IrEne^{+}: A cross-platform extension of IrEne

Tom Rooney

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

Accurate measurement of the energy consumed by machine learning processes, across a diverse range of hardware platforms, is necessary to facilitate a reduction in the exponentially growing energy demands of models [1]. However, many existing methods fail to simultaneously meet these criteria. Those that do are either incapable of estimating energy at the low temporal resolutions required in the machine learning domain or are extremely impractical.

In this project, we extended the state-of-the-art approach to estimating the energy consumed by model inferences, IrEne [7], to satisfy all the aforementioned requirements. Our overhauls to the training process within this prediction architecture ensure easy compatibility with a diverse set of hardware platforms, through the use of widely available software interfaces. Furthermore, we reduced the overhead of setting up IrEne on a new machine by over 50%. All of this is achieved with no negative impact on the accuracy of energy estimation, as we replaced IrEne's tree-based prediction approach with a simpler, yet more powerful, multiple regression model. In fact, we observe an average reduction of 2% in the observed error in predictions made. Our resulting architecture is subsequently named, IrEne^{+}.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Tom Rooney*)

Acknowledgements

I'd like to thank my supervisor, Mahesh Marina, for his guidance and enthusiasm, particularly in some of this project's more difficult moments. Special thanks must also go to Leyang Xue for always being available to answer my questions, no matter how silly, and for assisting with the technical side of the project. This project would not have been possible without either of these individuals.

Table of Contents

Chapter 1

Introduction

1.1 Motivation and objectives

The compute and energy demanded by modern Machine Learning models has been growing exponentially, doubling every 6-10 months [1, 28]. This trend facilitates the continued progress observed in the state-of-the-art capabilities of Artificial Intelligence [29]. However, the increased energy consumption introduces a significant financial and environmental cost to train and deploy models. Strubell et al have argued that this raises barriers to entry into machine learning research, so the development of more efficient algorithms and hardware systems must be prioritised [30]. Therefore, to facilitate such work, means of accurately measuring the energy consumption of machine learning processes are necessary.

In addition to providing accurate results, energy measurement should be cross-platform and fine-grained. Cross-platform solutions are those compatible with many heterogeneous systems, without a lot of overhead. Measurement systems meeting this constraint are highly accessible and more likely to see widespread adoption. Fine-grained solutions are not only capable of accurate energy measurement but can do so at very low temporal resolutions. Consequently, permit energy analysis of a model architecture's constituent layers, providing insight to enable the identification of energy bottlenecks [7].

Simultaneously providing accurate, cross-platform, and fine-grained measurement is a challenging problem. Floating Point Operations (FPO) quantify the work performed by a computational process, so have been proposed as a metric to indicate the energy efficiency of models [27]. However, FPO provides little useful information when drawing comparisons across different model architectures and machines [18]. Most other existing works estimate energy consumption as a linear sum of the energy consumed by each of the CPU, GPU, and DRAM [30, 18, 2]. The problem here is that the granularity of measurement is too high, limited by the rate at which the underlying, low-level software interfaces providing component-wise measurement are updated. Finally, the state-of-the-art, IrEne, provides accurate and fine-grained measurements using a model tree abstraction [7]. However, training this technique on new systems requires the use of cumbersome hardware power monitors, so it does not generalise easily to new platforms.

The main goal of this project was to solve this problem and fill the gap in the existing literature by modifying the design of IrEne to facilitate cross-platform measurement. The sub-goals forming this primary objective were as follows:

- To replace the impractical use of a hardware power monitor to obtain training data with a software-based approach, compatible with as many heterogeneous systems as possible.
- To reduce the significant compute time required to obtain this data and therefore, to train an IrEne instance.
- To ensure that the changes made do not have an adverse impact on the accuracy of measurement.
- To ensure that accurate measurement is still possible at a very fine temporal granularity.

1.2 Report Structure and contributions

The main contributions made as a result of this project are as follows:

- Identification of a major flaw in the evaluation of IrEne (§3.3)
- Subseqneut replacement of the complex, tree-based prediction approach of IrEne with one multiple regression model (§3.3).
- Identification and removal of a feature within the IrEne architecture that cannot be accurately profiled (§3.4.1).
- Identification and removal of a further nine features used by IrEne which provide little useful information in estimating energy consumption (§3.4.2).
- Replacement of the hardware monitor, required to obtain data to train IrEne, with a software-based approach constructed using widely available hardware component interfaces. (§3.5.1).
- A reduction of over 50% in the overhead required to train to obtain each training sample. (§3.5.2).
- Evaluation of the impact of these design changes on the accuracy and granularity of energy measurement, demonstrating no adverse effect.

The remainder of the report is organised into four further chapters. Chapter two provides a detailed discussion of existing approaches to measuring the energy consumed by machine learning processes, assessing them against the accuracy, granularity, and cross-platform criteria. The major changes we make with respect to the IrEne architecture to facilitate cross-platform measurement are motivated and detailed in Chapter three. Then, in Chapter four, we individually evaluate each of these changes and our estimation architecture as a whole, showing that it appears to be more accurate than IrEne. Finally, a summary of our achievements and a discussion of both the limitations of this work, and resulting future directions, are provided in Chapter five.

Chapter 2

Existing energy measurement approaches

Here we present a detailed discussion of existing energy measurement strategies. For each approach, we provide a brief overview and an assessment against the accurate, cross-platform, and fine-grained criteria. First, we focus on methods agnostic to the domain of machine learning, before moving on to those which are more specific and designed with training or inference in mind.

2.1 Machine learning-agnostic approaches

2.1.1 Hardware Power Monitors

Deriving energy directly from power data provided by a hardware sensor external to the system under test is generally considered to be the 'gold standard' in energy consumption measurement, regardless of domain [5]. Although the capabilities of hardware power monitors vary, many can make highly accurate measurements at a very fine granularity. For example, the PowerInsight and PowerMon2 tools have sampling rates of 1KHz and 3KHz respectively, with both devices producing measurements with an error of $<6\%$ [21, 4, 10].

While all hardware power monitors are technically cross-platform, we do not generally classify them as such. The most capable tools, including those previously mentioned, require physical access to the system under test to make connections on the rails between the power supply and motherboard [5]. Consequently, cross-platform measurement is extremely cumbersome or even infeasible. Simpler power monitors requiring only a connection between a device and its wall power source exist as alternatives. However, these boast insufficient sampling rates of 10 to 100 Hz [5].

2.1.2 Low-level Software Monitors

In recent years, hardware component manufacturers have introduced built-in sensors to their products, which can be queried via software interfaces to provide power data.

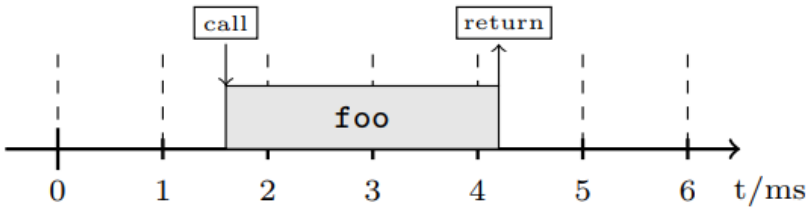


Figure 2.1: Taken from [15]. An example of the inaccuracy introduced when measuring the energy consumed by short code paths using RAPL. On calling *foo*, the value read equates to the total energy consumed up to and including time 1 ms. However, this does not reflect the true initial energy. A similar issue arises when *foo* returns.

For example, Intel provides the Running Average Power Limit (RAPL) API for its CPUs, while Nvidia GPUs can be monitored using the Nvidia Management Library (NVML) [9, 23]. Although these software monitors are widely similar in function and limitations, there are some subtle differences and so we discuss each independently.

2.1.2.1 Running Average Power Limit (RAPL)

Measurement of energy consumed by the CPU[1] in executing a program, or part of it, using RAPL is straightforward. The value it provides represents the total processor energy consumption since system power-on (J). Therefore, at both the beginning and end of execution, we query the interface yielding two values, e_{start} and e_{end}. Then, provided the process under test runs in isolation, the total energy consumed during execution simply evaluates as

$$e_{\text{total}} = e_{\text{end}} - e_{\text{start}} \tag{2.1}$$

Furthermore, RAPL is accurate and cross-platform. When placing a processor under different loads, a very strong correlation exists between RAPL measurements and wall-socket power readings taken using a hardware power meter [14, 19]. The observed power values differ only in magnitude due to the scope of their measurement i.e., CPU vs full-system. Furthermore, any Intel CPU released since the inception of the Sandy Bridge architecture (Jan 2011) supports RAPL. The vast number of processors supported easily classifies the interface as cross-platform.

Unfortunately, energy estimation using the typical approach previously described is not fine-grained. The value reported by interface queries is updated every millisecond, with this period called the *sampling interval* [9, 19]. Furthermore, the execution of the code under test can begin at any point in this interval and is likely to do so. Therefore, the values obtained by the two RAPL reads are probably stale (Figure 2.1). For sections of code with sufficient run-time relative to the sampling interval, this error is amortised. However, fine-grained measurement encapsulates periods both on the order of this

[1] RAPL can also be used to measure DRAM energy consumption using the same process described here.

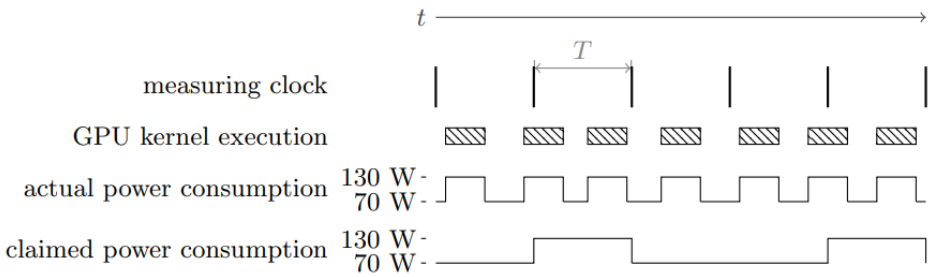


Figure 2.2: Taken from [20]. Demonstration of issues arising when measuring fine-grained GPU energy consumption using an NVML-based approach. *T* represents the sampling interval, while the GPU kernels represent the fine-grained code paths we are trying to measure. Here, we see examples of the worst-case scenario where the entire execution time falls within a sampling interval. Note how for many kernels, their sections of the claimed and actual power consumption lines are vastly different.

period and smaller. In these cases, significant underestimation or overestimation occurs, depending on precisely how execution aligns with RAPL updates [15].

2.1.2.2 Nivida Management Library (NVML)

Similar to RAPL, NVML provides accurate and cross-platform means of measuring the energy consumed by a GPU in running sections of code. Reporting instantaneous power draw (mW), the interface is queried repeatedly during execution and the readings recorded. Then, total energy consumption is evaluated according to Equation 2.2. Again, the process under test must be the only one running. Regarding accuracy, Nvidia reports that each query falls within ± 5 W. Similarly, Arafa et al find that this method of deriving GPU energy consumption yields an error of $<7\%$ relative to a hardware power monitor [3]. Finally, the vast majority of Nvidia GPUs built on the Kepler architecture or newer support NVML-based energy measurement, making the approach cross-platform.

$$E(mJ) = \frac{Elapsed\ time\ (sec)}{\#\ power\ readings\ (N)} \times \sum_{i=1}^{N} Power_i\ (mW) \tag{2.2}$$

However, NVML also fails to provide fine-grained energy measurement, with a sampling interval of 20 ms. Derivation of energy consumption as described previously requires constant sampling of GPU power consumption. Furthermore, as with RAPL, code execution can start and end anywhere within a sampling interval. Therefore, samples near the beginning and end of run-time often differ vastly from the true power draw, as visualised in figure 2.2. Naturally, the shorter the execution time, the larger the portion of erroneous samples and thus the higher the error. So, like RAPL, NVML is only capable of accurately measuring the energy consumption of code paths with run-time at least on the order of its sampling interval. Clearly, this is insufficient to provide fine-grained measurement as we later define it.

Furthermore, the resolution of measurement can be even more coarse due to power

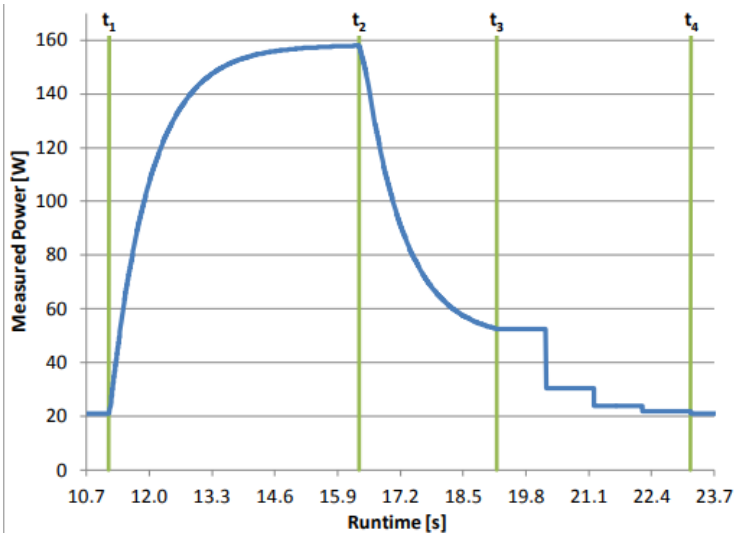


Figure 2.3: Taken from [6]. Illustrates what a power lag looks like. Execution begins at *t_1* and stops at *t_2*. However, the power consumption reported by NVML takes a few seconds to adjust, despite the fact that the true power consumption of the GPU changes instantaneously.

lags [6]. The values reported by NVML are derived from a GPU's onboard power sensors. However, on some models of GPU, these sensors lag behind the true power draw associated with program execution. Consequently, values reported by NVML do the same. As with the sampling interval, these lags cause a portion of samples at the beginning of execution to be incorrect, introducing significant error when deriving energy according to Equation 2.2. However, as seen in Figure 2.3, power lags are much longer in duration than NVML's sampling interval. Therefore, much more execution time is required to amortise their associated error and so, where present, they become the dominant force limiting granularity.

