Paper Reviews and Potential Next Steps

Hongyu Hè
hongyu.he@inf.ethz.ch

1 INTRODUCTION

In this document, I first review two relevant papers (§2, §3), briefly summarizing their respective strengths and weaknesses, along with my takeaways. Following the reviews, I motivate and describe two research questions in §4, with the aim of extending the insights derived from the these works.[1]

2 REVIEW OF HABITAT

In this section, I provide a brief review of Habitat [10], given my current level of understanding.

2.1 Brief Summary

Target problem. The paper addresses the challenge faced by deep learning researchers and practitioners in choosing a GPU for training their deep neural networks (DNNs). The problem is twofold: (i) there are many GPU options available, and (ii) users struggle to balance the competing concerns of maximizing compute performance while minimizing costs.

Key contributions.

(1) The paper introduces a new technique called Wave Scaling, which scales the execution time of a kernel measured on one GPU to a different GPU. The result is achieved by using scaled ratios between the number of compute units on each GPU and their memory bandwidths.
(2) The authors develop Habitat, a new library that utilizes Wave Scaling along with pre-trained multilayer perceptrons (MLPs) to predict the execution time of model training iterations on different GPUs.
(3) The authors evaluate Habitat across six different GPU architectures, demonstrating its effectiveness in making accurate iteration execution time predictions with an average error of 11.8% on various DNN models.

Main takeaways. Firstly, the paper presents a *practical technique* that leverages runtime information to assist users in making cost-efficient GPU selections for DNN training (**S2**). Secondly, the combination of analytical and predictive modeling methods (**S3**) makes Habitat both generally applicable and cost-efficient (compared to pure analytical/predictive methods).

2.2 Strength

Below, I describe the strong points (**S**s) of this work.

(**S1**) **Demonstrating the predictable dynamics of ML training workloads.** This study illuminates the inherently repetitive nature of ML workloads, underscoring their predictability. Notably, this predictability extends consistently across diverse model and device configurations, encompassing critical factors such as batch size and GPU architecture, which can substantially influence model quality.

(**S2**) **Notable monetary and environmental impacts.** This research holds timely significance, given that ML workloads represent some of the most resource-intensive jobs in the cloud, demanding substantial financial investments and consuming thousands of MWh of energy. The ability to determine the optimal performance configuration without relying on trial and error stands to substantially reduce both financial expenditures and environmental costs.

(**S3**) **Combining analytical and predictive models.** The authors employ the Wave Scaling method to model the performance of kernel operations of fixed implementations. This method cleverly utilizes the GPU-specific scheduling mechanism of thread blocks, offering an accurate and cost-effective analytical performance model that is applicable across various GPU types. In scenarios where the kernel's implementation is platform-specific, black-box predictive models are employed. This hybrid approach is pivotal for system modeling.

(**S4**) **Efficient generation of synthetic training data.** A central challenge in ML for systems is the initial scarcity of training data. Addressing this hurdle, the authors employ a cost-effective strategy to generate substantial amounts of training data by systematically varying the model and input configurations.

2.3 Weaknesses

In this section, briefly summarize the weak points (**W**s) of this work, as well as potential opportunities for improvements.

(**W1**) **Simple heuristic baseline.** In Section 2.3, the authors introduce a straightforward heuristic based on peak FLOPS to predict execution time, serving as a baseline. However, its simplicity prompts consideration of alternatives, such as simulation (e.g., [20, 23]), which may offer a stronger baseline for comparison. Notably, Figure 1 reveals that the model predictions consistently overestimate actual execution times by approximately 50–60%. As a potential enhancement, a straightforward adjustment would be reducing the estimation by around ∼ 33–37% for all heuristic predictions, which could narrow the margin between this baseline and the error rate achieved by Habitat (10.2%).

(**W2**) **Performance variation and accumulative errors.** While acknowledging the repetitive nature of ML training workloads, prior work has shown substantial performance variation (as large as 14% [6]) among iterations even using the same configuration. Although the paper has well motivated the repetitiveness, some illustration on variance (e.g., error bars) would be valuable. Moreover, it is noteworthy that the paper employs prediction error on single-iteration performance throughout the conducted experiments. However, it is essential to recognize that such prediction errors may accumulate over thousands of training steps, potentially leading to substantially larger errors than those observed in a single iteration. Consideration of cumulative errors over extended training durations would offer a more comprehensive understanding of the model's predictive performance.

