AMPERE: A Generic Energy Estimation Approach for On-Device Training

Jiaru Zhang[1] Zesong Wang[1] Hao Wang[2] Tao Song[1] Huai-an Su[3]
Rui Chen[3] Yang Hua[4] Xiangwei Zhou[5] Ruhui Ma[1] Miao Pan[3]
Haibing Guan[1]✉
[1]Shanghai Jiao Tong University [2]Stevens Institute of Technology
[3]University of Houston [4]Queen's University Belfast [5]Louisiana State University
{jiaruzhang, taihaozesong, songt333, ruhuima, hbguan}@sjtu.edu.cn,
hwang9@stevens.edu, {hsu3, rchen19, mpan2}@uh.edu, Y.Hua@qub.ac.uk, xwzhou@lsu.edu

ABSTRACT

Battery-powered mobile devices (e.g., smartphones, AR/VR glasses, and various IoT devices) are increasingly being used for AI training due to their growing computational power and easy access to valuable, diverse, and real-time data. On-device training is highly energy-intensive, making accurate energy consumption estimation crucial for effective job scheduling and sustainable AI. However, the heterogeneity of devices and the complexity of models challenge the accuracy and generalizability of existing methods. This paper proposes AMPERE, a generic approach for energy consumption estimation in deep neural network (DNN) training. First, we examine the layer-wise energy additivity property of DNNs and strategically partition the entire model into layers for fine-grained energy consumption profiling. Then, we fit Gaussian Process (GP) models to learn from layer-wise energy consumption measurements and estimate a DNN's overall energy consumption based on its layer-wise energy additivity property. We conduct extensive experiments with various types of models across different real-world platforms. The results demonstrate that AMPERE has effectively reduced the Mean Absolute Percentage Error (MAPE) by up to 30%. Moreover, AMPERE is applied in guiding energy-aware pruning, successfully reducing energy consumption by 50%, thereby further demonstrating its generality and potential. [1]

1. INTRODUCTION

Artificial Intelligence (AI) applications are shifting from *wall-plug-powered AI devices* to *battery-powered mobile AI systems*, such as smartphones, tablets, laptops, AR/VR wearables, and IoT devices. Those mobile AI systems/devices are becoming increasingly powerful and provide easy access to a rich source of diverse, real-time data, making them highly valuable for training AI models. Nascent on-device training is gaining traction due to its data privacy and efficiency benefits, as it circumvents the need to upload data to the cloud. Furthermore, it is especially advantageous in situations with unreliable network connectivity or for providing personalized services [1] . However, given the fact that almost all contemporary mobile devices are powered by battery, their energy consumption is commonly constrained. Therefore, deploying DNN training onto these devices presents a challenge. It is a high-power task and requires estimating the overall energy expenditure beforehand to prevent power failure . Addressing these energy constraints is critical for developing sustainable AI, promoting efficient energy use and environmental friendliness.

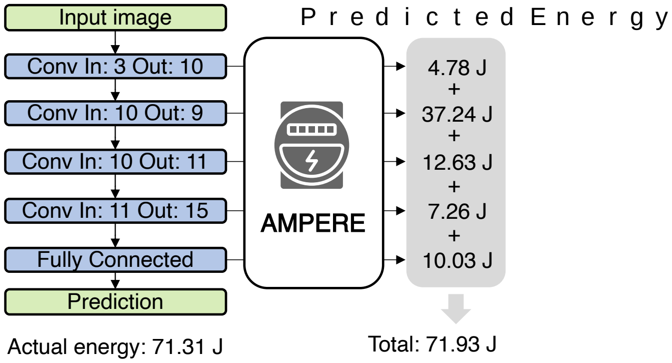


Figure 1: Illustration of the energy estimation process on a 5-layer CNN. In: a and Out: b indicate the input channel a and output channel b for convolutional layers.