It should be noted that not every Nvidia GPU exhibits these effects. For example, the Tesla K20x has been found to measure true instant power [6]. Similarly, different GPUs can exhibit different lags of different lengths. Both of these observations are explained by the fact that power lags are derived from hardware sensors specific to each model of GPU, not NVML. Despite this, we should still be aware of the existence of GPU power lags due to their significant effect on measurement granularity where they exist.

2.2 Machine learning-specific approaches

Having considered general approaches to energy consumption measurement applicable to any domain, we now consider works whose focus is on machine learning processes. An understanding of all domain-agnostic techniques discussed §2.1 is assumed here, as they form the basis for many of these domain-specific approaches.

2.2.1 Floating Point Operations (FPO)

Proposed as a proxy for estimating energy consumption, FPO provides fine-grained measurement. For a given model operation e.g., an inference, the associated FPO constant quantifies the work done in executing it. Consequently, it has been recommended as a measure of machine learning energy consumption, with the intuition being that less work done demands less energy [27]. The fine-grained nature of an FPO-based approach stems from its recursive derivation. First, the cost of two base operations ADD and MUL are defined. Then, the operation under test is expressed as a recursive function of these two base cases, from which FPO count is calculated [27]. Operations at all levels of a given model architecture can be described in this way, so this approach is fine-grained.

With respect to cross-platform measurement, FPO falls short due to its hardware-agnostic nature. The constant reported depends only on the computational complexity of the target operation and is not affected by the hardware platform on which it is run [27]. However, real energy consumption is often impacted by a variety of system-dependent factors including, firmware optimisations, number of memory accesses, and hardware efficiency [18]. FPO cannot account for these effects and so we do not classify it as cross-platform.

Similarly, an FPO-based approach is inaccurate as different model architectures utilise hardware resources in different ways. While it is true that FPO cost accurately indicates relative energy cost within the same model [18], this does not generalise. Across distinct model architectures, it is common to observe the same FPO value being shared for the same operation. However, the corresponding true energy consumption varies significantly. Figure 2.4 illustrates several examples of this. In particular, the pairs (densenet201, hardnet68) and (densenet161, vgg111). Given the limited context in which FPO is a good estimate of energy consumption, we do not deem it to be generally accurate.

2.2.2 Fully Defined Linear Model (FDLM)

Frequently, in domain-specific literature, energy consumption is derived using a fully defined linear model. Initially introduced by Strubell et al , the same function (Equation 2.3) lies at the heart of several machine learning energy estimation tools [30, 2, 18]. The component-wise energy values e.g., e_{gpu}, are measured at run-time using the low-level software monitors described in §2.1.2. Then, the sum of these terms is scaled by a Power Usage Effectiveness (PUE) factor. The inclusion of the PUE term aims to account for the vast energy consumption required to provide cooling infrastructure etc. in data centers, as most of this literature focuses on measurement in these facilities. This parameter is fixed, not learned from training data, hence we refer to this approach as fully defined. In summary, total energy is evaluated as:

$$e_{\text{total}} = PUE(e_{\text{gpu}} + e_{\text{cpu}} + e_{\text{dram}}) \tag{2.3}$$

While tools based on this model provide cross-platform measurement, granularity

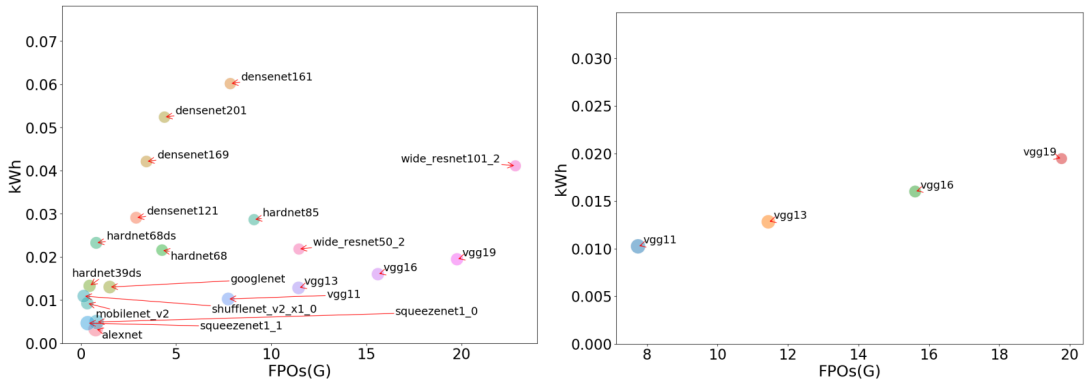


Figure 2.4: Taken from [18]. Demonstration of the limited accuracy of FPO-based measurement by visualising the relationship between FPOs (G) and inference energy consumption (kWh) across different model architectures (left) and within a single architecture (right) on the same machine. Within a single architecture, FPOs are strongly correlated with energy consumption (R^2 = 0.999, Pearson = 1.0). However, extending comparisons to include different architectures demonstrates a much weaker correlation (R^2 = 0.083, Pearson 0.289).

requirements are not satisfied. Recall that NVML and RAPL are each supported by a large number of Nvidia GPUs and Intel CPUs respectively (§2.1.2). Consequently, a significant portion of machines support energy measurement using these interfaces. So, any tool built atop these low-level monitors is cross-platform. Similar reasoning applies to the fine-grained criterion. We know that the temporal resolution of low-level software monitors is limited. Therefore, tools reliant solely on data supplied by these interfaces cannot themselves produce fine-grained results.

Given the accuracy of low-level software monitors, it is interesting that the fully-defined linear model is apparently not. We have seen how these interfaces provide accurate measurement, given sufficient run-time (§2.1.2). Therefore, we would expect Equation 2.3 to accurately predict full-system energy consumption under the same condition. However, one tool deploying this function was found to have an ”average error of over 55%” relative to hardware monitor measurements [18, 7].

Understanding the source of this error requires a deeper understanding of the role of the PUE parameter in Equation 2.3. In the context of a data center, let e_{total} be the total energy required by the entire center. Then,

$$e_{\text{total}} = e_{\text{fac}} + e_{\text{compute}} \tag{2.4}$$

where e_{fac} denotes the energy required by dedicated infrastructure to provide cooling etc. and e_{compute} represents the total consumption of all servers [31]. Suppose there are N servers within the facility. Then, each server is responsible for $\frac{e_{\text{fac}}}{N} J$ of the total infrastructure energy, despite not directly consuming it. The PUE term seeks to assign a particular server its ’share’ of e_{fac}, by appropriately scaling the measurement of its true energy consumption.

Given this, the high error observed is not surprising. The evaluation of this approach compares model predictions to hardware monitor measurements [7]. Now, we know that for server i the model estimates $\frac{e_{\text{fac}}}{N} + e_{\text{compute}_i}$, while the ground truth only accounts for e_{compute_i}. Therefore, we expect a high error given that different quantities are being compared. Furthermore, in this work a PUE of 1.58 was used, the global average for data centers [18, 30]. Recall that the average error observed was 55% [7]. Therefore, it could be reasoned that almost all of the error observed can be explained by the PUE constant.

We believe that a model without the PUE constant, defined by Equation 2.5 is more appropriate for our context. This work is concerned with the measurement of energy directly consumed by a given machine i.e., e_{compute_i}. Estimating amortised, total data center consumption i.e., $\frac{e_{\text{fac}}}{N}$ goes beyond our scope, so we believe the PUE to be irrelevant. Therefore, we suppose that estimating a system's energy consumption by providing data from low-level software monitors to Equation 2.5 should prove accurate. Further discussion of this proposal is provided in §3.5.1, and an evaluation demonstrating its accuracy can be found in §4.2.4.

$$e_{\text{total}} = e_{\text{gpu}} + e_{\text{cpu}} + e_{\text{dram}} \tag{2.5}$$

2.2.3 IrEne

The state-of-the-art approach, IrEne, predicts the energy consumed by a model inference using a model tree abstraction [7]. Nodes at each layer of the tree represent increasingly granular model components, with the model itself at the root and machine learning primitives such as LayerNorm, Tanh etc. at the leaves (Figure 2.5). First, the model under test is run once and its model tree is constructed. Simultaneously, four hardware-agnostic model features and eight hardware resource features, listed in Table 3.1, are profiled.

Then, the architecture predicts the energy consumed at each node of the tree. First, the energy consumed by each leaf is predicted using a separate, primitive-specific multiple regression model. Once complete, energy for every non-leaf node is evaluated recursively as a weighted sum of the predicted values of its children, using weights provided by one, shared multiple regression model. All linear models mentioned use the same 12 features measured at run-time. Finally, upon termination, the value associated with the root node is the amount of energy consumed by the entire model inference.

The process of training IrEne is fairly straightforward but comes with a few important notes. First, a wide range of (tree node, input size) pairs are executed, with the corresponding feature values and ground truth energy measured simultaneously. Note that obtaining the ground truth requires the use of a hardware power monitor. Then, IrEne's various constituent linear regression models are trained. It is important to understand that all the models in the training set should have similar architectures. For example, the presentation of IrEne focuses on transformer-based, natural language processing (NLP) models. However, once trained, accurate predictions can be made for any other model with a similar architecture i.e., prediction generalises to unseen

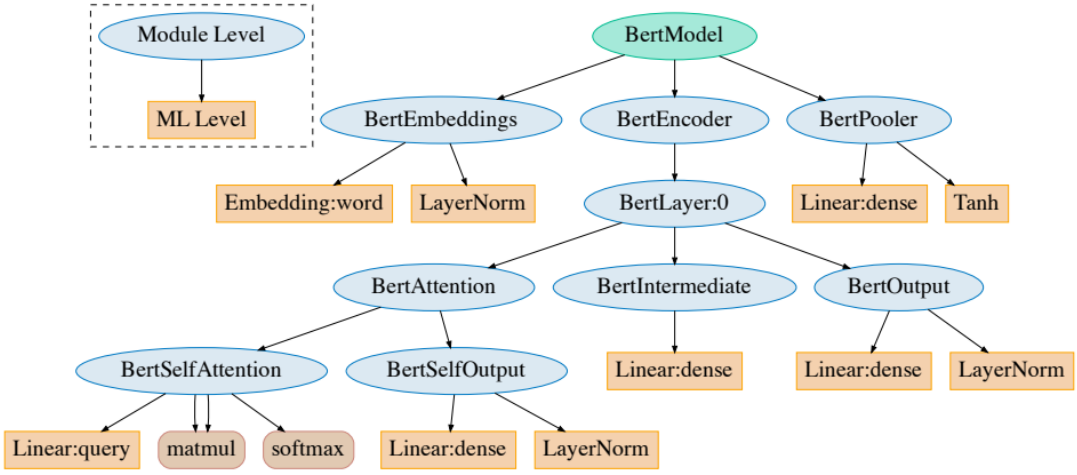


Figure 2.5: Taken from [7]. An example model tree, demonstrating the decomposition of BERT. At the leaves, we see model-agnostic, machine learning primitives. The intermediate, blue nodes represent module-level operations, specific to the model. Note how these can be further decomposed and reflect the high-level model architecture. Finally, at the root, we have the model itself.

models.

IrEne is highly accurate, but we find that similar results can be achieved in a simpler way. Evaluated against ground truth measured using a hardware monitor across different transformer models on two different devices, IrEne showed an average error of just 5-7% [7]. The authors claim that most of this performance comes from the use of the tree structure. However, in §4.2.1, we show that this is not the case, achieving improved error at all architectural levels without it. Similarly, they claim that ”resource features provide most of the benefits for energy estimation ... at all levels” [7]. Yet §3.4.2 highlights that many of these features are actually redundant and lead to unnecessary profiling overhead. Regardless, the model as presented makes accurate predictions.

Furthermore, although measurement is reportedly fine-grained we raise concerns as to whether this holds in practice. To evaluate the architecture, a data set is constructed by repeating the execution of model inferences for 20 seconds to ensure all variables are accurately profiled. However, IrEne is actually designed to run the model under test just once, measuring the metrics required to make predictions during this period. One model inference usually executes in time on the order of 10-30ms (Table 4.3), but this duration is insufficient to accurately profile all features. For example, *gpu_energy* is obtained via NVML which cannot take accurate measurements at this temporal resolution (§2.1.2). Therefore, as it relies on data from coarse, low-level software interfaces to make predictions, IrEne itself is not fine-grained.

Finally, we do not describe IrEne as providing cross-platform measurement. The key observation here is that the data on which an instance of IrEne is trained is tied to a particular hardware platform. Once trained on this data, predictions can be made only for that system. Creating a new IrEne instance for another machine requires the construction of a training set using a hardware power monitor. However, recall that

Method	Accurate	Fine-grained	Cross-platform
Hardware Power Monitors	✓✓	✓✓	✓✓
Low-level Software Monitors	✓✓		✓✓
FPO	✓	✓✓	
FDLM	✓		✓✓
IrEne	✓✓	✓	

Table 2.1: Summary of existing energy measurement approaches, both agnostic and specific to the domain of machine learning. The number of ticks indicates the degree of satisfaction i.e., two ticks imply clearly meeting a criterion, one tick means weak satisfaction, etc. For example, FPO has a single tick for accuracy because it is accurate in a limited context.

these tools are often infeasible to deploy. Therefore, although it is technically possible to move IrEne from system to system, this is not always the case and so measurement is not cross-platform.

