is seldom demand for doing so, since quite often only little can be gained by using more than three variables.

A further possibility is to sequentially add the next best L variables to the model. This means that we can start with an empty model, and then continuously find the best combination of L variables to add to the model. This approach still increases exponentially with L, but only linearly with N. We therefore also state the resulting models for $L = 1$.

IV. Analysis Results

The data consists of the six datasets "Burn CPU", "Mem Loop", "Network", "Tar Kernel", "Disk Read" and "Disk Write". The sizes of these sets are quite large with the exception of "Tar Kernel". The datasets do differ significantly with respect to their power consumption. The empirical cumulative distribution functions (ECDFs) are shown in Figure 1. An ECDF is the integral of the empirically measured density of a dataset. Data is concentrated at those values where the ECDF rises steeply. The datasets influenced by CPU and memory show a rather broad spectrum of power values. Those making heavily use of I/O like "Network" and "Disk X" show a rather narrow spectrum. "Tar Kernel", making use of both CPU and I/O is also narrow, but with a significant shift to the right.

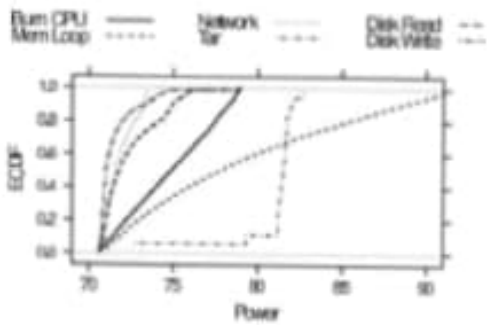


Figure 1. Empirical CDFs (ECDFs) of the respective power consumptions.

In the following we show results of the regression analysis, which also includes plots of the respective residuals. In these plots the x-axis is as important as the y-axis, since it shows the range of the power measurements. Residual sizes must also be related to these ranges.