State-of-the-art solutions may not be sufficiently accurate for estimating DNN training's energy consumption. Evaluating a neural network model's Floating Point Operations (FLOPs) is a common energy-estimation practice for most models , based on an assumption that the devices' working status is fixed. However, in reality, the energy estimation based on FLOPs may deviate greatly from the observations. Beyond FLOPs, most simulation-based methods are limited to specific hardware and frameworks . Developing an accurate and generic energy estimation approach for on-device training presents the following fundamental challenges:

- **System Heterogeneity**. Mobile devices have diverse hardware combinations, leading to potential variations in behavior even within the same model. Furthermore, the Lookup Table provided by Neural Architecture Search (NAS) proves impractical for making estimations across a wide range of devices due to this variability .
- **Model Diversity**. DNN models' energy consumption characteristics can vary greatly, inherently resulting in heterogeneous resource demands. Some models may primarily rely on computational power as input scale increases, while others remain largely power-insensitive.
- **Runtime Complexity**. It arises from optimization techniques such as kernel fusion and CPU hand-off during model train-

[1] A version with appendix available at arxiv.org/pdf/2501.16397. Correponding author: Haibing Guan. This work was supported by the National Key R&D Program of China (2022YFB4402102), and the Shanghai Key Laboratory of Scalable Computing and Systems. The work of Hao Wang was supported in part by NSF 2534286.

ing. The unpredictable access patterns from various levels of the memory hierarchy impede the ability to calculate energy usage merely as the sum of data movement and computation.

In this paper, we present AMPERE, a novel and generic method designed to provide precise energy estimates. It is agnostic to both the running platform and the DNN model, therefore solves the challenges of system heterogeneity and model diversity. Based on the observations that layer-associated operators are emitted sequentially and the inter-layer effects are insignificant, we present the layer-wise energy *additivity* and *subtractivity*. The total energy consumption of a DNN is estimated by summing the costs of its layers, with the cost of each layer derived by subtracting the residual layers' costs from the total. The energy consumption data for individual layers is gathered through layer-wise subtractivity. Predictive GP models are fitted with these data, hence avoiding the challenge associated with runtime complexity. Lastly, the total energy consumption of the entire model can be obtained from summation through layer-wise additivity. Gaussian Process (GP) models are fitted using the observed energy consumption of individual layers. This fitting process is a one-time endeavor as the resulted models are reusable. Then, the total energy consumption can be estimated by summing the estimated energy costs of all layers.

We deploy and evaluate AMPERE across multiple models on five different devices. Compared to the current leading approaches, AMPERE reduces the Mean Absolute Percentage Error (MAPE) by up to 30%. Moreover, we use energy-conscious model pruning to create a leaner architecture with the same performance and 50% energy consumption, which further verifies the effectiveness and practicality of AMPERE. The codes will be publicly released if the paper is accepted.

Our main contributions can be summarized as follows:

- We study the energy consumption characteristics and find the promising layer-wise *additivity* of energy consumption during DNN training.
- Based on the observations above, we design AMPERE, a generic method for accurate estimations of energy consumption for training of DNN models.
- We implement AMPERE and conduct tests on various AI systems. The results indicate a reduction in MAPE by up to 30%. Moreover, we apply it to guide model pruning and successfully reduce 50% energy consumption.

2. BACKGROUND AND MOTIVATION

2.1 Modeling Deep Learning's Energy Consumption

DNN training fundamentally encompasses two primary stages: *forward* propagation and *backward* propagation. During the process of *forward* propagation, the model takes the input data and calculates the output for each layer. On the other hand, *backward* propagation involves evaluating the difference between the model's output and the target label using a loss function. Subsequently, derivatives are computed in a backward manner, leveraging the chain rule, to guide the update of model parameters. Each of the aforementioned processes necessitates computational operations, thereby consuming a corresponding amount of energy.

Modeling deep learning energy consumption is a widely studied topic. Energy consumption essentially stems from computation and data movement, hence the majority of existing studies primarily focus on estimating this consumption by considering the number of FLOPs inferred from a DNN's framework [9, 10] . In these methods, the Floating Point Operation per Second (FLOPS) and FLOPS per watt serve as indicators for computational performance and energy efficiency, respectively. we refer to our Appendix Sec. A2 for more thorough exploration of this literature.

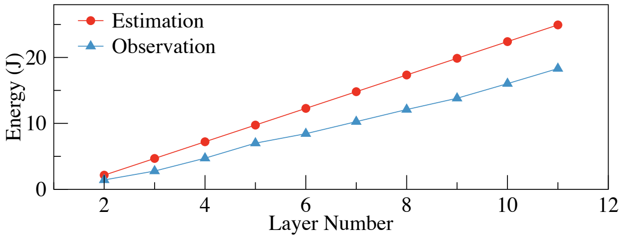


Figure 2: Energy consumption from NeuralPower estimation and from observation for a CNN.