Even when the use of a hardware power monitor is feasible, there is still significant overhead required to train IrEne on a system. The one-in-ten rule states that each predictive variable in a model requires an additional 10 entries in the training set [16, 17]. Therefore, IrEne's 12 independent variables necessitate a training set with at least 120 observations. Furthermore, as each training instance requires repeated execution for 20 seconds, it takes a minimum of 40 minutes of compute time to obtain enough training data. Placing systems under heavy loads for this period of time consumes a lot of energy. Hence, the process of transferring IrEne between machines is clearly compute and energy-intensive.

Such large amounts of overhead are counter-productive and further prevent IrEne-based measurement from being classified as cross-platform. Recall that the development of energy measurement approaches is motivated by the need to design more efficient machine learning algorithms [30]. Widespread adoption of IrEne, across a huge range of different systems to facilitate such work, would incur a vast energy footprint. Therefore, IrEne itself would become a large contributor to the overall energy cost of machine learning research, standing in direct contradiction to its original purpose. Consequently, using IrEne for cross-platform measurement may start to do more harm than good, so we do not describe it as meeting this constraint.

Table 2.1 surmises the contents of this chapter, exhibiting not only whether or not an existing approach meets the given criteria, but also to what degree. Clearly, designing a general solution, compatible with a large number of systems, that is capable of accurate measurement at a fine temporal granularity presents a challenge. We note that some approaches get very close, but fall short on one criterion. For example, low-level software interfaces provide a very accurate measurement of the energy consumed by a diverse spread of hardware components. However, they require sufficient time to do so. Therefore, we believe a combined solution, playing to the strengths of multiple existing approaches, is required.

Chapter 3

Design

3.1 Design goals

The overall goal of this project was to modify the state-of-the-art energy prediction architecture, IrEne, to allow it to generalise to unseen hardware [7]. Obviously, our final design should provide accurate measurements. Furthermore, we argue that there are two additional criteria that should be met by all solutions measuring energy consumption in the domain of machine learning.

First, the ability to provide accurate measurements at a fine temporal granularity is important. While measuring the energy consumed by a full model inference is certainly useful, performing the same measurement for the model's components can prove much more insightful. Energy consumption at the model level can allow us to compare different models, exploring their relative energy demands. However, this does not tell us *why* any differences may exist. If we can measure the energy at the component level, we can determine which parts of a model architecture consume relatively large amounts of energy and make appropriate optimisations [7].

Furthermore, energy measurement within the domain of machine learning should be cross-platform which has two main implications. First, solutions should be simple and intuitive to set up on a wide range of hardware platforms. Gefen and Straub showed that solutions perceived as complex are less likely to see widespread adoption [12]. In addition, it has been suggested that the current difficulties associated with energy consumption measurement are the main reason much research reports no such data [18]. If work is being done on reducing the energy footprint of machine learning by more researchers, then it is more likely that significant improvements will be made. Therefore, methods need to be intuitive to deploy on as many systems as possible.

In addition, the deployment of any cross-platform strategy should itself demand as little energy consumption as possible. While this does not apply to all tools, it could present an issue in certain scenarios. For example, modeling techniques may require the collection of training data to configure an energy prediction model for a new system. Depending on the approach used and the design of the solution, this could require a lot of compute time and could itself consume a lot of energy. If a method requires a

significant amount of energy to set up on each new machine and is widely adopted, the cumulative footprint of all that energy consumption could become huge. Consequently, we argue the energy cost of transferring a measurement strategy to new hardware should be optimised and that solutions which fail to do so are not cross-platform.

Now, we present a more formal definition of each of the three criteria discussed.

1. **Accurate** - To evaluate the accuracy of predictions made throughout this report, we use mean absolute percentage error as our error metric. Doing so allows for direct comparison to results presented by the authors of IrEne as they quantify accuracy in the same way. A formal definition of how we calculate this metric is presented in §4.1.1.

   We define measurement producing a mean error at least as good as that of IrEne as accurate. More formally, we require the mean error at all levels of the model tree to be $< 10\%$ [7].

2. **Cross-platform** - Due to the almost incalculable number of possible unique systems, it is difficult to define cross-platform measurement in a concrete, quantitative sense. Therefore, our definition aims to encapsulate 'enough' unique systems. Let n be the number of distinct systems, constructed from components released in the last decade, supporting low-level software-monitor energy measurement. Then, we require that a cross-platform approach support $m \geq n$ unique machines.

   Also, recall that setting up IrEne on a new hardware platform demands 40 minutes of intense computations. However, we argue that this overhead is too large for IrEne to be widely adopted and must be optimised. Let xJ be the amount of energy required to train IrEne on a new device, D. Then, the cost of setting up a cross-platform tool on D must consume yJ such that $y << x$.

   The term cross-platform applies only where both of these constraints are met. Failure to meet either yields a tool that, although better than the IrEne baseline, is not truly cross-platform.

3. **Fine-grained** - Recall that IrEne defines a tree abstraction, decomposing a target model into its constituent operations at varying levels of granularity (§2.2.3). Given a model M, the set of nodes in its tree is defined as

   $$N = \{n | n_{\text{ml}} \cup n_{\text{mod}} \cup n_{\text{root}}\} \tag{3.1}$$

   where n_{ml} is the set of primitive level nodes, n_{mod} is the set of module level nodes, and n_{root} is the model itself.

   We define measurement capable of accurately estimating the energy consumed $\forall n \in N$ for any model M as fine-grained. For most hardware platforms, this roughly translates as the ability to operate at a sub-millisecond temporal resolution.

It should be noted that despite our best efforts, the specifications provided are not completely orthogonal. Ideally, the criteria would be completely independent. However,

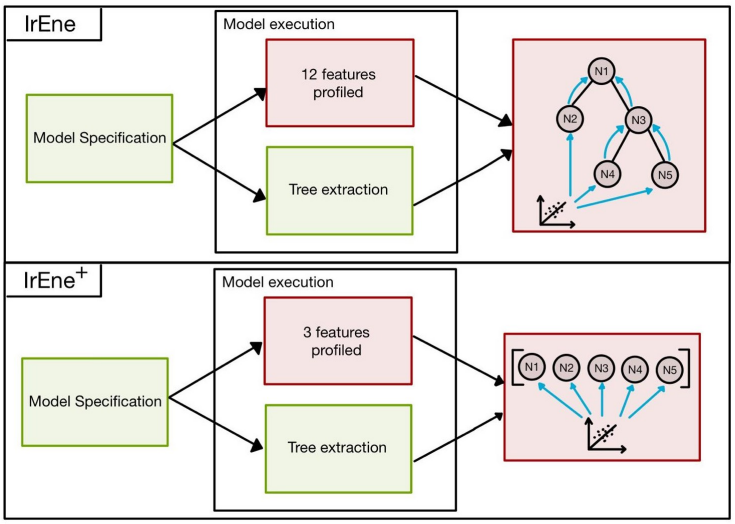


Figure 3.1: Visualisation of the differences between the IrEne and IrEne$^+$ architectures. Energy prediction from input to termination is illustrated for each approach. Green boxes indicate sections of the energy prediction process where no changes have been made, while red boxes indicate areas where IrEne$^+$ builds on its predecessor.

some of them are intrinsically linked. For example, any method is technically capable of providing results at any granularity, but these are useless if not accurate. Similar reasoning applies to the cross-platform criterion. In both cases, it is assumed that any measurements discussed meet the accuracy bounds defined previously. Provided we are aware of these assumptions, the definitions above provide a framework for the binary classification of our design with respect to each requirement.

3.2 IrEne$^+$ overview

Our prediction architecture, called IrEne $^+$, is visualised in Figure 3.1. As with IrEne, the model under test is first run[1] once and its corresponding model tree representation is extracted. During this same run, IrEne$^+$ also profiles all features needed to produce energy consumption estimates. A list of these features can be found in Table 3.1. Then, IrEne predicts the energy consumed by each node of the tree. We redesign the aforementioned steps in two key ways to facilitate cross-platform measurement and simplify prediction without loss of performance.

First, we reduce the set of features profiled, removing those that are inappropriate or appear to provide little useful information when making predictions. As explained in §3.4, a smaller feature set aids in optimising the overhead associated with deploying IrEne to a new hardware platform. Therefore, this change helps our architecture to meet the cross-platform design goal. In §4.2, we evaluate the impact of reducing the number of independent variables and show that it is minimal.

Furthermore, the model tree serves a reduced purpose in the IrEne$^+$ architecture. The

[1]Note: when we say we 'run' or 'execute' a model, we mean running a single model inference. This terminology is used interchangeably throughout this report.

creators of IrEne report that the recursive prediction of this structure is crucial for accurate measurement. However, in §3.3, it becomes clear that the methodology used to establish this was fatally flawed. We still extract a model tree at run-time to establish the set of nodes requiring predictions. However, we replace the tree-based approach, and all its associated regression models, with one multiple regression. This sole model is used to predict the energy of every node in the extracted model tree. An evaluation of the accuracy of each scheme is presented in §4.2.1.

At a high level, IrEne$^+$ obtains the training data required to configure it for deployment on a new system in the same way. Execution of the model associated with each training sample is repeated for a prolonged period, with the total energy during this time measured. Then the mean energy consumption across all the inferences completed is evaluated and recorded as the sample's ground truth. Note that, like IrEne, this is done across a range of models focused on the same task, allowing IrEne$^+$ to predict the energy consumed by other, similar models not included in the training data.

However, recall that training of IrEne incurs significant overhead, which we seek to reduce. We have already seen that IrEne$^+$'s reduced feature set minimises the number of training samples required and thus optimises the training overhead. On top of this, we make two further changes to this process to fully satisfy the cross-platform design goal.

First, we swap the hardware power monitor used to obtain ground truth training energy for the simplified FDLM-based approach described at the end of §2.2.2. Mandating the use of a hardware monitor can prevent cross-platform measurement in two ways. First, doing so assumes ownership of one which may not be the case. Furthermore, even if access to a monitor is available, they are cumbersome to use and require physical access to the system under test which isn't always possible. Instead, when executing a model for a training sample, IrEne$^+$ measures the energy consumed by individual hardware components using low-level software monitors. Then, these values are fed into Equation 2.5 to estimate full-system energy consumption and the mean across all inferences evaluated. In §4.2.4, we evaluate that this approach yields accurate energy measurements given sufficient execution time.

Finally, we reduce the amount of execution time required to obtain each training sample from 20 seconds to 10. Our architecture measures ground truth energy using an approach built on low-level software monitors. Therefore, each model in the training set needs to be run multiple times consecutively to ensure execution time is sufficient to account for their granularity limitations. However, IrEne's 20-second duration was chosen arbitrarily and we find that the overhead per sample can be reduced by 50% with just a 5% difference in the resultant ground truth energies. Therefore, we make this change to further optimise the cost of moving IrEne$^+$ to new hardware in line with the cross-platform design goal.

3.3 Role of the Model Tree

The model tree still serves an important role in Irene$^+$. Understanding the energy consumed at each node of this abstraction provides useful insight, facilitating informed

optimisations of model architectures and computer systems [7]. Therefore, we extract an identical tree at run-time to determine all of the constituent model operations whose energy consumption requires measurement.

However, beyond establishing this collection, the tree representation serves no further purpose. To make energy predictions, we consider its nodes as a set with no structure. Then, we apply the same multiple regression model to each of them. We refer to the simplification of the IrEne architecture, removing the recursive nature of computations, as unstructured measurement.

3.3.1 Flawed evaluation

In evaluating their design, the architects of IrEne pose the question ”is [the] tree structure necessary?” [7]. To test this, they measure the energy consumed by each node of the tree using our unstructured approach. Focusing on the model and module levels, the average errors are reported as 39.8% and 278.0% respectively. Repetition of the same analysis, replacing unstructured measurement with the recursive approach of IrEne, yields corresponding errors of just 8.5% and 5.5%. Therefore, the conclusion that the tree structure is crucial for accurate estimates seems reasonable.

However, the sheer magnitude of the error reported was intriguing. While the idea that adding structure to estimation improves performance feels plausible, no explanation for why is presented. Furthermore, IrEne predicts energy at the leaves of the tree directly in an unstructured way with an error of $<1\%$ [7]. There is a slight difference in approach, as they train a new multiple regression model for each unique leaf node. On the other hand, the unstructured method described uses the same model across all nodes. However, we found it curious that this small change allegedly introduced so much error. Therefore, further investigation was required.

After familiarisation with the publicly available source code [8], it quickly became apparent that a critical mistake had been made. Recall that IrEne makes predictions using various constituent linear regression models. In particular, one of these models produces weights associated with child nodes, used in evaluating the energy consumed by their parent. When making unstructured predictions using the provided code base, this is the sole model responsible for producing per-node energy estimates. Therefore, the values produced for each operation in the tree are *weights*, not *energies*.

Further analysis of the ’energy’ numbers produced confirmed the use of the incorrect model as being the source of the large error observed. In Figure 3.2, we visualise the distribution of the predictions made at the model level by the IrEne code, using both unstructured and tree-based measurement strategies. Clearly, the recursive calculations yield values much closer to the ground truth, while unstructured estimates all appear identical. Figure 3.3 zooms in on the unstructured predictions, highlighting that they all fall in the range $[0.9, 1.1]$ so we conclude they are weights. Naturally, comparing these weights to the generally larger ground truth energy yields a significant error.