[1] To maintain clarity, I use § to refer to the sections in this document. "Section" is exclusively used to refer to that of the reviewed papers.

(**W3**) **Optimizing model selection.** The authors employ MLPs to model kernel-varying operations, a task considered relatively straightforward, especially when analytical modeling is feasible for a specific a device. Given the specificity of the task and the relatively small amount of synthetically generated training data in this work, it is worth noting that MLPs, as universal function approximators [14], tend to require more data to achieve a certain level of performance compared to other models of similar size. To explore model efficiency, introducing simpler models such as decision trees/forests, along with corresponding ablation studies, could provide valuable insights.

(**W4**) **Exploring multitasking opportunities.** The authors train a separate MLP to model each kernel-varying operation, while these downstream tasks belong to the same category. To enhance the cost-effectiveness of the proposed method, it would be beneficial to investigate the feasibility of employing a unified model for some or all kernel-varying tasks. Exploring multitasking capabilities, where a single model addresses multiple related tasks.

(**W5**) **Scalability concerns in multi-device deployment.** The increasing size of ML models, exceeding the capacity of single GPUs like A100 for models larger than 7 billion parameters, necessitates the exploration of distributed training with multiple devices. Multi-device settings introduce complexities related to parallelization schemes and network bandwidth, influencing performance metrics such as tail latency and throughput, especially at scale [7, 13]. However, this work lacks experiments on such multi-device settings, leaving doubts regarding the transferability of findings to larger distributed setups (Section 6.1.1).

(**W6**) **Alternative hardware representations.** The authors opt for a straightforward representation of GPUs, relying on specifications such as memory capacity and peak FLOPS. Although simple, the information captured by these data points might not be sufficient for the prediction model. Alternative methods, such as encoding hardware directly into a latent space and utilizing embedding vectors as input to the prediction model [1], have been proposed. Similarly, hardware representation can be enriched by leveraging runtime statistics collected from executing a set of representative workloads.

(**W7**) **Integrating energy efficiency as a metric of interest.** Although this work could reduce both monetary and environmental costs (**S2**), it does not directly incorporate energy efficiency as an optimization objective. In the cloud, ML workloads are power bound [17, 25–28, 30], where energy consumption stands as a bottom line of cloud providers [4, 18]. Recognizing that performance does not translate directly to energy efficiency, it is pivotal to extend the scope of metrics to encompass energy considerations. Relying solely on performance metrics dismisses the significant impact of power consumption, and incorporating energy efficiency in the design space could drastically change the tradeoff continuum [29], which partially motivated **RQ1** (§4.1).

3 REVIEW OF ZEUS

In this section, I provide a brief review of Zeus [29].

3.1 Brief Summary

Target problem. The paper addresses the increasing resource- and energy-intensiveness of training DNNs. It recognizes that existing works tend to focus on optimizing DNN training for faster completion without considering the impact on energy efficiency. The primary problem is the tradeoff between performance optimization and energy consumption in DNN training.

Key contributions.

(1) The paper introduces the idea that common practices aimed at improving DNN training performance can lead to inefficient energy usage, highlighting the existence of a tradeoff between energy consumption and training time. The work claims to be the first to characterize this tradeoff, providing a fresh perspective on the optimization challenge.

(2) The authors develop and evaluate Zeus, an optimization framework designed to automatically discover optimal job- and GPU-level configurations for recurring DNN training jobs. It employs an online exploration-exploitation approach along with just-in-time energy profiling, reducing the need for costly offline measurements, and learning from and adapting to workload dynamics over time.

Main takeaways. Firstly, achieving performance optimality does not directly translate to achieving energy efficiency. In fact, they naturally form a Pareto frontier. Secondly, similar to Habitat, the Zeus framework provides a practical solution for the community; it automatically tunes job- and GPU-level configurations, and offers real-world applicability and environmental impacts.

3.2 Strength

(**s1**) **Emphasizing neglected tradeoff.** This work urgently highlights a commonly overlooked tradeoff between performance and energy efficiency. It emphasizes that improved performance does not necessarily equate to enhanced energy efficiency; instead, they shape a Pareto frontier, with each metric residing on one axis of the design space. Navigating this space effectively involves striking a delicate balance between the linear power-time relationship and the quadratic (or even cubic, accounting for frequency scaling) frequency-power relationship.