2.2 Existing Methods' Limitations

Existing methods fall into three categories: **Proxy-based methods** estimate energy costs by a model's FLOPs, parameter size, and number of layers, with FLOPs-based estimation being the most common . While adaptable to any model, these methods overlook device heterogeneity and runtime optimizations, assuming stable computation performance and performance-per-watt. However, when the model structure changes, the system utilization will undergo significant changes. The kernel configure tends to launch fewer threads for pruned models [7]. Upon compilation, the framework generates both forward and backward computational graphs and fuse operations into a single CUDA kernel. This approach enhances computation for activation functions, optimizers, custom RNN cells, *etc*. Some in-place optimizations, such as `Convolution-BatchNorm-ReLU` fusion, are also implemented and make the execution more like a black box [3] . These factors contribute to inaccuracies in Proxy-based estimations. **Simulation-based methods** simulate the full computation and data movement process [5, 9] . They require detailed knowledge about the algorithm implementations as well as the energy cont of each hardware component. These methods allow for the identification of the hardware that incurs the highest energy consumption and help in locating performance bottlenecks caused by memory stalls. However, this type of approach loses its generality and is only applicable to specific devices, models, and Machine Learning (ML) frameworks. **Architecture-based methods** utilize framework-provided profilers during the inference phase to obtain the execution time for specific layers or kernels, significantly improving the accuracy of energy prediction [10, 2, 6] . Nevertheless, this type of approach relies on specific framework and still faces challenges in obtaining energy costs. As a validation, we extend the forward pass to the whole training process like NeuralPower [2] and adapt it to the training phase as shown in Fig. 2. We conduct profiling on the operators involved in each of these stages separately and obtain the final energy by summing them up. The results show that this method tends to overestimate the cost of each layer, which indicates the introduction of systematic biases and verifies our analysis.

3. AMPERE'S DESIGN

3.1 A Bird's-Eye View

Fig. 3 illustrates an overview of AMPERE. It broadly contains three processes listed below, where the profiling and the fitting processes are carried out in an iterative manner.

Profiling: In AMPERE, all layers are parsed into input layer, hidden layer, and output layer. To estimate the energy consumption of these layers, AMPERE generates 1-layer, 2-layer, and 3-layer variant NNs with different parameters, respectively. Initially, the parameters of variant models are selected as the bound value, and

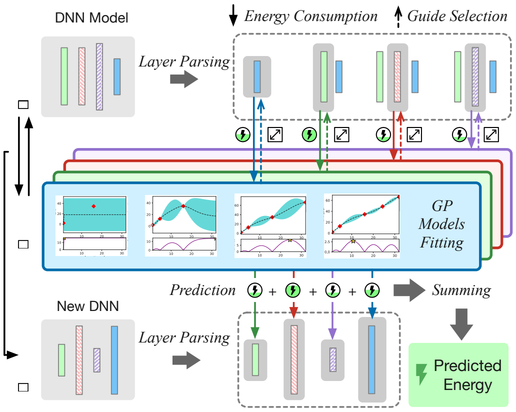


Figure 3: An overview of AMPERE.

the subsequent parameters are selected and guided by the profiling stage. Training these variant models provides information about their consumption.

Fitting: Based on the *additivity* and *subtractivity* of the energy consumption, AMPERE separates the model as different layers and fits with Gaussian Process (GP) models based on the consumption data from the profiling stage. Moreover, the GP models guide the selection of the next profiling point based on predictive uncertainty and utilize the returned data to optimize the prediction. Therefore, the fitting stage can be seen as an active learning process to some extent.

Estimation: Once sufficient energy consumption data are obtained, GP models are fitted to predict the energy consumption of each layer. After that, AMPERE can obtain the energy estimation by summing the energy from each GP model based on the additivity of the energy consumption.

The key advantages of our framework are its inherent generality and minimal overhead. By implementing the actual training workflow without the need for an additional operator-level profiler, AMPERE can be easily integrated into any training framework and device to provide accurate estimations. In essence, AMPERE offers a highly adaptable and low-overhead solution for a wide range of use cases across diverse computing environments.

3.2 Profiling

Layer-wise Energy Additivity of DNN. In the energy cost estimation task, the existing method, NeuralPower [2], estimates energy costs by profiling the forward, backward, and update stages. However, this approach tends to overestimate costs due to data reuse during the DNN training process. In contrast, we view the DNN model as a combination of different layers and propose that the total energy consumption of the entire DNN can be obtained by summing the costs of each layer, a concept we refer to as layer-wise *additivity*. This also implies that the cost of a single layer can be independently assessed by subtracting the cost of the residual layers from the total sum, a concept we call layer-wise *subtractivity*.