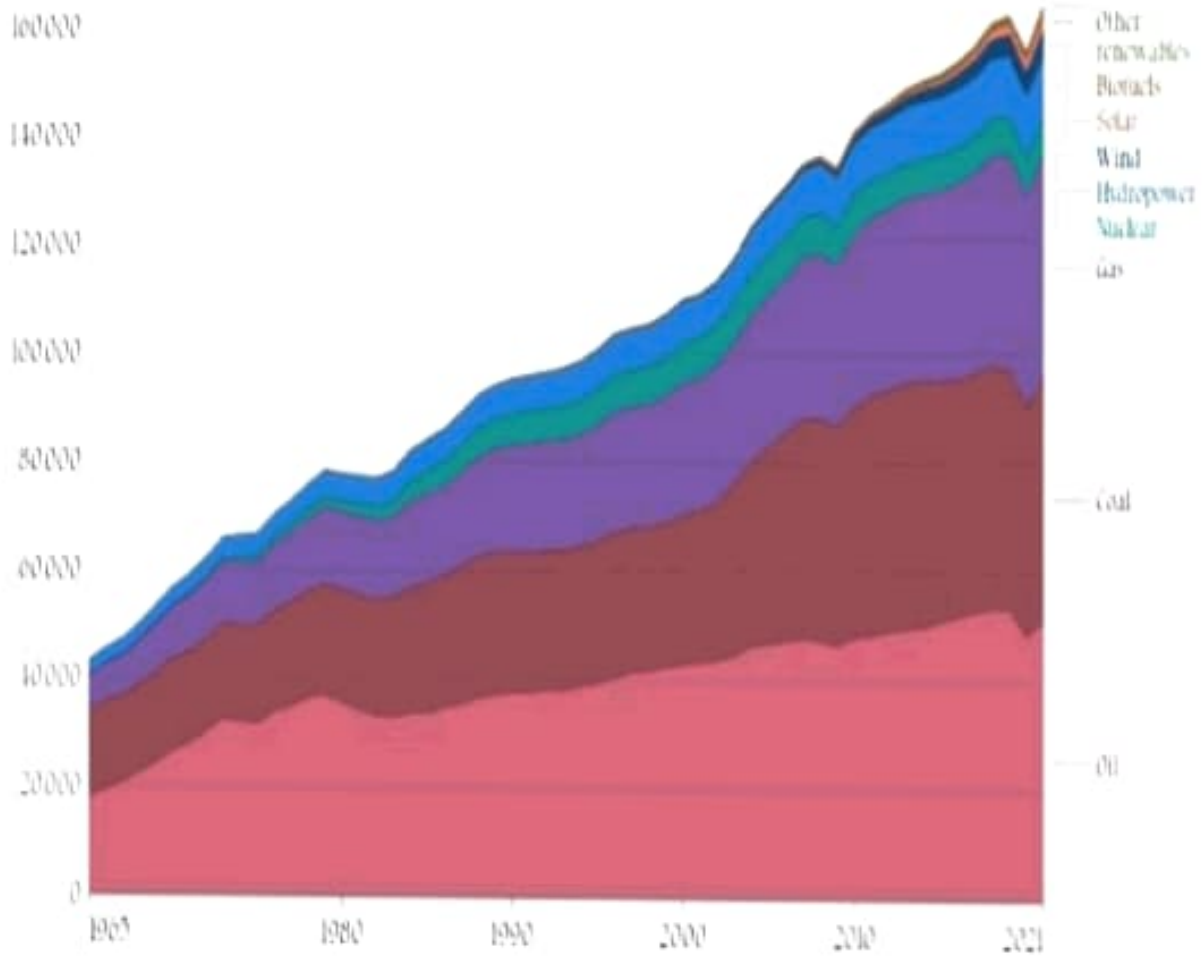
A. Burn CPU

The dataset "Burn CPU" consists of 2074 complete data records (without N/A values). Both datasets "Burn CPU" and "Mem Loop" are similar in the sense that power can be explained by using one variable only. Table I shows the best variables explaining the power, with outliers taken into account, and when removing outliers. It must be noted that it is not clear why e.g. host_df_df.root_free_SQ yields such good results. There are some other host related variables also yielding high correlations. However, for instance the variables perf_instructions and perf_cycles do make sense

Energy consumption by source, World

Our World in Data

Primary energy consumption is measured in terawatt-hours (TWh). Here an inefficiency factor (the substitution method) has been applied for fossil fuels meaning the shares by each energy source give a better approximation of final energy consumption.



Source: BP Statistical Review of World Energy

Note: Other renewables includes geothermal, biomass and waste energy.

OurWorldInData.org/energy • CC BY

n a pure CPU stress test and also well explain the power consumption.

Variable	R^2	σ	%	#Out
host_df_df.root_free_SQ	0.992	0.216	0.289	0
	0.996	0.146	0.195	12
host_df_df.root_used	0.992	0.216	0.289	0
	0.996	0.146	0.195	12
perf_instructions	0.991	0.232	0.310	0
	0.996	0.143	0.191	12
perf_cycles	0.991	0.233	0.310	0
	0.997	0.144	0.192	12

Table I
BURN CPU.

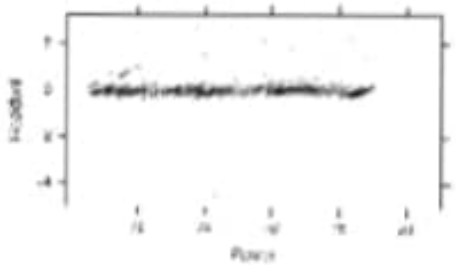


Figure 2. Burn CPU residuals for variable host_df_df.root_free_SQ.

Figure 2 shows the model residuals when using a model with only the variable host_df_df.root_free_SQ as single explanatory variable. There are only a few outliers, nevertheless influencing σ significantly.

B. Memory Loop

The dataset "Mem Loop" consists of 2017 complete data records (without N/A values). Table II shows the best variables explaining the power.

Variable	R^2	σ	%	#Out
perf_context.switches	0.998	0.268	0.339	0
	0.999	0.213	0.270	18
perf_cache.references	0.998	0.285	0.361	0
	0.999	0.196	0.248	25
perf_context.switches_SQ	0.977	0.919	1.163	0

Table II
MEMORY LOOP.

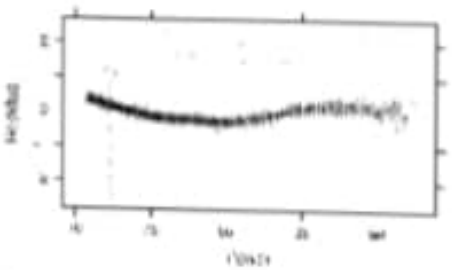


Figure 3. Loop memory residuals for variable perf_context.switches.