(**s2**) **Model- and device-agnostic online tuning.** A key challenge for existing online optimization tools lies in the necessity for users to define a range of instance-specific configurations. The proposed approach addresses this challenge by introducing automated online tuning, enhancing usability and thereby increasing the likelihood of adoption. Notably, this method eliminates the need for offline profiling or hardware modification. It exhibits the ability to generalize across different models and GPU types without inducing accuracy degradation.

(**s3**) **Strategic selection of tuning parameters.** Among the tunable parameters affecting the duration and/or energy consumption of a training workload, such as learning rate scheduling and DVFS policy, the authors strategically selected two specific parameters: batch size (impacting training duration) and GPU power cap (influencing energy consumption). This choice is smart for two primary reasons: (1) these two factors exhibit direct and relatively isolated effects on each of the two tradeoff dimensions, and (2) they have the ability to hide and influence various underlying tuning options

(e.g., adjustments in the power cap automatically trigger changes in DVFS).

(**s4**) **Applicable to single-node multi-GPU setup.**[2].

3.3 Weaknesses

(**w1**) **(Potentially) redundant characterization and experiments.** While acknowledging the importance of emphasizing the tradeoff between performance and energy (**s1**), it is worth noting that the non-linear relationship and associated implications have been extensively explored over the past decades. For instance, the outcomes illustrated in Figure 2 align with expectations: energy consumption scales almost quadratically with training duration (due to increased frequency). Similarly, the observed characteristics of the `(32,100W)` configuration, with the lowest energy cost due to the tradeoff between runtime and energy efficiency, are also anticipated. Consequently, these experiments could potentially be considered redundant, offering limited new insights to the community.

(**w2**) **No evaluation on the performance/energy overheads of the proposed method.** While the authors claim that Zeus has "negligible overhead," the evaluation results provided do not offer insights into the costs associated with integrating its workflow. Specifically, such runtime optimization tools can induce performance overheads [21, 22], as well as introduce additional energy costs [12].

(**w3**) **Questionable assumption on retraining.** The authors hinge on the assumption that the same model undergoes periodic retraining, utilizing these retraining jobs to explore diverse model and device configurations. However, in my experience, practical industry scenarios often involve companies updating models during retraining, introducing potential alterations to the optimization space. Thus, providing additional evidence or insights into the frequency and patterns of retraining could help substantiate this proposition.

(**w4**) **Unjustified use of overall average power.** The energy-to-accuracy calculation in Eq. 1 employs the overall average power over the training period, rather than using sampling. Aggregating over an extended duration may lead to inaccuracies, as demonstrated by the example: $2 \times 2 + 1 \times 3 \neq (2 + 1) \times (2 + 3)/2$. Such errors have the potential to accumulate significantly over time.

(**w5**) **Rough evaluation for multi-GPU settings.** The comparison with the prior work (Pollux) in Section 6 appears to be a bit hasty, lacking sufficient setup details or specific information on the sources of energy consumption. Critical details, such as the parallel scheme employed and the measured device power lanes in the evaluation, are notably absent.

4 POTENTIAL NEXT STEPS

In this section, I elaborate on two potential research questions (**RQ**s) inspired by the insights gained from prior work (§2, §3).

[2]However, the evaluation needs further refinement and specificity (**w5**)

4.1 Balancing Energy and Performance for Distributed ML Training

Motivation. Hardware specialization is rapidly advancing, leading to increasingly heterogeneous devices deployed in the cloud. This evolution is particularly pronounced when it comes to ML workloads in the cloud, where a variety of accelerators (e.g., GPUs and systolic arrays) are being employed. Unfortunately, these accelerators are power-bound [17, 25–28, 30], and their designs are heavily centered around cooling due to extreme power density [4, 18]. ML jobs often span prolonged durations, sometimes lasting for days or even months. For instance, the training of massive models like GPT-3 incurs an astonishing energy consumption, equivalent to the energy usage of an average American household over 120 years. Consequently, ML workloads are one of the biggest energy consumers in the cloud, which makes them expensive both for the users and cloud providers. Therefore, optimizing energy efficiency of these training workloads becomes imperative.