To validate this, we train a Convolutional Neural Network (CNN) model on the MNIST dataset, starting with a rudimentary model including only an input layer and an output (i.e., `FC`) layer. After this, we integrate identical `Conv2d` layers into the existing structure and examine the resulting changes in energy consumption. Experimental results in Fig. 2 empirically illustrate that existing method overestimates the costs. Moreover, the trends in energy consumption corresponding to the increase in the number of layers. This underscores that the incremental cost associated with each convolutional layer remains roughly constant, creating a linear trajectory. This observation suggests that the energy consumption of each layer is additive and the positioning of a layer has no impact on the final outcome.

Layer Parsing. Based on the presented layer-wise energy additivity pf DNN, we dissect the network model into sub-components of the three distinct categories: input layers, hidden layers and output layers. A DNN usually contains a single input layer, a single output layer, and multiple hidden layers organized with *Modular Design*, i.e., repeating layers or blocks of layers. Therefore, we employ a similar strategy to partition them into blocks, where non-parametric layers are consistently grouped together with preceding layers. Deduplication is carried out based on the layer type and the associated hyperparameters, *i.e.*, input height and weight, kernel size, and batch size. This separation rule can hide the effects of optimization by the framework, therefore, making the estimation more accurate. The details of our layer characteristics are further described in Appendix Sec. A3.

In AMPERE, layers are characterized by their output channels (for input layers), input channels (for output layers), or both input and output channels (for hidden layers). For an n-layer DNN, the channels are denoted as C_1, C_{n-1}, and C_i for $i = 2, \cdots, n-2$, respectively. Note that layers with different kernel sizes, steps, and batchsizes are encoded as different layers since their energy cost patterns have a large gap.

Profiling Process. In AMPERE, we firstly profile the output layer, treating it as a complete model for training purposes. The energy consumption of this single-layer model is denoted as $E_{\mathit{output}}(C_1)$ and we fit a GP model in the fitting process to predict the energy consumption denoted as $\hat{E}_{\mathit{output}}(C_1)$ with varied C_1s.

Secondly, the input layer's energy cost can be obtained by subtracting the output layer's estimation result from the total measured cost according to the layer-wise subtractivity:

$$E_{\mathit{input}}(C_2) = E_{\mathit{input+output}}(C_1, C_2) - \hat{E}_{\mathit{output}}(C_1), \qquad (1)$$

and we fit a GP model in the fitting process to predict the energy consumption denoted as $\hat{E}_{\mathit{input}}(C_2)$ for varied C_2s.

Lastly, we assemble the energy consumption of NNs containing an input layer, a single hidden layer, and an output layer with varied C_1s and C_2s. Therefore, the energy costs of the hidden layer can be obtained by subtracting estimated costs of others:

$$E_{\mathit{hidden}}(C_1, C_2) = E_{\mathit{model}}(C_1, C_2) - \hat{E}_{\mathit{input}}(C_1) - \hat{E}_{\mathit{output}}(C_2), \qquad (2)$$

and we fit GP models in the fitting process to predict the energy cost of each kind of hidden layers denoted by $\hat{E}_{\mathit{hidden}}(C_1, C_2)$.

It is noteworthy that the profiling is guided by the fitting process instead of randomly sampling, as shown in detail below. It improves the efficiency of the whole framework.

3.3 Fitting

Gaussian Process Model. A Gaussian Process (GP) model is a non-parametric, probabilistic model popularly used in machine learning and statistics. In AMPERE, we employ GP as the model to fit the energy consumption characteristics of each kind of layers. GP solely demands the consumption data for fitting, making it align well with the diverse requirements posed by system heterogeneity and model diversity. It is also capable of handling noise, which is unavoidable due to the potential awakening of background processes in practice. Moreover, its probabilistic nature provides empirical confidence intervals, allowing adaptive data fitting and assisting in determining termination conditions.

Fitting Accurately. The GP models are fitted with energy consumption data from the profiling process. As illustrated in Fig. 5, an increase in the input channel directly correlates to a linear increase