Consequently, IrEne$^+$ predicts the energy consumption of each node in the extracted model tree in an unstructured fashion. Clearly, the evaluation of this approach is (accidentally) performed using a model that predicts weights, not energy consumption.

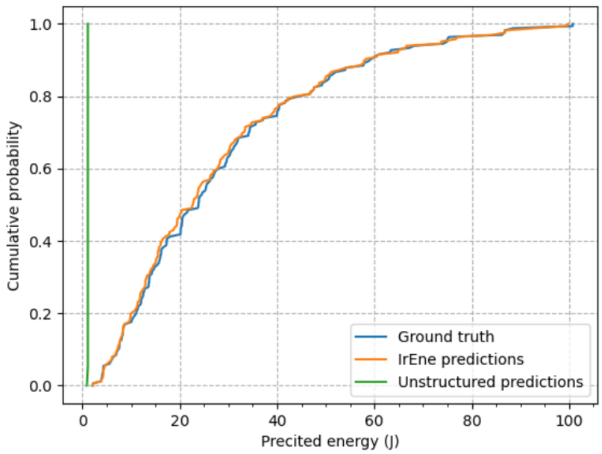


Figure 3.2: Distribution of model level predictions made by IrEne and the unstructured predictor used in IrEne's evaluation. The distribution of ground truth energies is also included for comparison.

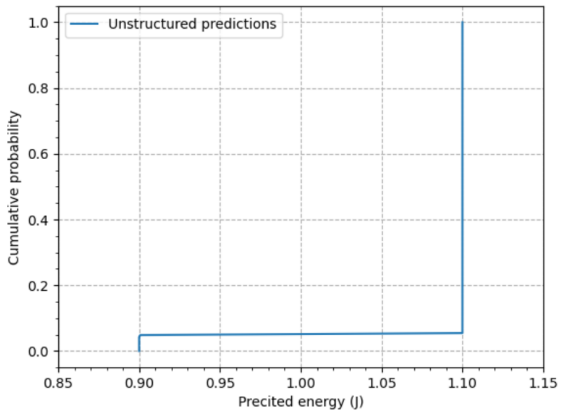


Figure 3.3: Distribution of the model level predictions made by the unstructured model used in IrEne's evaluation.

Furthermore, IrEne evaluates the error of predictions at the primitive level to be $<1\%$, with these estimates generated using a slightly modified unstructured prediction strategy. Therefore, all the evidence suggests that unstructured estimation is actually accurate. In addition, it is much simpler than IrEne's recursive approach. So, we deploy it in our architecture and use the model tree only to identify the set of model operations requiring energy estimation. §4.2.1 evaluates this decision as sound, finding the error produced to be low.

3.4 Feature set

As a first step toward providing cross-platform measurement, we seek to optimise the energy cost of training IrEne$^+$ by reducing the number of predictor variables. Previously, we discussed the one-in-ten rule which states that a linear model with n parameters requires a training set containing $10n$ or more samples [16, 17]. Therefore, by removing independent variables where sensible, we can reduce the size of the training set and the

energy cost of its construction.

The IrEne$^+$ architecture reduces the number of predictive variables from twelve to just two, through a combination of domain knowledge and coefficient analysis. Note, Table 3.1 provides a breakdown of the complete feature sets in both architectures. The remainder of this section explains the process of arriving at this reduced set of features.

3.4.1 Removing the *gpu_energy* feature

First, *gpu_energy* is removed as it should not have been included as a feature to begin with, given the design of IrEne. This feature represents the GPU energy consumption and is profiled using NVML. Recall that all features are profiled during a single run of the model under test, usually with an execution time of around 10-30 ms (Table 4.3). Furthermore, we previously described how NVML, cannot provide accurate results at such a fine granularity (§2.1.2). Therefore, estimating *gpu_energy* in this way yields highly erroneous values, so it must be removed as a predictor variable from the IrEne$^+$ architecture.

The original evaluation of IrEne failed to identify this as the data used was not profiled during a single trial. Instead, models were executed repeatedly for 20 seconds, and features were measured as averages across all complete inferences in this time. However, recall that given sufficient execution time, low-level software monitors provide a good approximation of the true energy consumption (§2.1.2). Therefore, the IrEne evaluation data sets contain accurate *gpu_energy* figures, masking the problem.

While the removal of *gpu_energy* makes sense intuitively, we quantify the inaccuracy of NVML over a single run on a test system, specified in Table 4.2, to support this design choice. For ground truth, we use this machine's associated evaluation data set containing GPU energy measurements for seven different natural language processing models across a range of input sizes. More details about this data set, its corresponding machine, and how it is obtained are presented in §4.1.1. Then, we run a subset of the (model, input size) pairs included in the evaluation data set once and record the energy consumed by the GPU via NVML. Finally, this was repeated four more times using the same subset, and the single-run NVML estimate for each pair was taken as the mean energy across these five runs.

We find the error between the ground truth and the single-run measurements to be very large, as illustrated in Figure 3.4. Almost 50% of the reported values suffer from an error of at least 50% relative to the evaluation data set, with at least 10% experiencing an error of more than 100%. Although the evaluation performed could have been improved through the use of a hardware power monitor to measure ground truth, single-run energy measurement using low-level software monitors is clearly inaccurate. Therefore, *gpu_energy* (when profiled this way) cannot be used as an independent variable in estimating energy consumption.

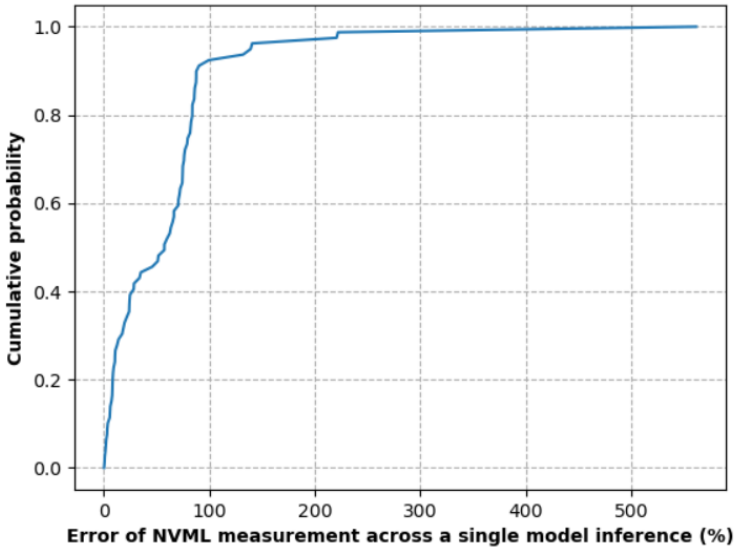


Figure 3.4: Distribution of error when measuring *gpu_energy* during a single model inference using NVML, relative to taking the same measurement as the mean per inference over 10 seconds of repeated model execution.

3.4.2 Further reduction of feature set

Next, we performed an analysis of the model coefficients to determine if the number of features could be further reduced. In a linear model, each feature is assigned a coefficient to determine its weight in evaluating the target variable. Therefore, those features that have coefficients with the largest magnitudes are generally considered the most important [13].

First, the training data was standardised to handle the different scales of measurement. For this experiment, we took the **PC1** evaluation data set from the original paper to be the training set [8]. Then a multiple regression model, with the eleven remaining features, is trained. Finally, the resulting coefficients are inspected to establish which features (if any) are relatively unimportant and can be removed.

Figure 3.5 visualises the resulting coefficients and shows a clear classification of features into two groups. First, there are the important features, namely *times_mean* and *flops* representing the execution time and FPO cost of a model respectively. Then, there are the other nine less important features. The grouping is so distinct, and the coefficients of the less important features are so small, that all nine of these variables are removed from the IrEne^{+} architecture. Therefore, as shown in Table 3.1, only two features need to be profiled to produce an energy estimate. Note, we performed the same analysis on the **PC2** data set and observed identical results. Therefore, we omit these for brevity.

Given that the training set contains data from different model architectures, the strength of FPO as an indicator of energy consumption is surprising. Recall that FPO serves only as an accurate proxy for energy consumption when considering samples within a single architecture (§2.4). Therefore, we might expect its coefficient to be small as FPO should intuitively provide little useful insight when training on a diverse data set. However, the

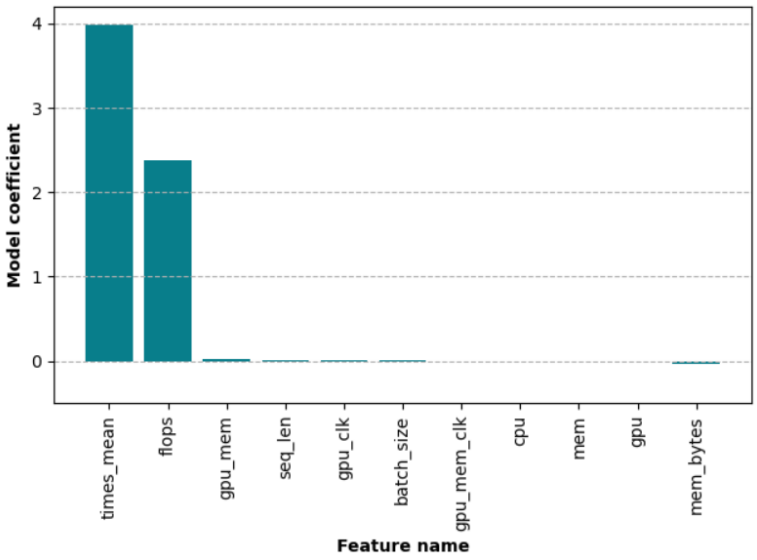


Figure 3.5: Coefficients of a multiple regression model trained on **PC1** evaluation data set with all IrEne features except *gpu_energy*.

times_mean feature naturally groups samples from the same model architecture together, allowing *flops* to provide predictive insight.

Furthermore, it is curious that two-thirds of the less important features relate to hardware resource utilisation. We might expect that the system's load would provide some information regarding energy consumption. However, the use of repeated execution of models to create the data sets provided placed the system under an almost identical load for each sample. For example, all GPU utilisation values fall in the range [96%, 100%], while the equivalent range for the utilisation of the CPU is [5.4%, 6.0%]. Obtaining a more complete picture of these features' impact on energy consumption requires the creation of broader ground truth data sets covering a wider range of loads. However, this project was already quite broad in scope and so we lacked the time to investigate this thoroughly. Therefore, we leave this to future work and omit these features based on the data we have access to.

On the other hand, the small coefficients for the remaining three features, *batch_size*, *seq_len*, and *mem_bytes*, make sense. In the original architecture, they all fall into the category of model features, reflecting "hardware-independent compute and memory information for a given model" [7]. However, *flops* also falls into this category. Therefore, as *flops* is such a strong predictor of energy consumption, and the model features all capture similar information, the remaining three features cannot provide any further insight beyond that of *flops*. Consequently, their coefficients are small and they can be removed from the feature set. As §4.2.3 shows, doing so has no negative impact on the accuracy of IrEne$^+$ energy estimates.

In summary, we introduce and motivate two changes to the set of features used to estimate energy consumption across the IrEne and IrEne$^+$ architectures. First, we remove *gpu_energy* from the set of independent variables as it cannot be reliably profiled. A further nine variables deemed to be less important in making predictions

Feature	Abbreviation	IrEne	IrEne$^+$
FPO	*flops*	✓	✓
Batch size	*batch_size*	✓	
Sequence length	*seq_len*	✓	
Model size (MiB)	*mem_bytes*	✓	
Execution time (s)	*latency*	✓	✓
GPU driver energy (J)	*gpu_energy*	✓	
CPU utilisation (%)	*gpu_util*	✓	
GPU utilisation (%)	*cpu_util*	✓	
Memory utilisation (%)	*mem_util*	✓	
GPU memory utilisation (%)	*gpu_mem_util*	✓	
GPU processor clock speed (MHz)	*g_clk*	✓	
GPU Memory clock speed (MHz)	*gm_clk*	✓	

Table 3.1: Adapted from [7]. Enumeration of all features used by the IrEne architecture, and those still used in IrEne$^+$. The line separates features into the categories of model-based and hardware-based respectively.

are also pruned. The consistency in predictive performance after doing so, discussed in §4.2.3, supports this decision.

With our reduced feature set, it takes approximately seven minutes to obtain the data needed to train IrEne$^+$ when using the data procurement strategy proposed in the original paper [7]. Consequently, these modifications alone allow IrEne$^+$ to satisfy part of the cross-platform design goal defined in §3.1 concerning deployment overhead.

3.5 Training Process

In place of a hardware monitor, as required by its predecessor, IrEne$^+$ obtains ground truth energy for its training samples using a software-based approach. First, low-level software monitors measure the total energy consumed by the CPU, GPU, and RAM. As with IrEne, model execution is repeated due to the limited granularity of these tools (§2.1.2). Then, the mean energy consumed by each component across all repeated inferences is supplied to the modified FDLM described in Equation 2.5 and the output recorded as the ground truth. Given that in §3.4 the cost of training IrEne$^+$ was significantly reduced, the use of a hardware power monitor is the only factor preventing cross-platform measurement. Therefore, switching this to a software-based approach ensures our architecture meets this design goal in its entirety.

In addition, the cost of training an IrEne$^+$ instance is further optimised by reducing the amount of time spent obtaining each training sample. In §3.4, a reduction in the size of the architecture's feature set decreased the overhead required to construct a training set from 40 minutes to just seven. In addition, the IrEne$^+$ architecture optimises the period spent repeating the execution of models when measuring ground truth energy for training samples. By doing so, a further improvement of 50% can be achieved. Of course, shrinking the feature set alone yielded a sufficient reduction in training overhead

to meet the definition of cross-platform measurement. However, if a lower cost can be achieved without penalty, then the corresponding design changes must be implemented.