Figure 3 shows the model residuals when using a model with only the variable perf_context.switches as single explanatory variable. Again a few outliers are visible, which however do not significantly influence the results. Also, it is obvious that the residui do exhibit a deterministic component. In fact this deterministic dependence could be exploited by further refining the regression model, e.g., by adding a sin()-like function. It turns out that this is not really necessary since the one-variable model is already good enough such that any model extension is superfluous.

C. Network

The dataset "Network" consists of 1937 complete data records (without N/A values). Table II shows the best variables explaining the power.

Variable	R^2	σ	%	#Out
host_interface_if_octets.eth1_tx	0.954	0.178	0.247	0
	0.983	0.106	0.147	13
host_interface_if_packets.eth1_rx	0.953	0.181	0.251	0
	0.980	0.117	0.163	13
host_interface_if_packets.eth1_tx	0.951	0.183	0.255	0
	0.981	0.114	0.159	13
perf_cycles +	0.967	0.152	0.211	0
host_interface_if_packets.eth1_tx	0.991	0.079	0.110	16
perf_instructions +	0.967	0.152	0.211	0
host_interface_if_packets.eth1_tx	0.991	0.079	0.110	16
perf_cycles +	0.967	0.152	0.211	0
host_interface_if_octets.eth1_tx	0.991	0.077	0.108	17

Table III
NETWORK.

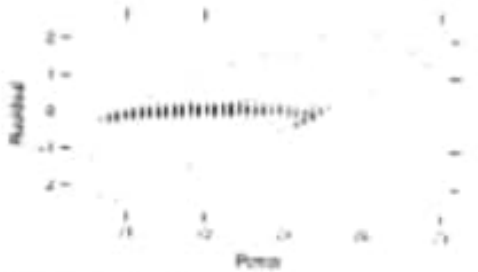


Figure 4. Network residuals for variable host_interface_if_octets.eth1_tx.

Figure 4 shows the model residuals when using a model with only the variable host_interface_if_octets.eth1_tx as single explanatory variable. This variable makes sense when keeping in mind that the workload mainly stresses the network card. Furthermore, Table III and Figure 4 show that although two variables are enough to explain the power consumption, there are a few outliers that do have a significant influence in the result. Removing only 13 out of 1937 data records increases R^2 to 0.98 for one, and 0.99 for two variables.

D. Tar Kernel

This dataset contains only 18 complete data records. Inasmuch it was not possible to reduce the number of

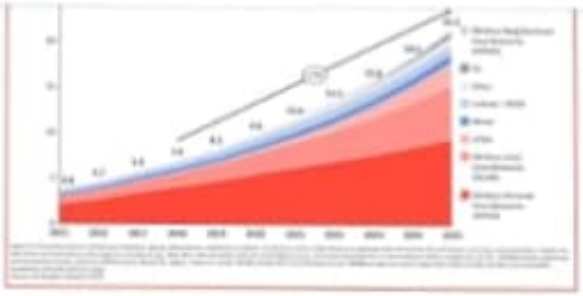


FIGURE 1. Global number of connected IoT devices1. It is predicted that the number of globally connected IoT devices will be 2.5 billion by 2025.

power analysis at higher hardware abstraction levels, such as register transfer level [9]. Further up is measuring or modeling at the level of hardware components, such as processors, memory, disks, and peripheral circuits, where many works have been done [10], [11].

On the other hand, as the driver of hardware executing, software instructions indirectly consume energy, which means the energy consumption can be mapped to software structure. The granularities of the mapping includes assembly instructions, functions, and software components [12], [13]. The study of energy consumption measurement of embedded systems provides traceable energy behavior information at various granularity levels, which is not only for evaluating the energy consumption of software/hardware, but also improve the developers' cognition of software energy characteristic for optimization. Besides, the energy measurement of embedded systems can be used to recognize rogue software since the malicious program execution can lead to additional energy consumption [14]–[16].

However, diverse research emphases and multi-levels energy consumption measurement schemes lead to the complexity of this field for researchers. We aim to provide a detailed analysis of the latest energy measurement techniques in the current background. Based on the analysis, we put forward the defects of the existing methods and future research directions. The main contributions of this work:

1) This paper proposes a novel and thematic taxonomy for classifying the existing works for energy consumption measurement. They are categorized into: a) methods of measuring and profiling, b) model-based energy estimation schemes, and c) simulator-based energy consumption estimation approaches.
2) We make a comprehensive comparison of these available methods in multiple metrics, including the power data source, measuring or modeling object, modeling methodology, granularity, and error rate. The capabilities and shortcomings within each category are analyzed.