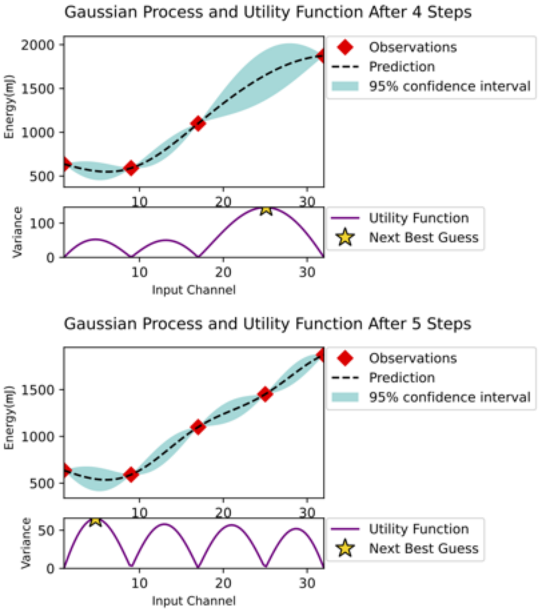


Figure 4: GP after 4 and 5 steps for `FC` layer on OPPO taking 500 batches of (10, input channel, 28, 28) input.

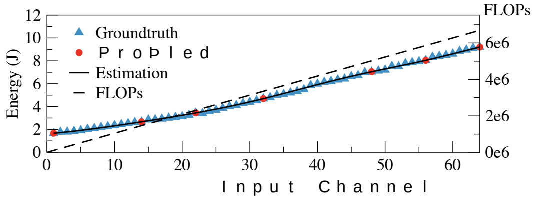


Figure 5: Energy consumption of a Fully Connected (FC) layer taking 500 batches of (4, input channel, 50, 50) input on Xavier.

in FLOPs. However, the energy consumption does not follow the same linear pattern. Hence, utilizing FLOPs to estimate the energy could result in inaccurate performance predictions, while the GP model aptly fits the energy prediction task.

Kernel Selection. In GP, the prior's covariance is given by a kernel, which can describe how similar two neighbor points are. In AMPERE, we utilize the Matérn kernel [8]:

$$k\left(x_i, x_j\right)=\frac{\left(\frac{\sqrt{2 \nu}}{l} d\left(x_i, x_j\right)\right)^\nu K_\nu\left(\frac{\sqrt{2 \nu}}{l} d\left(x_i, x_j\right)\right)}{\Gamma(\nu) 2^{\nu-1}}, \quad (3)$$

where $d(\cdot,\cdot)$ is the Euclidean distance, $K_\nu(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function. This kernel is generally robust to misspecification of the length-scale parameter [8]. It has an additional parameter ν to control the smoothness of the resulting function. Considering the runtime optimization and cache thrashing, we choose $\nu = 2.5$ which only requires twice differentiable. Experiments provided in Appendix Sec. A6.2 support the superiority of the Matérn kernel.

Guided Profiling. GP has the unique ability to model the uncertainty hence can guide the selection of the next point in the profiling process. As our main purpose is to obtain accurate estimation, we choose the point with the largest variance to eliminate the uncertainty. Fig. 4 illustrates how GP selects the next point and fits the data. After fitting the point with the largest variance, the uncertainty is diminished. Therefore, it can be seen as an instance of active learning to some extent, where the acquisition function guides the selection of the most informative points for fitting in the next step.

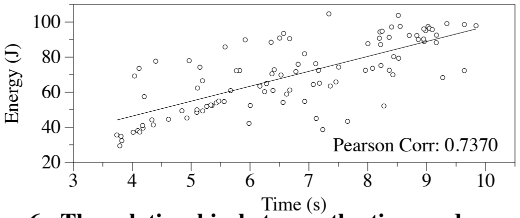


Figure 6: The relationship between the time and energy consumption for 5-layer CNN. They have an obvious positive relationship.

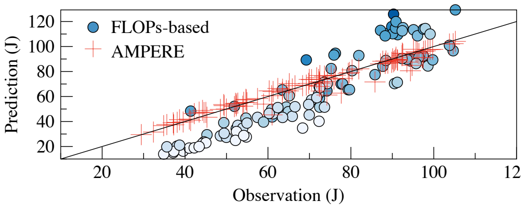


Figure 7: Estimation results of FLOPs-based method and AMPERE for a 5-layer CNN.

However, the real-time energy consumption acquisition is challenging in certain devices like smartphones. As a solution, we argue there exists a direct correlation between time consumption and energy consumption. It is verified by experiments conducted on a 5-layer CNN model, as illustrated in Fig. 6. Therefore, we utilize the time uncertainty as a practical surrogate for energy uncertainty to guide profiling.

Starting Points and End Condition. To cover the full range of channels, we use the the upper and lower bounds as the starting points. On the other hand, as GP is designed for the continuous situation while the channel number is discrete, the whole process may fall into a dead loop and fails to converge. Therefore, we set two end conditions to prevent the worst case: When the number of profiled points exceeds the limits or when the variance is smaller than 5% of the profiled data, we terminate the profiling and fitting process.