In the remainder of this section, we first expand on why replacing a hardware monitor with the simple FDLM introduced in equation 2.5 makes sense. Then, the process of arriving at 10 seconds as the amount of time to spend repeating the execution of models when gathering training data is described.

3.5.1 FDLM-based ground truth measurement

The use of a hardware power monitor to obtain training data is the main remaining factor preventing IrEne from providing cross-platform measurement. Hardware monitors can be used to measure ground truth on almost any system, so IrEne is technically cross-platform. However, the effort required to set these tools up adds significant overhead to the training process, so classifying IrEne this way is incorrect. It is important to note that they are required only to obtain ground truth energy in the training process. Therefore, if the same data can be obtained via cross-platform means, then the wider architecture can be described as such.

Consequently, the substitution of hardware power monitors for low-level software-based alternatives is a natural first step. Recall that these interfaces provide accurate results so long as the measurement period is sufficiently long (§2.1.2). Recall further that IrEne repeats the execution of each model for 20 seconds and measures the total energy consumed in this period, taking the ground truth as the mean across all model executions [7]. Therefore, by simply replacing the system-wide hardware power monitor with per-component software interfaces, we can accurately estimate the true energy consumed by each of the CPU, GPU, and RAM during a model inference.

This is progress, but the ground truth energy of a training sample must reflect the energy consumed by the entire system. In IrEne, the hardware power monitor produces this figure directly, measuring the energy consumed at the machine level. However, low-level software interfaces do not, instead providing consumption figures for only a few of the machine's components. Therefore, a final step needs to be taken to accurately infer system-wide energy consumption from these component-wise measurements.

IrEne^{+} achieves this through the use of the FDLM defined in Equation 2.5 i.e., by taking full system energy consumption as the sum of values produced by the low-level software monitors. For the remainder of this work, we refer to this method of obtaining ground truth energy for training sets as simple FDLM-based measurement.

Intuitively, this approach feels inaccurate because of its simplicity and consequent perceived failure to account for energy consumed beyond the included components. Furthermore, a previous evaluation of a similar model, with an additional PUE scaling factor, used widely in the domain-specific literature found it to be highly inaccurate [7]. However, in §2.2.2, we explained that the perceived error stemmed from a methodology error. Therefore, we argue that as the components profiled account for most of the energy consumed by machine-learning processes, simply summing accurate measurements of their energy consumption should yield an accurate estimate at the system level. In §4.2.4 we evaluate this empirically, finding our hypothesis to be correct.

Consequently, it is natural to ask why we don't just scrap IrEne$^+$ and use this simple, accurate FDLM instead. However, the underlying component-wise energy figures on which it relies are obtained using low-level software monitors. Therefore, the FDLM itself cannot provide fine-grained estimates. On the other hand, IrEne$^+$ itself is capable of such measurement, as the features it needs to make predictions can all be accurately profiled during a single model inference. Therefore, although the simple FDLM is useful to provide cross-platform measurement of the ground truth energy, its granularity restrictions necessitate the design of further models that do not depend on low-level software monitors at prediction time.

In summary, simple FDLM-based measurement involves taking per-component data from low-level software monitors and using these to estimate full-system energy consumption as their sum. Unlike its counterpart incorporating a PUE coefficient, these estimates are accurate when we consider only the energy consumed directly by the system itself. Furthermore, as simple FDLM-based measurement is essentially just an amalgamation of the low-level software interfaces associated with each component, it shares their benefits and is cross-platform. However, it cannot operate at fine granularities for the same reason.

3.5.2 Reduing the time per training sample

When producing training samples IrEne executes the relevant model for 20 seconds but, given that IrEne$^+$ uses simple FDLM-based measurement, this seems excessive. In their architecture, this period is chosen to account for the granularity of their hardware monitor (170ms). However, 20 seconds seems to be chosen as an arbitrary, 'long-enough' duration. Similarly, our training process must always use sufficient execution time to handle all potential granularity limitations of simple FDLM-based measurement, including power lags. However, even in the worst-case scenario, with a significant GPU power lag of a few seconds, 20 seconds feels unnecessarily long. Therefore, we suppose that the time per sample could be reduced while maintaining the accuracy of ground truth measurement.

More concretely, we seek to establish some minimum execution time, n seconds, which ensures the accuracy of simple FDLM-based energy measurement on any system[2]. Recall that FDLM-based measurement is not fine-grained, but becomes accurate with sufficient time. Therefore, the idea that there is a threshold beyond which measurement becomes accurate seems reasonable. However, we saw in §2.1.2 how GPU power lags sometimes exist and make the granularity of measurement possible using NVML much coarser. Therefore, this threshold may vary from system to system. So, to define a single duration for all systems, we must make n sufficiently large to handle the worst-case power lag. To determine n quantitatively, we investigated the relationship between the accuracy of FDLM-based measurement and execution time.

First, we needed to establish the ground truth. Given our suspicion that the execution time required by simple FDLM-based measurement to produce accurate results may vary across platforms, we performed our experiment on two different systems. These

[2]Within the set defined by our cross-platform measurement design goal in §3.1.

are specified in Table 4.2. To construct our data sets, we used the method described in §4.1.1. However, each (model, input size) pair's execution was repeated 1000 times, rather than for 10 seconds.

We now needed the same data for a range of sample repeat counts. Listing 4.1 illustrates the procedure by which ground truth data was collected, using a repeat count of 1000. Therefore, we carried out the same procedure on each of our target machines with a repeat count of five. Then, we did the same for further repeat counts of 10 through to 150 with a step size of 20.

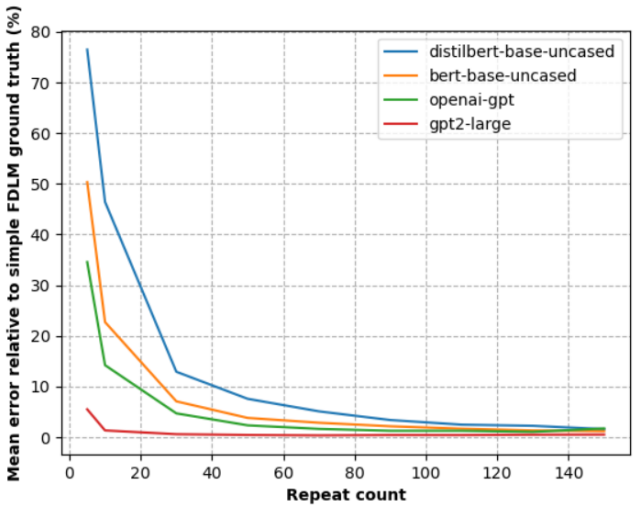


Figure 3.6: Relationship between the number of times a model inference is repeated, and the error observed in the mean energy measurement on **PC3**. The reported error is relative to ground truth obtained by repeating execution 1000 times, measuring total energy consumed using a simple FDLM-based approach, and taking the mean per inference.

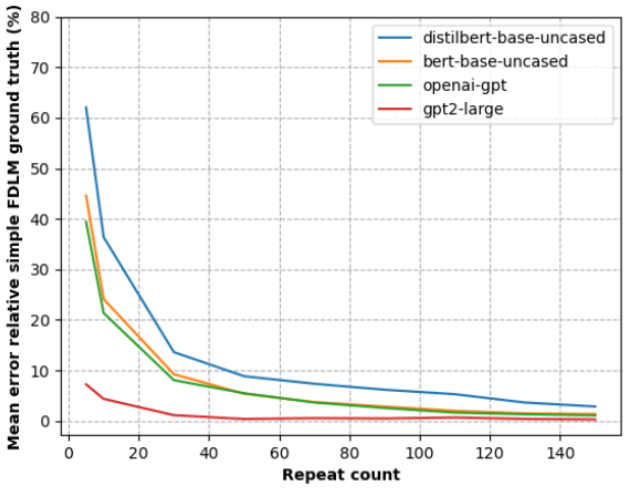


Figure 3.7: Relationship between the number of times a model inference is repeated, and the error observed in the mean energy measurement on **PC4**. The ground truth energy is obtained as described in the caption of Figure 3.6.

Figures 3.6 and 3.7 illustrate the results of this experiment on **PC3** and **PC4** respectively. First, we note that across both machines, larger, slower models generally take fewer repeats to converge. The promising implication here is that it takes the same amount of time to measure the energy consumed by any model on a given system. Furthermore, we note from Table 4.3 that **PC4** is clearly the more powerful of the two, performing model inferences $2.4\times$ faster on average. Therefore, all things equal, we might expect accurate measurement of each model to require around twice as many repeats on **PC4**. However, we see that each model takes a very similar number of repeats to produce the same error across both machines.

To investigate further, we sought to concretely quantify the amount of time required for accurate measurement of each combination of system and model. For a given pair, we first determined the smallest repeat count where the mean error was $\leq 5\%$. This threshold was chosen by inspection of Figures 3.6 and 3.7 where we observe that errors tend to decrease exponentially to around this level. Then, we multiplied the selected repeat count by the mean model inference time across all input sizes.

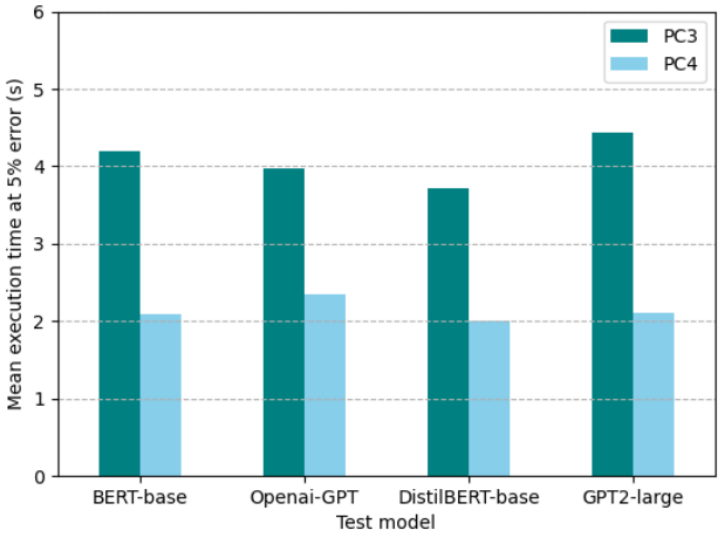


Figure 3.8: Minimum execution time at which simple FDLM-based measurement produces a mean error of $\leq 5\%$ across all models on each of our test systems.

The resultant minimum required execution times for each (model, system) pair are compared in Figure 3.8. We observe first that for a given machine, the run times required for simple FDLM-based measurement to yield accurate results are almost identical. Particularly, we note that there does not appear to be any relationship between the model size and the required duration. Furthermore, we see clearly that simple FDLM-based measurement takes approximately twice as long to produce accurate results on **PC3**.

We believe that these observations can be attributed to the presence of a GPU power lag on **PC3**. Recall from §2.1.2 that both these and the sampling intervals of low-level software monitors introduce a number of erroneous GPU power samples at the beginning of execution. However, we also discussed how power lags result in a much larger number of them, so the error they introduce requires a much longer execution to amortise. Therefore, as the duration required across all models is much longer on **PC3**,

clearly its GPU suffers from a power lag. As for the GPU of **PC4**, this may or may be the case. It is impossible to say without performing the same analysis on a PC with a GPU known not to have a lag, using this as a baseline.

Regardless, the insight provided by this experiment allows us to make an informed decision to set $n = 10$ seconds, a reduction in overhead of 50% per training sample relative to IrEne. Our investigations imply that even with power lags, a simple FDLM-based measurement approach can produce accurate results in much less than 20 seconds. Furthermore, we note that **PC3**'s GPU was released in 2016, and **PC4**'s in 2020. Therefore, it seems that the effects of these lags on the time required are reducing with advances in technology. To satisfy our cross-platform design goal in §3.1, n must be sufficient to handle the effects of the power lags of any Nvidia GPUs released in the last decade. Therefore, we feel $n = 10$ seconds should provide enough a sufficient to meet this requirement while simultaneously providing a significant reduction in training overhead.

Of course, the effects of power lags on the execution time required by simple FDLM-based measurement should be investigated across a wider range of GPUs. Doing so would concretely determine whether $n = 10$ is sufficient, and whether it can be optimised further. However, we only had access to the two machines detailed here and so we leave this to future work.

In summary, the IrEne^{+} energy estimation architecture differs from its predecessor in four key ways, with three of these relating to its training process. First, the hardware power monitor required to obtain training data is replaced with our simple FDLM-based energy measurement strategy. This approach estimates total energy as the sum of the values yielded by low-level software interfaces, so is obviously cross-platform and intuitively accurate. Furthermore, we significantly reduce the amount of time required to obtain training data. This is achieved through a smaller feature set, removing redundant or improper features, and by quantifying the minimum amount of execution time our FDLM-based approach requires to yield accurate training samples. Finally, our architecture does not make predictions using the same recursive, tree-based approach as IrEne. We find that the evaluation supporting this design decision is fatally flawed, so instead opt for the simpler approach of directly estimating the energy consumed by each model tree node directly. This is done using the same multiple regression model for all tree nodes.

Chapter 4

Evaluation

Having explained the design changes made from IrEne to IrEne$^+$, and reasoned about why they are sensible along with the benefits they bring, we now evaluate the resulting architecture. First, we introduce our general evaluation procedures in §4.1, describing the data sets used, the machines on which they were measured, and the metric used to quantify error. Then, §4.2 assesses the reasoning behind each design decision. Finally, an evaluation of the predictive performance of our architecture as a whole is performed in §4.3.