The rest of this paper is organized as follows. Section II introduces the first kind of energy consumption measurement method, which is the combination of general instrument measurement and data analysis. The model-based energy consumption estimation schemes are described in Section IV. Section V presents software simulation-based energy consumption estimation approaches. Section VI discusses the challenges and future works in this field. And Section VII we concludes the paper.

II. THE TAXONOMY OF ENERGY MEASUREMENT METHODS

When investigating the existing literature about energy consumption measurement techniques of battery-powered embedded devices, it has been found that these available methods can be classified from multiple perspectives. For example, in literature [17], power modeling schemes of embedded systems are categorized into transistor or switch, logic gate, register transfer, and system level, according to different stages in design flow. To map energy estimation techniques to software applications, literature [18], [19] categorize the existing energy mesurment methods into code-analysis and mobile components power model based schems, which is based on the hierarchy of embedded systems. In terms of whether the energy consumption profiler is online or not, the existing methods could be classified into online and offline [20].

Through the analysis of above papers, it can be found that researchers choose the corresponding taxonomy according to the research purpose, and it is difficult to make comprehensive comparisons according to one single metric. Therefore, from the principle of measuring and modeling for embedded systems, this paper classifies existing methods into three categories: measurement-based energy profiling, model-based, and simulator-based energy estimation schemes, which is shown in Fig. 2. In the measurement-based energy consumption profiling methods, due to different tools for obtaining power data, there are two methods: instrument-based and platform-based. In the method of model-based energy estimation, many papers have obvious classifiable characteristics in the information types required for modeling. Therefore such methods can be categorized into hardware utilization based, system-call based, and program-analysis based modeling.

Secondly, after extensively investigate and summarize the characteristics of existing methods, we compare these techniques under each category with multiple metrics to achieve a comprehensive understanding. The comparison metrics we used include granularity, power data source, measuring or modeling object, modeling approach, overhead, and error rate. The granularity refers to the level of energy consumption analysis. For example, the granularity at hardware level includes transistor, logic, and component levels, and it a

FIGURE 4. The framework of PowerScope. The energy analyzer obtains PC/PID samples and current data from the system monitor and the energy monitor, respectively, and then combines the symbol table for energy consumption analysis.

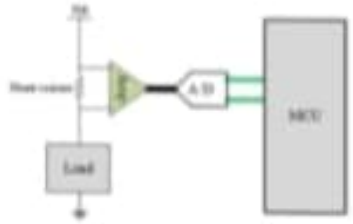


FIGURE 5. The typical system architecture of power measurement platform.

set corresponding parameters when they use PowerScope to estimate the energy cost for application, which will cause additional workload for developers.

Although the above methods can accurately measure the total energy consumption for overall mobile devices, the fine-grained power measurement is powerless. Due to hardware circuit conversion and other hardware call effects when application operation, it will unevenly attribute the consumed energy. To solve these issues, researchers adopt the approaches which use probes of instruments inserting the hardware circut to measure the interested component objects. For example, in [10], aiming at evaluating the energy usage of mobile devices, researchers proposed an energy measurement method for both the overall system and main hardware components, afterwards, energy models are built for the device under some usage scenario. In this method, a mobile device which can obtain circuit schematics is used because they need to set placeholders in power supply rails for each target hardware component. Then, sense resistors which with high precision and little resistance are inserted in these placeholders. DAQ card is used to collect the voltage drop data across these resistors directly. In this manner, energy consumption can be measured accurately.

Another component-level energy consumption analysis without modifying the hardware circuit is introduced in [24]. Aiming at improving the energy-efficiency for heart rate monitoring of smartwatches, researchers analyzed the energy consumption of specific hardware components such as acceleration sensor, screen, Bluetooth, and heart rate monitoring components. In this article, PCI-6230 is used to obtain the voltage drop over the shunt resistor at a certain frequency. They stress each component running separately for getting run-time power data, then the power consumption of each component is obtained by subtracting the energy when the watch is idle. However, DAQ cards or other instruments used in the above work are expensive, especially those with multiple probes. Also, in order to obtain the power data of hardware components, hardware circuit modification is needed, which is hard to implement for users.