3.4 Estimation

After finishing the profiling and fitting process, we obtain the GP models for energy cost estimation of each kind of DNN layers. According to the layer-wise energy additivity, we dissect any given $n-$layer DNN model model into three distinct sub-components: input layers, hidden layers and output layers, based on the layer parsing rules described in Sec. 3.2. Hence, the total energy consumption of the DNN can be estimated by summing the estimated energy costs of all layers:

$$\hat{E}_{model} = \hat{E}_{input}(C_1) + \sum_{i=2}^{n-1} \hat{E}_{hidden}(C_{i-1}, C_i) + \hat{E}_{output}(C_{n-1}), \quad (4)$$

where $\hat{E}_{input}$, $\hat{E}_{hidden}$, and $\hat{E}_{output}$ represent GPs to estimate energy costs of input, hidden, and output layers, respectively.

As discussed in Sec. 1, system heterogeneity and model diversity result in significant variations in energy consumption across different systems. Therefore, GP models trained on one device or system cannot be directly transferred to estimate energy consumption on other devices or systems. Nevertheless, our AMPERE framework is generic and broadly applicable, as demonstrated in Sec. 4.

4. EVALUATION

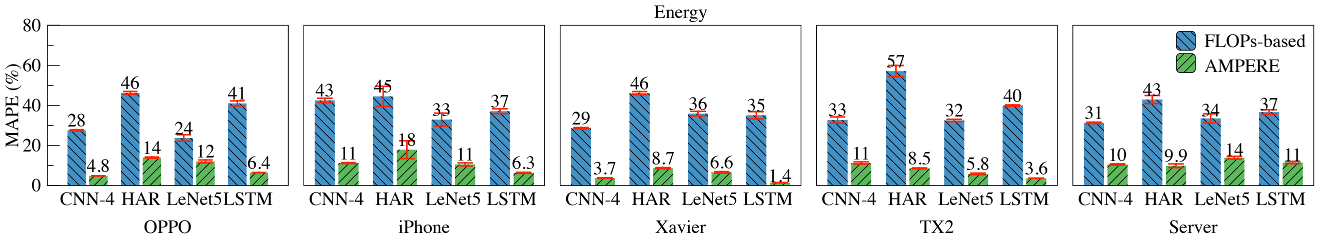


Figure 8: End-to-end energy estimation for five devices.

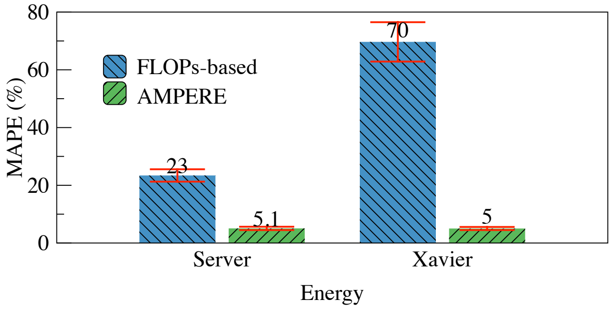


Figure 9: Energy estimation of Transformer.

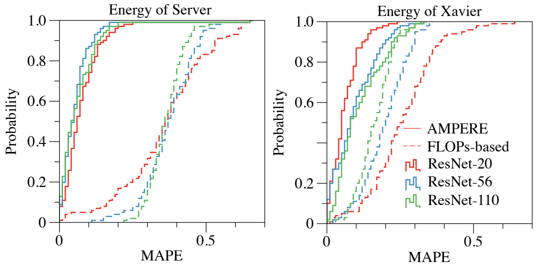


Figure 10: ResNet evaluation on Server and Xavier.

We conduct an evaluation of AMPERE using multiple representative models on five distinct, realistic devices, including OPPO, iPhone, Xavier, TX2, and Server. Details about the data, model architectures, devices, and implementation are provided in Appendix Sec. A5. AMPERE significantly decreases the relative errors, with reductions of up to 30% when compared to previous studies. We also apply AMPERE in a case study focusing on energy-aware model pruning, where it demonstrates a remarkable ability to reduce energy consumption by 50% without compromising the level of accuracy.

4.1 End-to-End Estimation Evaluation

In real-world scenarios, the estimation should be capable of handling unseen models, such as new architectures and parameters. To comprehensively evaluate the performance of AMPERE, we randomly sample the DNN architectures across channels ranging from 1 to the original channel. For the Transformer model, we randomly sample the number of encoder layers and hidden dimensions.