4.1 Setup

4.1.1 Machines and data sets

For the purposes of evaluation, the authors of IrEne construct two data sets, one for each of the systems detailed in Table 4.1 [8]. These data sets cover the model tree nodes of the first six transformer models enumerated in Table 4.3 across a wide range of input sizes, covering batch sizes eight through 64 with a step size of 8. The corresponding sequence lengths are 32 through 256 with a step size of 32. Batch size 32 is also included, with reduced sequence lengths of 32, 64, and 96 due to GPU memory limitations.

Each sample contains all 12 of the IrEne features specified in Table 3.1, including ground truth energy measured using a hardware power monitor. Measurements were taken by repeating the execution of each (model operation, input size) pair for 20 seconds and recording the mean for each feature. This repetition is necessary only to ensure only that ground truth energy is accurately measured, as the hardware monitor used has a relatively high sampling interval of 170 milliseconds. However, as §3.4.1 discussed, this leads to *gpu_energy* measurements being accurate which doesn't reflect the design of IrEne.

In addition, we create two similar data sets of our own for each of the systems specified in Table 4.2. These data sets differ from the IrEne evaluation sets in a few key ways. First, these data sets include only entries for the root nodes of each model's tree. This is because it would have taken too long to gather all the required data if other levels were included. Each sample also contains only the features required by IrEne$^+$. Furthermore,

Specification	**PC1**	**PC2**
CPU	Intel i9-7900X	Intel i7-6800K
Memory	32 GiB	32 GiB
GPU	2 x GTX 1080 Ti	2 x GTX 1070
GPU Memory	11.2 GiB per GPU	8 GiB per GPU
Storage	1 TiB SSD	1 TiB SSD

Table 4.1: Taken from [7]. Specification of the systems used to evaluate original IrEne architecture.

Specification	**PC3**	**PC4**
CPU	Intel Xeon Silver 4116	AMD Ryzen 9 5950X
Memory	64 GiB	128 GiB
GPU	2 x GTX 1080 Ti	1 x RTX 3090
GPU Memory	11.2 GiB per GPU	24 GiB
Storage	1 TiB SSD	2 TiB SSD

Table 4.2: The two systems on which we constructed our own data sets. Note how **PC3** here is very similar to **PC1** in Table 4.1, except for the CPU and amount of memory. Note RAPL support by other hardware manufacturers is becoming common, hence the AMD CPU of **PC4**.

an additional model, GPT2-Large, is included in our sets to facilitate the work of §3.5.2. Finally, the ground truth energy was obtained using the IrEne^{+} training approach i.e., with software tools and a reduced execution time of 10 seconds per sample. As shown in Listing 4.1, we first obtained five data sets using this approach and then recorded the final ground truth for each sample as the mean across these.

```
@record_energy_and_features
def repeat_execution(model, input_size, repeat_duration):
        start_time = time.time()
        while time.time() < start_time + repeat_duration:
            _ = model(input_size)

def construct_dataset(repeat_duration):
    model_configurations = [...]

    for model, input_size in model_configurations:
        repeat_execution(model,
            input_size,
            total_repeats
        )

if __name__ == "__main__":
    for i in range(5):
        construct_dataset(10)

    # Now, evaluate and record mean energy consmuptions across
```

Model	Paper	Parameter Count (M)	T_{PC3} (s)	T_{PC4} (s)
BERT	[11]	110	0.084	0.030
RoBERTa	[22]	125	0.083	0.031
OpenaiGPT	[24]	110	0.079	0.034
GPT2	[25]	124	0.078	0.034
DistilGPT2	[25]	82	0.039	0.017
DistilBERT	[26]	66	0.041	0.015
GPT2-Large	[25]	774	0.444	0.212

Table 4.3: The models found in the IrEne evaluation sets and those we constructed. The number of model parameters gives a sense of their relative size, While T_{PC3} and T_{PC4} represent each model's mean inference execution time on each of our systems across all input sizes studied. Note that GPT2-Large is not present in the IrEne evaluation data sets.

```
    # the five resultant data sets.
```

Listing 4.1: Pythonic pseudo-code outlining how we constructed our evaluation data sets at a high level.

4.1.2 Error measurement

For the sake of consistency, and to allow a direct comparison of results across the two architectures, we use the same error metric found in the evaluation of IrEne. Formally, given an energy prediction, PE, and a corresponding ground truth energy, GE, the error percentage is defined as

$$error = 100 \times \frac{|PE - GE|}{GE} \tag{4.1}$$

Note that, unless stated otherwise, all errors we report are cross-validated. By this, we mean that given an evaluation set with entries relating to n unique models, we leave one model out and train on the data for the remaining $n-1$. Evaluation is then performed using the data of the omitted model. This process repeats n times until all models have served the role of the test model, yielding n sets of model and module-level errors.

The evaluation of IrEne uses this cross-validation strategy to demonstrate that its predictions generalise to unseen models. Leaving one model out of the training set and using it for evaluation shows that the architecture can accurately predict the energy consumption of similar models not included in the training data set. IrEne$^+$ should exhibit the same benefits with respect to unseen models, therefore we too evaluate in this cross-validated sense.

4.2 Impact of IrEne$^+$ design decisions

First, we evaluate some key design decisions made in Chapter 3 individually to quantify the impact they may have on the overall predictive performance of the IrEne$^+$ architecture. With each decision, the specific goal of its evaluation is different. For example,

	Module-level	Model-level
Reported Unstructured Error (%)	278.0	39.79
Actual Unstructured Error (%)	0.93	0.45

Table 4.4: Comparing the inflated error reported using unstructured measurement in the IrEne evaluation, to the error we saw when correctly performing the same evaluation [7]. All results observed on **PC1** (table 4.1).

with the shift to an unstructured prediction, we seek to perform the work carried out incorrectly and determine whether an unstructured or recursive tree-based measurement approach is more accurate, all things equal. On the other hand, for decisions aimed at facilitating cross-platform measurement, we aim to concretely measure any accuracy trade-offs they may bring. Considering each design decision in an isolated context is important to strengthen claims we later make about the performance of IrEne$^+$, providing evidence that differences observed relative to its predecessor are not coincidental.

4.2.1 Unstructured measurement is accurate

Recall from §3.3.1 that the evaluation performed by the authors of IrEne to determine the accuracy of unstructured measurement was flawed. In their analysis, they compared the outputs of one of IrEne's constituent prediction models to tree-based energy estimates. However, they accidentally used an unstructured weight prediction model, rather than one trained to estimate energy. Therefore, the conclusions they drew were invalid as the correct quantities were not compared.

To quantify the true error associated with unstructured measurement, we repeated the experiment correctly. We constructed a multiple regression model using the same twelve features as the IrEne architecture, and evaluated its mean error across all models in the **PC1** dataset. Then, we repeated this process for **PC2**. We used the data from these machines as they are the ones on which the flawed evaluation was performed. This left us with a mean model and module-level error for each of the six test models, on each system.

We found that the evaluation of unstructured measurement previously performed greatly overestimated the error in its predictions. As we see in Table 4.4, it overestimated the true error by nearly 300% at the module level, and over 80% at the model level. While we would expect there to be a degree of overestimation, it is interesting that an unstructured approach is so accurate with a mean error or $< 1\%$ at all levels of the tree abstraction. Similarly, Figure 4.1 implies that an unstructured estimation approach is much more accurate than the tree-based, recursive predictions of IrEne.

There is a slight caveat here. For the sake of consistency, we have repeated the analysis *exactly* as it was previously performed. For example, we have used all twelve features of the IrEne architecture and data obtained with a hardware monitor. Therefore, the results here do not reflect the true accuracy of IrEne$^+$ measurements and are not to be interpreted as such. Instead, what this evaluation showed clearly was that all things equal, directly predicting the energy consumption of each tree node using the same multiple regression model is highly accurate, even more so than IrEne.

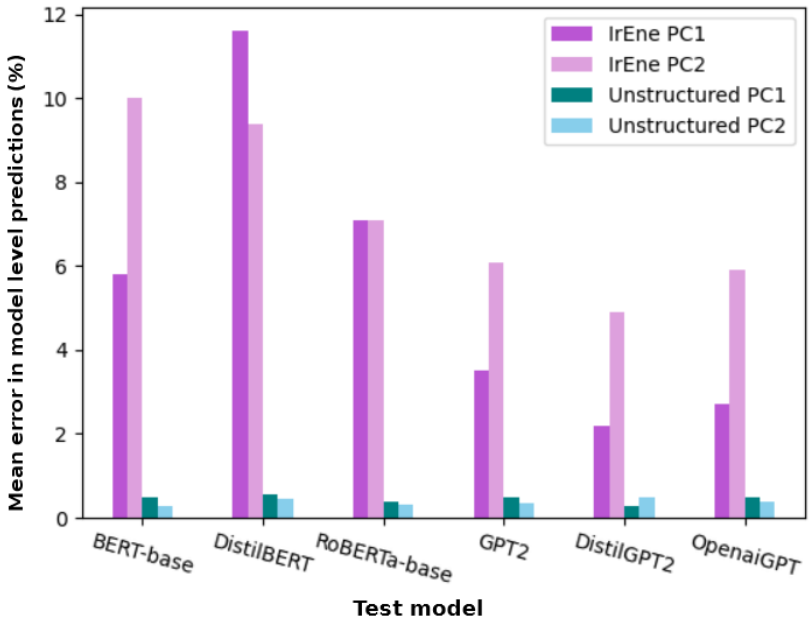


Figure 4.1: Comparison of the average error produced by IrEne and unstructured measurement at the model level on both **PC1** and **PC2**, relative to corresponding hardware ground truth measurements. Errors are reported for each test model were obtained using the cross-validation strategy described in §4.1.1. We omit errors at other levels of the model tree to ensure readability. However, almost identical results were observed.

4.2.2 Mandatory removal of *gpu_energy* has little impact

Next, we evaluated the impact of removing *gpu_energy* as a feature from our multiple regression model. Of course, there is no choice as to whether this feature is to be removed or not. It cannot be accurately profiled over a single model inference, and so it must be pruned. However, it is still useful to understand the resulting impact on the predictive accuracy of IrEne^{+}'s multiple regression model. Furthermore, doing so allowed us to establish a baseline accuracy to be used when evaluating the optional removal of other features in §4.2.3.

The steps undertaken were very similar to those of §4.2.1. First, we evaluated the accuracy of the same multiple regression model as before, however *gpu_energy* is removed as a feature. Again, this was performed using each of the **PC1** and **PC2** data sets in turn. Then, we compared the results to the error observed when performing the same analysis with *gpu_energy* included.

While the mandatory removal of *gpu_energy* as a feature did have a negative impact on accuracy, the penalty was very tolerable. Figure 4.2 visualises the results of this evaluation on **PC1**, showing that the unstructured model trained without *gpu_energy* produced a higher error. However, recall that the IrEne evaluation data sets were constructed using repeated model execution so they contain accurate *gpu_energy* values. Therefore, it is not surprising that, in the context of these data sets, *gpu_energy* was a good indicator of total energy consumption. More importantly, we observed that despite

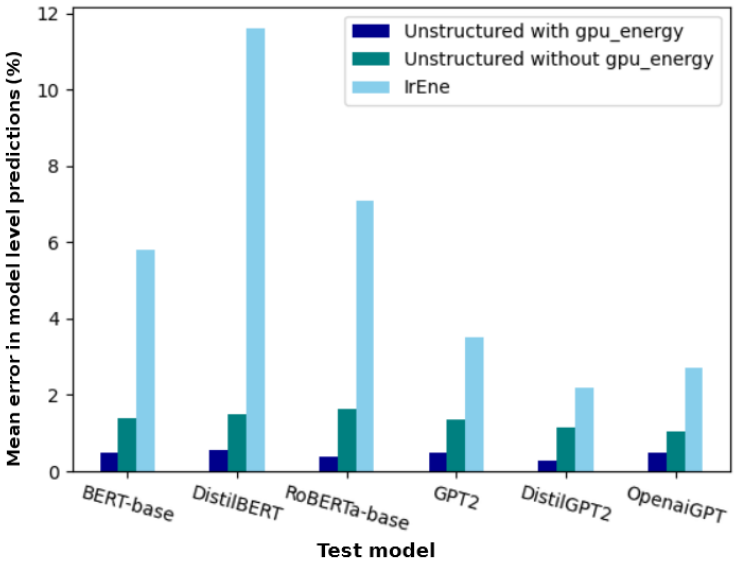


Figure 4.2: Comparison of the average error produced by unstructured measurement with and without the *gpu_energy* feature, and IrEne. Errors shown are for the model level on **PC1**. Clearly, when profiled correctly, *gpu_energy* is a good indicator of total energy consumption. However, we have thoroughly explained that the inclusion of this feature is a poor design decision. The key observation here is that all things equal, predictions energy consumption of all model tree nodes using one multiple regression model were more accurate than IrEne, even with *gpu_energy* removed.