2) Platform-based measurement

In addition to using general instruments to measure current and voltage, other researchers have designed power measurement platforms to replace instruments, and mainly to solve the problems with instruments such as expensive, large volume, and difficulty in flexibly establishing measurement experiments.

The typical system architecture of this kind of power measurement platform is shown in Fig. 5, a shunt resistor is put in series between the power supply and the load. Following a current sense amplifier or differential amplifier is used to amplify the small voltage drop across the shunt resistor and then provide it to the analog-to-digital converter (ADC). The ADC module is used to convert the analog signal into a digital signal and send it to the MCU, which is responsible for collecting the current data and sending it to the upper computer or directly processing it locally to obtain the load power consumption.

For instance, the researchers in [21] had developed a prototype board based on micro-controller used for measuring the current, but not used in the experiment in the paper. In [25], a testbed FlockLab was developed to reveal the system behavior into wireless embedded systems. The services provide by FlockLab include logic analysis, power profiling, adjusting power supply, and serial I/O logger. In terms of power profiling, it uses a small shunt resistor to sense the current density from battery to target device. A 24-bit delta-sigma ADC with sample rate of 28kHz is used to sample the voltage across the shunt resistor, which amplified by a current sense amplifier. PowerBench [26] is a testbed used to record the power traces of several sensor nodes in parallel. The current measurement method adopted in PowerBench is similar to that in [25]. However, its current measurement range is only from 0-60 mA. Meanwhile, the measurement range of FlockLab is from 2 μA to 160 mA, which is not suitable for high current devices so that limiting the scope of use.

Most of the previous works use a single shunt resistor's voltage drop to catch the current of the device, which leads to a small measurement range [25]-[27]. This kind of measurement platform is hard to meet the requirements of the modern embedded device since it may have high peak currents [28]. For that reason, some researches carried out to extend the measurement range of power consumption by adaptively adjusting the resistance of the shunt resistor. In Nemo [29], five resistors (0.1 Ohm, 1 Ohm, 10 Ohm, 100 Ohm, 470 Ohm) and four MOSFETs are used to compose a so-called

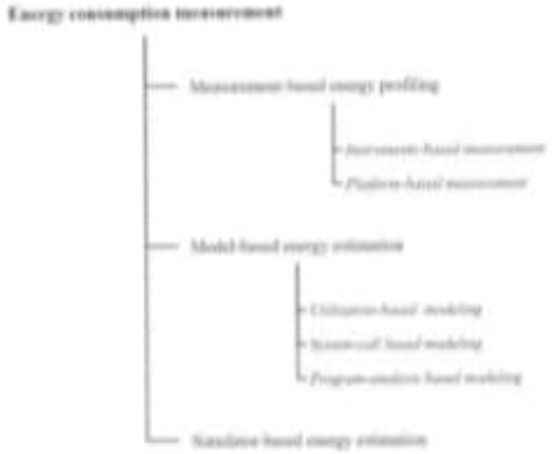


FIGURE 2. The category of energy consumption measurement methods.

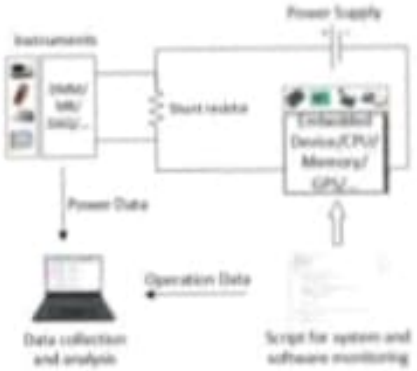


FIGURE 3. The basic framework of power measurement. A high precision shunt resistor is connected between the power supply module and the device. The probes of instruments (e.g., DMM, MR, DAQ) are attached to both sides of the resistor and sampling the voltage across it. The server obtains the corresponding data for energy consumption analysis.

the software level includes assembly instruction, source code line, process, function, and application levels. The power data source represents where the device power is acquired. The measurement or modeling object identifies the described targets for power consumption, including instruction, function, phone feature, component, and system. The modeling methods include linear and second-order polynomial regression model, piecewise constant model and finite state machine model. Based on these detailed comparisons, the pros and cons of existing methods can be summarized.

The more detailed analysis for each class of approach is given in Section III, IV and V, respectively.