Intuitive Comparison. To visualize the strengths of AMPERE compared to the FLOPs-based method, we illustrate the energy consumption for a 5-layer CNN in Fig. 7. In the experiment, we generate 100 models with different channels using random sampling. Alignment with the line indicates an accurate result. This reveals that the FLOPs-based method neglects system utilization changes; hence, it tends to overestimate when FLOPs are lower and underestimate otherwise. In contrast, AMPERE maintains high accuracy across all ranges.

Table 1: Time cost (sec) of profiling and fitting.

	LeNet5	5-layer CNN	HAR	LSTM
OPPO	694	1688	2188	1615
iPhone	1201	1012	2446	1168
Xavier	184	421	740	1145
TX2	285	1211	4433	422
Server	235	268	562	436

Quantitative Comparison. As illustrated in Fig. 8, our approach has successfully reduced the MAPE from an average of approximately 40% to around 10%. This significant decrease in MAPE suggests that our method provides more accurate and stable results. The performance of the final estimations varies across devices because of their inherent heterogeneity. Among the tested devices, the Jetson series, which allows for a fixed frequency, exhibits the most favorable results. The estimations for various models on smartphones show a degree of disparity, with some cases, like the HAR model, demonstrating larger errors. This might be due to the influence of Dynamic Voltage and Frequency Scaling (DVFS) policies and power throttling effects. On the Server, predictions for different models are relatively consistent, though they have higher error rates compared to other devices.

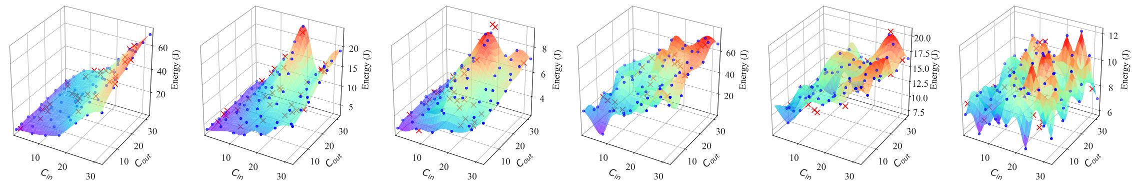
Time Costs. Tab. 1 presents the time costs for AMPERE profiling and fitting for various DNNs. Most of these tasks are completed within 20 minutes, demonstrating both the efficiency and practical feasibility of the AMPERE method.

More Results on Transformers. To further demonstrate the superiority of AMPERE, we present the energy estimation results for the Transformer architecture using both AMPERE and FLOPs-based methods in Fig. 9. Due to memory restrictions, only the Xavier and Server platforms can fully execute the Transformer model. AMPERE consistently outperforms FLOPs-based methods in energy estimation of the Transformer, further underscoring the effectiveness and generality of AMPERE.

CDF Plot of ResNets. To further assess the scalability of AMPERE, we conduct experiments on the ResNet family of models. Due to memory constraints, we can only execute these experiments on the Xavier and Server devices. During the profiling phase, we sample ten different layers based on existing rules in Sec. 3.2. The resultant Cumulative Distribution Function (CDF) plot is shown in Fig. 10. A step curve closer to the top-left corner indicates higher prediction accuracy. The results from both the Xavier and Server devices show improved performance compared to the FLOPs-based approach. Moreover, as the number of layers increases, the prediction accuracy does not appear to decrease.

4.2 Layer Characteristics

We present the energy consumption of `Conv2d` layers in the 5-layer CNN model, as they account for the majority of computational costs. The relevant sampling and estimation results across different devices are illustrated in Fig. 11, where H and W represent height and width, respectively, and C_{in} and C_{out} denote the input and output channels. The batch size is set as 10. We use additional random points to test the estimation results, and the dif-



(a) Xavier, H=W=42 (b) Xavier, H=W=21 (c) Xavier, H=W=10 (d) Server, H=W

Figure 11: Profiled v.s. estimated energy for Xavier and Server (The blue • represents the profiled samples, the red × represents the test samples, and the surface represents the estimated data).