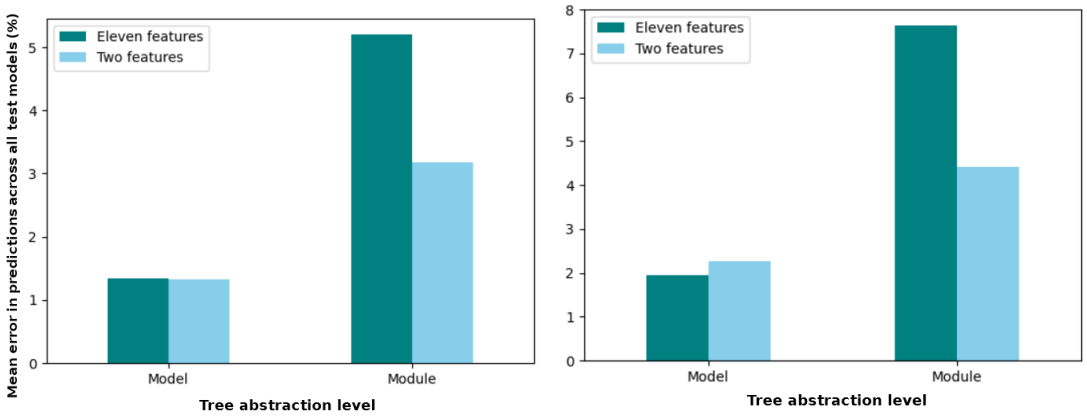
the increase, unstructured measurement trained without *gpu_energy* was still much more accurate than IrEne. Therefore, our decision to move away from tree-based estimation and use this strategy instead appeared sound.

Again, these results are not to be interpreted as errors associated with IrEne$^+$ predictions. Instead here we quantified the baseline error of a multiple regression model trained on hardware power meter ground truth with every feature found in the IrEne architecture except for *gpu_energy*. Doing so was a logical preliminary step to facilitate the experiment presented in §4.2.3.

4.2.3 Optional feature removal has no negative impact

In §3.4.2, we proposed removing a further nine of the IrEne's features from our multiple regression model based on coefficient analysis. Doing so yielded over an 80% reduction in the training overhead, helping IrEne$^+$ to meet part of the cross-platform design goal defined in §3.1. However, we needed to establish if there was an associated accuracy trade-off. To evaluate this empirically, we performed the cross-validation procedure described in §4.1.2 with two multiple regression models trained on different subsets of the **PC1** evaluation data set. The first contained all IrEne features except for *gpu_energy*, while the other provided data for just the two features used by our architecture. Then, we repeated this process for the **PC2** evaluation data set.

We found that the removal of these features had no clear negative impact on the accuracy



(a) **PC1** (b) **PC2**

Figure 4.3: Comparison of error produced by a multiple regression model trained on the eleven original IrEne features versus the another using the IrEne^{+} reduced feature set. Note, the errors presented are means of those for all six test models obtained using the cross-validation strategy described in §3.3.

of energy estimates produced across all hardware surveyed, as seen in 4.3. Even though the error was generally higher on **PC2**, the trends remained the same and, overall, the error produced was low. Furthermore, recall that in §4.2.3, we demonstrated that the eleven-feature multiple regression produced more accurate energy estimates than IrEne. Note also that our evaluation procedure was identical to the one used to evaluate IrEne. Therefore, given that this two-feature model lies at the core of IrEne^{+}, we can conclude that using the same training strategy, our architecture exhibits superior predictive performance. This is a significant finding as that IrEne represents the state-of-the-art in our field of work.

There is an important caveat to both this design decision and the conclusions just drawn which we briefly discussed in §3.4.2 but reiterate here. When performing the coefficient analysis to determine if any features could be removed, we noted that very low coefficients of resource utilisation features, such as *gpu_util*, *cpu_util* etc. appeared rather counter-intuitive. It seemed plausible to expect that higher component utilisation should lead to higher overall energy consumption, particularly for the GPU which is responsible for most of the energy consumed by a model inference. However, the method by which all evaluation data sets discussed in this report were generated placed the system under an almost constant load.

Therefore, further evaluation with data sets obtained across varied system loads is required across both architectures, to study whether this affects their predictive capabilities. Unfortunately, as we make several significant changes to the design of IrEne, we didn't have time to thoroughly explore this route. Therefore, while we believe our design decision and evaluation of it are valid given the data we had access to, we concede that more work needs to be done to determine if our conclusions hold more broadly.

4.2.4 Simple FDLM-based measurement

The decision to provide cross-platform measurement by moving from a hardware power monitor to a simple FDLM-based approach seems logically sound, but the accuracy of this technique requires evaluation. It has been shown empirically that the low-level software interfaces, on which simple FDLM-based measurement relies to obtain per-component energy consumption, are accurate given sufficient execution time [19, 23]. However, previous evaluation has found existing tools using an FDLM with a PUE term to be highly inaccurate. In §2.2.2, we explained how it appears this error stems from the fact that the PUE term aims to account for energy not directly consumed by the system. Therefore, we proposed that our simplified version of FDLM measurement, which simply estimates system energy consumption as the sum of the energy consumed by its major components, is accurate. The remainder of this subsection describes how we evaluated this claim.

Originally, we planned to estimate the energy consumed for each (model, input size) pair in the **PC1** data set using our simple FDLM-based approach on **PC3**, and then compare the results directly to the corresponding hardware ground truth. If we are to use our software-based approach to obtain ground truth energy for training samples, then we first need to ensure that it produces similar figures to the hardware monitor used by IrEne. Given that one of our systems, **PC3**, is very similar in specification to one of the systems used to develop IrEne, **PC1**, it seemed we could run tests on our machine and compare directly to the relevant provided hardware ground truth.

However, while this approach isn't perfect and some degree of error was to be expected, we saw that our measurements were systematically underestimating the energy consumption relative to the **PC1** ground truth. Upon further analysis of the data, it became clear **PC3** was running model inferences much faster than **PC1**. On average, the speedup was around 30% as seen in Figure 4.3. While this speedup was counter-intuitive given the similarity in system specification, we believe it was caused by a more recent version of CUDA running on **PC3**. Unfortunately, the authors of IrEne did not specify the version running on **PC1** at the time of measurement, so this speedup factor was beyond our control.

Therefore, the depth of evaluation we were able to perform was somewhat limited, but the results produced still supported our proposal. Recall that simple FDLM-based measurement simply predicts total energy consumption as the sum of the energy consumed by each component; it does not require any features or use any complex modeling techniques. Therefore, we could simply investigate the relationship between estimated energy consumption and execution time for both the provided **PC1** data set and the **PC3** data set we obtained for this evaluation. Figure 4.5 illustrates the results of this analysis. Clearly, the relationship between energy and time was almost identical across the hardware ground truth and our simple FDLM-based measurements. Therefore, although there was a large error according to our error metric, we conclude that simple FDLM-based measurement is sufficiently accurate to measure ground truth energy in our architecture.

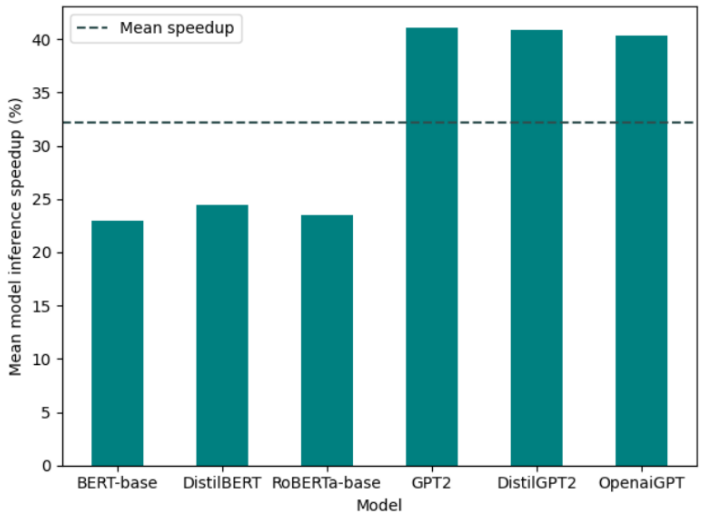


Figure 4.4: Mean speedup of model inference execution time on **PC3** relative to **PC1** across all models included in **PC1** evaluation data set.

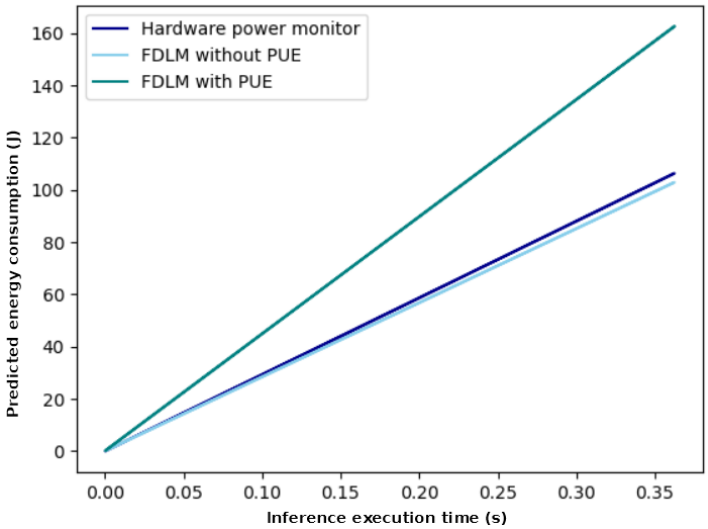


Figure 4.5: Relationship between inference execution time and predicted energy consumption for different ground truth energy measurement strategies. The hardware power monitor line is derived from **PC1** evaluation data, while the FDLM lines are derived from the data obtained on **PC3** in §4.2.4. We also include the relationship for the FDLM with PUE (derived from the same **PC3** data) to illustrate the error this term introduces.

4.3 Accuracy of IrEne$^+$

Having examined our key design decisions and obtaining positive results, we moved on to evaluate the predictive performance of our full architecture bringing all of these elements together simultaneously. The remainder of this section presents this analysis.

To make any claims about the accuracy of IrEne$^+$, we needed to evaluate its predictions against hardware ground truth as this is the gold standard in energy measurement research. Of course, our work in §4.2.4 implies that FDLM-based measurement serves as a good alternative where a hardware power monitor is not available. However, our evaluation of this approach is not as thorough as it could have been due to our own lack of access to a power monitor, so we cannot quantify exactly how good it is. Therefore, drawing very strong conclusions about our architecture's predictive performance is difficult without assessment against hardware ground truth.

Unfortunately, we lacked access to a monitor ourselves, so the best we could do was to train an IrEne$^+$ instance on **PC3**, and evaluate its accuracy using the **PC1** ground truth data supplied by IrEne's authors. First, we think it important to mention that the speedup from **PC1** to **PC3** observed in 4.2.4 should have no negative impact on the quality of our evaluation. We expect our model to be able to handle samples with the same FPO count and different execution times. On the other hand, this approach wass not perfect as the system specifications were not completely identical. As we have mentioned, there was no alternative and so we proceeded with this method

Due to the number of samples in the **PC3** evaluation data set, we needed to select a smaller subset every time we trained to stay true to the design of IrEne$^+$. As usual, we first selected a large subset to omit the test model, as specified by our cross-validation strategy. Then, for a reduced set of 20 input sizes, we selected a random model from those left in the training set and added its data for the relevant input size to the training subset This allowed us to ensure our evaluation accurately reflected the design of IrEne$^+$, without having to train on the same 20 samples every time, potentially introducing bias.

In addition, we performed cross-validation 1000 times. Normally, performing it once is sufficient as the model is trained and tested on the same data in each fold, every time the procedure is run. However, now there was a degree of randomness to the training set, so running multiple times yielded different errors each time. Therefore, we repeated the entire cross-validation 1000 times and took means across all runs as the true, cross-validated error.

Figures 4.6 and 4.7 illustrate the results of the mean errors we observed at the model and module levels respectively. First, note that errors here are much higher on average than those seen in Figure 4.3 which represent the true error of our architecture's underlying prediction model when trained on ground truth energy. This implies that there is an accuracy penalty associated with providing cross-platform measurement However, this is to be expected and again implies a more concrete evaluation of simple FDLM-based measurement is required. Furthermore, the loss in accuracy is tolerably low as our architecture still produces errors comfortably meeting our mean upper bound of $< 10\%$.

Another interesting and important statistic to consider is the mean error across all test

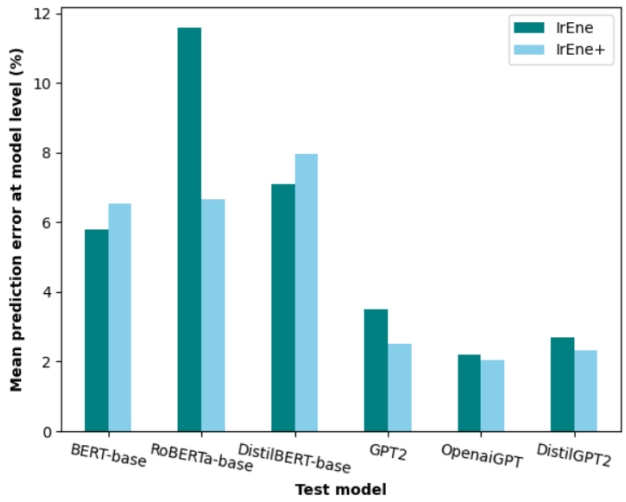


Figure 4.6: Comparison of the error produced by IrEne and IrEne$^+$ relative to hardware ground truth measured on **PC1** at the **model** level.