III. MEASUREMENT-BASED ENERGY PROFILING

A. CATEGORY

1) Instrument-based measurement

In embedded systems, software drives hardware, and the energy is consumed in the operation process of the hardware circuit. Therefore, it is an intuitive method to measure the system energy consumption on the hardware circuit by using instruments.

Fig. 3 shows the basic framework of power measurement approach. Generally, a high precision shunt resistor is connected between the battery or power supply module and the device, which the device can be the whole embedded device or the hardware modules (e.g., CPU, memory and GPS module). The probes of instruments are attached to both sides of the resistor and sampling the voltage across it at a certain frequency. Consequently, real-time voltage can be acquired when the system is running, and the corresponding current and power consumption are calculated by Ohm's law and power formula. The measuring instruments include high precision Digital Multimeter (DMM), Memory Recorder (MR), Oscilloscope, and Data Acquisition (DAQ) card. At the same time, scripts are used to monitor the detailed execution procedures of software and operating system (OS) in the embedded system. Finally, the energy consumption can be cooperative analyzed based on the power and the software operating data.

The mainstream literature which adopts instrument-based methods are shown in Table 1, these studies measure the energy for the whole system. Then the energy consumption is attributed to corresponding applications according to the running period. For example, in literature [21], a method for measuring the energy consumption of mobile application program is introduced. The power trace of the device is characterized by the framework in Fig. 3. In [22], aiming at reducing energy consumption of the embedded devices which employ P2P protocols, an offline method is proposed for measuring the energy consumption. Similar to [21], high precision resistor with small resistance and DAQ card are used to collect voltage data at a fixed frequency. The application's operation information is recorded by accessing the specific GPIO port in the hardware device. Moreover, a device driver is developed to record threads and functions information for function-level energy consumption analysis, thereby the problem of energy consumption attribution could be solved.

In [23], a tool PowerScope was developed for estimating energy consumption of applications in mobile devices. As shown in Fig. 4, PowerScope is a typical framework that includes three software modules: the system monitor, the energy monitor, and the energy analyzer which are responsible for collecting system activity data, current data, and estimating energy consumption, respectively. An important feature of PowerScope is that it can map energy consumption to corresponding program structure, and this capability is dependent on symbol tables. However, energy consumption cannot analyze online since it is a post-processing tool. In practice, PowerScope needs a set of modifications to the kernel. The developers need to call a series of APIs and

Presently, there are plenty of research works assessing the energy consumption and environmental impacts of buildings, but few encompass construction process in complete life cycle. Some studies have included the construction phase; however, this was limited to various stages of material extraction, production, and transportation and did not include on-site construction processes [7-9]. The industry's energy consumption during construction is not well understood because of its fragmentized nature and involvement of many parties during construction phase [2]. That is why, at the time of design and even before construction starts, it is hard to predict the energy required and its impact at the construction phase. Often researchers exclude the construction phase, stating that its contribution towards total life cycle energy consumption is insignificant. Researchers have also stated that energy consumption and environmental impact of construction activities have never been adequately quantified [2, 10, 11]. European and U.S. figures estimate the construction portion to be about 7-10% of total embodied energy [12, 13]. In the near future when low/net-zero energy buildings are more common, the embodied energy and, hence, construction energy, will gain more importance to achieve sustainable construction [14]. Figure 1 shows the change in ration of operational energy to embodied energy when building development moves from traditional buildings to low energy buildings. Embodied energy in the built environment, especially when buildings have short lives or when buildings are extremely energy efficient, will share an important portion in total energy consumption [15].

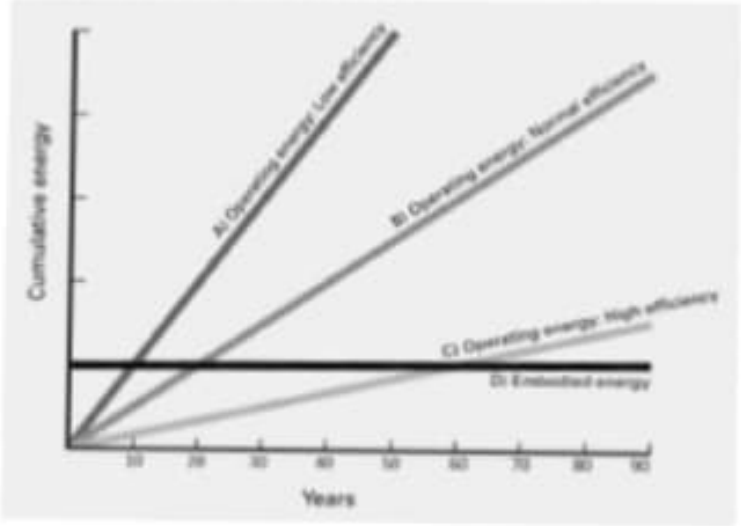


Fig. 1 Energy use in buildings: the changing relationship between embodied and operational energy [15]