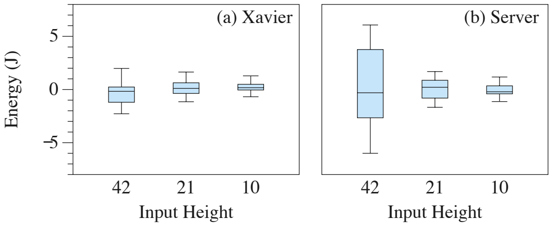


Figure 12: Differences of energy estimation and observation for Xavier and Server.

ferences are shown in Fig. 12. The energy consumption exhibits a non-linear variation with respect to the input and output channels. In these scenarios, the energy consumption is relatively smooth when H and W are large. For Xavier, the energy does not initially change significantly with the input channel and remains at a plateau. However, after C_{in} exceeds 20, there is a noticeable increase. When H and W are at their smallest values, the energy of Server fluctuates around the average, while Xavier demonstrates a distinct ridge in the middle. The power consumption characteristics of the Server are more complex, leading to poorer estimation results. Although the relative error may be small when the input size is large, it may still contribute the most to the final absolute error. We also compare the energy efficiency of these devices, with Xavier and Server exhibiting similar performance for these layers. Once we have identified these characteristics, we can compute the corresponding gradients. This allows us to more effectively guide the model pruning and architecture search, while avoiding extreme outcomes like increased energy consumption after pruning.

4.3 Case Study: Energy-Aware Pruning

To verify AMPERE meets the objective of being easy to combine with other methods, we undertake the task of gender identification from the real-world CelebA dataset on Xavier. The original model's total energy consumption stands at approximately 20,000J over 2,000 iterations. We simulate an energy-constrained environment where the available energy for consumption is curtailed to 50% of its initial level. To achieve this, we apply random pruning as suggested by Li et al. [4]. We incorporate AMPERE and FLOPs-based methods to guide the process until the energy consumption per iteration drops to 50% of its original value. As illustrated in Fig. 13, both methods effectively reduce the total energy consumption. However, only the AMPERE approach manages to keep the total energy consumption below the allotted budget (49.2%). The AMPERE method demonstrates superior performance by effectively keeping the total energy consumption within the defined budget. This result ensures efficient resource allocation and highlights the potential of AMPERE as a revolutionary tool in energy-constrained scenarios. Consequently, it presents AMPERE as a more viable solution compared to the traditional FLOPs-based method.

5. CONCLUSION

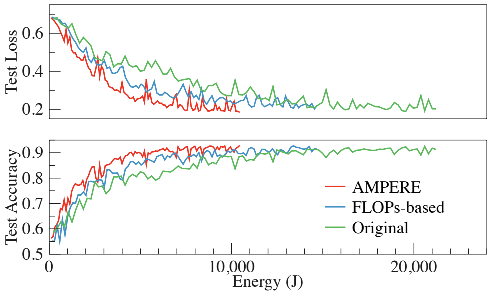


Figure 13: Test loss and accuracy for different models.

This paper proposes AMPERE, a generic method to estimate the energy consumption of DNN training. GP is used to fit layerwise consumptions, then the end-to-end estimation is obtained by summing the predictions of each layer based on the presented layer-wise energy additivity. We demonstrate the effectiveness of AMPERE for different DNN architectures and real-world devices. Moreover, AMPERE can be easily integrated into existing training frameworks to guide energy-aware job scheduling.

References

[1] Dongqi Cai et al. "Towards Ubiquitous Learning: A First Measurement of On-Device Training Performance". In: *EMDL Workshop*. 2021.

[2] Ermao Cai et al. "Neuralpower: Predict and Deploy Energy-efficient Convolutional Neural Networks". In: *ACML*. 2017.

[3] Zhihao Jia et al. "TASO: Optimizing Deep Learning Computation with Automatic Generation of Graph Substitutions". In: *SOSP*. 2019.

[4] Yawei Li et al. "Revisiting Random Channel Pruning for Neural Network Compression". In: *CVPR*. 2022.

[5] Linyan Mei et al. "A Uniform Latency Model for DNN Accelerators with Diverse Architectures and Dataflows". In: *DATE*. 2022.

[6] Dimitrios Stamoulis et al. "Hyperpower: Power- and Memory-Constrained Hyper-Parameter Optimization for Neural Networks". In: *DATE*. 2018.

[7] Yunsong Wang et al. "Time-Based Roofline for Deep Learning Performance Analysis". In: *IEEE/ACM Workshop on Deep Learning on Supercomputers*. 2020.

[8] Christopher KI Williams et al. *Gaussian Processes for Machine Learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[9] Tien-Ju Yang et al. "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning". In: *CVPR*. 2017.

[10] Li Lyna Zhang et al. "nn-Meter: Towards Accurate Latency Prediction of Deep-Learning Model Inference on Diverse Edge Devices". In: *MobiSys*. 2021.