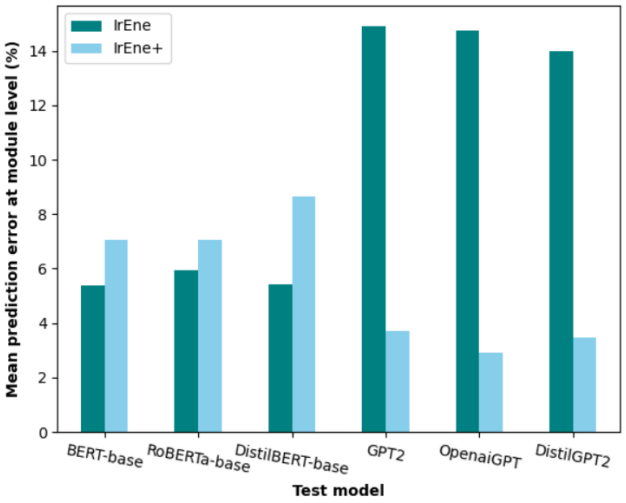


Figure 4.7: Comparison of the error produced by IrEne and IrEne$^+$ relative to hardware ground truth measured on **PC1** at the **module** level.

models at both the model and module levels. Figures 4.6 and 4.7 demonstrate the error of IrEne$^+$ was sometimes better, sometimes worse. However, the mean errors across all test models at the model and module levels of the tree abstraction were 4.68% and 5.25% respectively. These figures both represent improvements relative to the same numbers for IrEne, particularly at the module level which saw an improvement of over 3% on average. Therefore, the earlier conclusion in S4.2.3 that IrEne$^+$ outperforms its predecessor is strengthened. However, asserting this more convincingly required further evaluation across hardware platforms.

Unfortunately, we were not able to perform another similar analysis for two reasons. First, we did not have access to a hardware power monitor. Therefore we were unable to construct hardware ground truth data sets of our own and had to rely on those provided by the authors of IrEne [8]. Furthermore, the second system used in the evaluation of

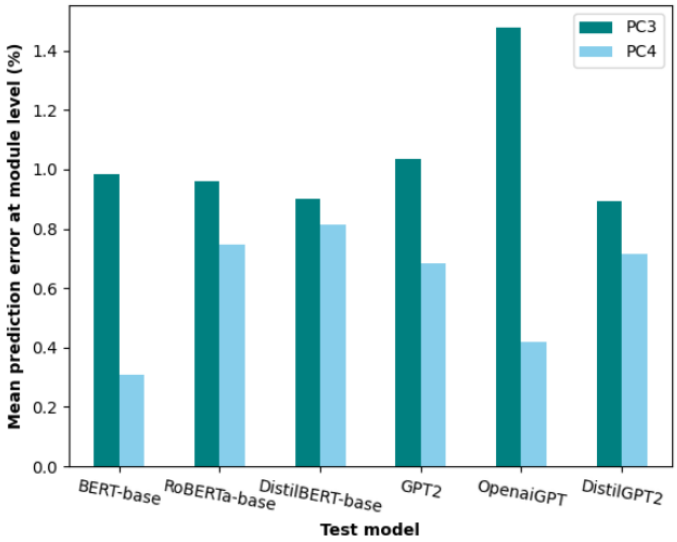


Figure 4.8: Mean error observed in IrEne^{+} predictions at the model level on **PC3** and **PC4**, relative to ground truth measured using a simple FDLM-based approach.

IrEne, **PC2**, was vastly different in specification to our second system, **PC4**. Therefore, an evaluation similar to the one just described was infeasible. The best we could do was to repeat the analysis just performed using the **PC4** evaluation data set. However, when it came to testing time on each cross-validation fold, we had to use data from the same software-based evaluation data set in place of hardware ground truth. To at least give us something to compare to, we did the same with the **PC3** evaluation data. We accept this method is much weaker, but feel it is still relevant and provides some useful insight.

Clearly, as seen in 4.8 this secondary evaluation produced a very low error relative to the software ground truth across the board. Therefore, we can say that our training approach appears to produce training data of consistent quality across different hardware devices. Furthermore, it seems reasonable to assume that the error between simple FDLM-based measurement and hardware ground truth is similar from system to system. If so, then the previous observation indicates that evaluating against hardware ground truth measured on **PC4** would present a similar level of error to that seen in Figures 4.6 and 4.7. Of course, these conclusions are weak and based on assumptions. However, this makeshift, secondary evaluation at least seems to indicate that our architecture is cross-platform.

To surmise, this chapter presents an in-depth evaluation of the IrEne^{+} prediction architecture at varying levels. First, we evaluated various design decisions in isolation. The main conclusions here were that unstructured measurement was generally a highly accurate energy estimation approach across a range of feature sets, and that simple FDLM-based measurement provided an accurate estimate of hardware monitor ground truth, given sufficient time. Then, we performed a broader analysis of our architecture as a whole and found it to be accurate on one system and seemingly so on another. However, our lack of access to our own hardware limited the evaluation possible both here and with respect to the accuracy of simple FDLM-based measurement. Therefore, further work is required to confirm our findings.

Chapter 5

Conclusion

5.1 Contributions

In conclusion, we have extended the state-of-the-art approach to estimating the energy consumed by a model inference, IrEne [7], to be cross-platform. We primarily do so by replacing the hardware power monitor required by its training process with a simple, FDLM-based approach. This approach estimates ground truth energy consumption as a simple sum of the measurements made by widely available, low-level software power monitors [19, 23], making it highly cross-platform. However, it requires a sufficiently long period of measurement to produce accurate estimates. So, modeling techniques like IrEne^{+}, which do not require data from these interfaces to make predictions, are still necessary to provide fine-grained measurement.

Furthermore, we designed the training process of IrEne^{+} to significantly reduce the associated overhead relative to its predecessor. A driving factor behind research into better energy estimation techniques in the domain of machine learning is to facilitate a reduction in its significant energy footprint [30]. Therefore, it is important that energy estimation techniques themselves are lightweight, so as to not contradict this objective.

First, we reduced the set of independent variables required for prediction to just two. By the one-in-ten rule [16, 17], this led to a reduction in the required size of training sets, so it costs less time and energy to obtain them. One of IrEne's features, *gpu_energy*, was discarded as it cannot be accurately profiled over a single model inference. A further nine were found to provide little information when making energy estimations and so were also pruned. However, we accept that further work is needed to establish if our two features are sufficient across different system loads.

Similarly, we reduced the amount of time our simple FDLM-based approach required to obtain each training sample. IrEne requires a period of 20 seconds per sample, serving as an arbitrarily 'long-enough' duration to handle the relatively high temporal resolution of their power monitor. Similarly, our training data procurement method, based on data from coarse software interfaces, also requires some degree of prolonged execution time to produce accurate samples. However, 20 seconds seemed redundant. To ensure no unnecessary overhead was introduced, we investigated the relationship between the

period of measurement and the accuracy of simple FDLM-based measurement. We observed that the amount of time required varied from system to system, depending on factors such as power lags [6]. Therefore, we decided on a period of 10 seconds per sample to provide a sufficient buffer to ensure compatibility with a range of systems, while also yielding a 50% reduction relative to IrEne. Further work across a more diverse range of hardware is needed to establish if this period can be optimised further.

Finally, we made a major change to the way in which energy estimates were actually produced. In IrEne, the energy consumption of each leaf node is predicted using a node-specific multiple regression model. Then, every other node's cost is estimated as a weighted sum of the costs of its children, using weights produced by a weight model shared across all nodes. The authors of IrEne considered an alternative, unstructured approach, where one multiple regression model would be used to directly predict the energy consumed by each tree node. However, they evaluated this as highly inaccurate [7], so decided on their tree-based approach instead. We found that this evaluation was critically flawed and unstructured measurement proved to be more accurate than IrEne's recursive estimation. Therefore, IrEne$^+$ makes predictions in an unstructured fashion.

5.2 Future work

IrEne$^+$, like its predecessor, estimates energy consumed by model inferences, so extension to measure training energy is a crucial direction for future work. Although measurement of inference energy is useful in a deployment context, the compute required to train cutting-edge models has been growing exponentially [1]. Therefore, techniques are required to accurately estimate the energy cost of training processes to facilitate improvements in efficiency and reason about accuracy-energy trade-offs. We believe that our approach is applicable in a training context. The energy consumed by a single forward and backward pass could be estimated using our architecture, and then scaled by the total number in the training process to estimate total energy without requiring training to run to completion. However, this assumes that each pass has a constant energy cost. Further work is required to determine if this holds.

Furthermore, we have already discussed how further study of the strength of resource utilisation features as predictors of energy consumption is required. In §3.4.2, we observed these features to provide little information when estimating energy, and so pruned them from our architecture. While this decision was valid based on the data provided [8], we later observed that the system load appeared to be constant. Unfortunately, we did not have time to explore this in depth ourselves, due to other parts of the project becoming unexpectedly time-intensive. Therefore, future work is needed to construct more comprehensive evaluation data sets across a range of system loads to determine whether our two features are sufficient in a broad sense.

Finally, further evaluation of IrEne$^+$ across a diverse range of hardware platforms. The observations in §4.3 are promising and we are confident our architecture should exhibit similar performance on other systems. However, we lacked access to a hardware power monitor to construct our own ground truth data sets and so the depth of evaluation possible was limited.

Bibliography

[1] Dario Amodei. AI and compute. https://openai.com/blog/ai-and-compute/, Jun 2021.

[2] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *CoRR*, abs/2007.03051, 2020.

[3] Yehia Arafa, Ammar ElWazir, Abdelrahman Elkanishy, Youssef Aly, Ayatelrahman Elsayed, Abdel-Hameed A. Badawy, Gopinath Chennupati, Stephan J. Eidenbenz, and Nandakishore Santhi. Verified instruction-level energy consumption measurement for NVIDIA gpus. In Maurizio Palesi, Gianluca Palermo, Catherine Graves, and Eishi Arima, editors, *Proceedings of the 17th ACM International Conference on Computing Frontiers, CF 2020, Catania, Sicily, Italy, May 11-13, 2020*, pages 60–70. ACM, 2020.

[4] Daniel Bedard, Robert Fowler, Min Yeol Lim, and Allan Porterfield. Powermon 2: Fine-grained, integrated power measurement renci technical report tr-09-04. 2009.

[5] Robert A. Bridges, Neena Imam, and Tiffany M. Mintz. Understanding GPU power: A survey of profiling, modeling, and simulation methods. *ACM Comput. Surv.*, 49(3):41:1–41:27, 2016.

[6] Martin Burtscher, Ivan Zecena, and Ziliang Zong. Measuring GPU power with the K20 built-in sensor. In John Cavazos, Xiang Gong, and David R. Kaeli, editors, *Seventh Workshop on General Purpose Processing Using GPUs, GPGPU-7, Salt Lake City, UT, USA, March 1, 2014*, page 28. ACM, 2014.

[7] Qingqing Cao, Yash Kumar Lal, Harsh Trivedi, Aruna Balasubramanian, and Niranjan Balasubramanian. Irene: Interpretable energy prediction for transformers. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2145–2157. Association for Computational Linguistics, 2021.

[8] Qingqing Cao and Yash Kumar Lal Harsh Trivedi. Irene source code. https://github.com/StonyBrookNLP/irene, 2021.

[9] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. RAPL: memory power estimation and capping. In Vojin G. Oklobdzija,

Barry Pangle, Naehyuck Chang, Naresh R. Shanbhag, and Chris H. Kim, editors, *Proceedings of the 2010 International Symposium on Low Power Electronics and Design, 2010, Austin, Texas, USA, August 18-20, 2010*, pages 189–194. ACM, 2010.

[10] David DeBonis, James H Laros, and Kevin Pedretti. Qualification for powerinsight accuracy of power measurements. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2013.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[12] David Gefen and Detmar W Straub. The relative importance of perceived ease of use in is adoption: A study of e-commerce adoption. *Journal of the association for Information Systems*, 1(1):8, 2000.

[13] Isabelle Guyon and Andre Elisseeff. An introduction to feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[14] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. An energy efficiency feature survey of the intel haswell processor. In *IPDPS Workshops*, pages 896–904. IEEE Computer Society, 2015.

[15] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. Measuring energy consumption for short code paths using RAPL. *SIGMETRICS Perform. Evaluation Rev.*, 40(3):13–17, 2012.

[16] Frank E Harrell Jr, Kerry L Lee, Robert M Califf, David B Pryor, and Robert A Rosati. Regression modelling strategies for improved prognostic prediction. *Statistics in medicine*, 3(2):143–152, 1984.

[17] Frank E Harrell Jr, Kerry L Lee, and Daniel B Mark. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15(4):361–387, 1996.

[18] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *CoRR*, abs/2002.05651, 2020.

[19] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in action: Experiences in using RAPL for power measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 3(2), mar 2018.

[20] Jens Lang and Gudula Rünger. High-resolution power profiling of GPU functions using low-resolution measurement. In Felix Wolf, Bernd Mohr, and Dieter an Mey, editors, *Euro-Par 2013 Parallel Processing - 19th International Conference, Aachen, Germany, August 26-30, 2013. Proceedings*, volume 8097 of *Lecture Notes in Computer Science*, pages 801–812. Springer, 2013.

[21] James H Laros, Phil Pokorny, and David DeBonis. Powerinsight-a commodity power measurement capability. In *2013 International Green Computing Conference Proceedings*, pages 1–6. IEEE, 2013.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[23] NVIDIA Corporation. Nvidia management library (nvml) api reference guide. `https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf`, 2021.

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[25] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[26] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[27] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020.

[28] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *IJCNN*, pages 1–8. IEEE, 2022.

[29] Stanford University. The AI index report 2022 – artificial intelligence index. `https://aiindex.stanford.edu/report/`.

[30] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics, 2019.

[31] Masaaki Terai, Fumiyoshi Shoji, Toshiyuki Tsukamoto, and Yukihiro Yamochi. A study of operational impact on power usage effectiveness using facility metrics and server operation logs in the K computer. In *IEEE International Conference on Cluster Computing, CLUSTER 2020, Kobe, Japan, September 14-17, 2020*, pages 509–513. IEEE, 2020.