2.1 Construction contractors and energy consumption during construction

The stakeholders for a building project include Investors, developers/owners, architects/engineers, and contractors. In general, selection of the site for a project is done by the owner, and designs and selection of materials are the responsibility of architects/engineers. Contractors are hired mainly to construct the building according to the construction documents within budget and on time. Selection of means and methods of construction and reducing the environmental impacts of the construction process is the sole responsibility of contactors.

Rising energy costs is also a concern to contractors. Arnold [16] studied several building projects and found that energy costs during construction vary and can be a significant part of the construction operation costs (as high as 5.7%). He also mentioned that it is difficult to measure energy costs because these costs are embedded in the materials, equipment, or overhead costs. The cost of energy affects the cost of construction, and dependence on fossil fuels makes

The construction industry uses more materials by weight than any other industry in the United States [1]. Whenever a building is constructed, it imposes loads on the environment in various forms, namely: resources depletion and contamination of air, soil and water, etc. These loads are generated while various demands, such as materials and energy, are met to furnish the designed building. The environmental impacts of building construction, partly caused by large consumption of energy, are imposed during the whole life cycle of a building [2, 3].

Typically, the life cycle of a building can be divided into the following phases: extraction of required raw materials; processing and manufacturing of construction materials and building components; transportation and installation of building materials and components; operation, maintenance, and repair of building; and, finally, disposal of materials at the end of the building lifecycle. Each phase demands energy, material and other resources to produce the required input for a successive phase to complete the cycle. In fact, each of these phases includes several sub-cycles to complete that particular phase. The accumulated consumption and impacts of these sub-cycles provide a comprehensive result for that phase. These results may vary from study to study depending on the various sub-cycles included and the boundaries of an analysis. Setting the boundaries and inclusion of the sub-cycles depend on the parameters being analyzed and/or on the importance of a sub-cycle for a particular study. Each phase of the life cycle of a building affects the environment. Therefore, each phase must be studied in search of providing a high performance energy efficient building.

Although conclusions about energy and environmental performance of a building should be based on the results of a whole life cycle analysis, there are always opportunities to explore each phase individually to collect detailed information and strengthen information databases. The collected data would be of interest to the stakeholers who strive to build environmentally friendly buildings. Changes in construction, along with changes in design and selection of building materials, are essential to the success of national efforts to minimize environmental impacts and reduce overall energy use and greenhouse gas emissions [4]. In a construction project, contractors provide resources and select the means and methods of construction. To maximize energy efficiency on a project by using the least energy intensive means and methods of construction, the energy profile for a project is needed. This is possible if the contractor, during pre-construction planning, has access to information regarding energy consumption during construction of the project to identify activities that consume more energy. The obtained information would help contactors focus on energy intensive activities and develop energy efficient means and methods to minimize energy consumption during construction.

2. Construction industry and energy consumption

The U.S. construction industry accounted for $611 billion or 4.4 percent of the nation's GDP in 2008 [5], which would increase to 10% if the equipment, furnishings, and energy required to complete buildings were included [4]. According to the U.S. Department of Energy [6], U.S. buildings - residential and commercial - consume around 40% of the total energy consumption.

8. Conclusions

Machine learning algorithms consume significant amounts of energy. However, the lack of evaluations based on energy consumption of these algorithms can be attributed to the lack of appropriate tools to measure and build power models in existing machine learning suites, and because estimating energy consumption is a challenging task.

This paper addresses that challenge by presenting a review of the key approaches to estimate energy consumption from the computer architecture field, mapped to machine learning applications. We also describe the state-of-the-art methods to estimate energy consumption in particular for data mining and convolutional neural networks. Our synthesis of the surveyed papers provides the necessary guidelines to expose energy consumption methods to machine learning audiences