Gap. While Zeus proposes an online optimization method for improving ML training energy efficiency by co-optimizing the training configuration for a given GPU device and a model, such a device-specific approach is inadequate. The existence of *a diverse range of accelerator types*, each manifesting distinct performance-energy characteristics, can result in significant variations in energy efficiency [15, 18, 28]. Furthermore, *energy efficiency is also task-specific*. For instance, even within the same model family (e.g., recommendation models [11, 15]), the energy-optimal configuration varies among tasks.

Although Habitat excels at predicting the performance of a model's training on a designated device, it lacks consideration for energy efficiency as a primary optimization objective (**W6**). Moreover, its applicability does not naturally extend to a distributed setting (**W7**, **W5**).

Hence, I propose to develop an extension to Habitat that facilitates users in automatically choosing the most energy-efficient combination of *device type* and *training parallel scheme*, tailored to a specific model and downstream task. This objective forms the basis for **RQ1**.

RQ1: How can we develop a cost-effective, automated method that helps users select the most energy-efficient ML training configuration tailored to a specific model and downstream task, while meeting user-defined performance objectives such time-to-accuracy (TTA)?

Challenges. Similar to estimating performance in Habitat, characterizing the energy efficiency of ML training jobs is equally resource-intensive in terms of both time and cost. Consequently, manual measurement for every training setup is impractical. Moreover, aside from the inherent stochasticity in ML workloads (**W2**), the mapping from performance counters and relevant statistics to accelerator energy consumption is inherently non-linear. Most accelerators (e.g., GPU [19, 23], TPU [18], and FPGAs [8]) utilize dynamic frequency scaling to prevent overheating, introducing additional complexities to power modeling and estimation [19].

Last but not least, estimating energy consumption in a distributed setting also poses challenges. The communication and data transfer between devices can contribute to non-negligible energy costs. Tracing these consumption sources, however, is a non-trivial task [2,

12, 33]. Another potential source of difficulty in a distributed setting is the increased uncertainty and variation, given that stragglers and other related hardware issues could skew the results [13].

Research Approach to RQ1. In comparison to Habitat (§2), **RQ1** introduces an additional optimization objective, energy efficiency, and extends the scope to include device type and parallel training scheme as automatic optimization targets, which were not explicitly considered in Zeus (§3). However, several methodologies from both Habitat and Zeus can be leveraged to address this question.

Firstly, as well motivated in Habitat, recognizing the prevalence and accessibility of GPUs, especially in the context of ML workloads, the study should initially focus on GPUs as a starting point. Secondly, the assumption in Habitat about users having a local device remains valuable, providing a basis for further estimations and predictions. However, this assumption's utility may be diminished considering the incorporation of parallel training schemes, since assuming users have a cluster of devices might be a tall order. Thirdly, drawing inspiration from **S3**, I plan to use a hybrid approach combining validated analytical models (e.g., [19, 23]) and predictive methods. This approach is necessary when dealing with the inherent difficulty or impossibility of tracing certain consumption sources [2, 33]. To train the predictive model, I plan to employ a strategy similar to **S4**, starting with generating synthetic training data by varying setup and input configurations. However, alternative hardware representations could be explored, distinct from those used in Habitat (**W6**).

Lastly, to navigate the expanded design space, automated tradeoff exploration akin to **s2** should be incorporated. Given the enlarged optimization landscape due to added dimensions, and the necessity of deciding certain configurations in advance (e.g., device type and parallel schemes), a combination of online and offline optimization appears to be more practical.

4.2 Using ML Training Workloads for Datacenter Power Objectives via Energy-Aware Scheduling

Motivation. Consuming almost 3% of global electricity, datacenters are not only one of the biggest energy consumers but also playing an increasingly critical role in balancing the supply and demand of the power grid thanks to their huge capacity and load flexibility. Specifically, by time- and space-shifting workloads, datacenters can absorb excessive renewable energy by increasing dynamic load and reduce their consumption when the carbon intensity is high (e.g., due to undesirable weather conditions). This datacenter power objective is known as demand response [24]. In fact, to help achieve net-zero carbon emission goals by 2050, datacenters have to at least double their participation in demand response programs [16].

Gap. Unfortunately, despite ML training being one of the most energy-intensive workloads in the cloud, datacenters predominantly leverage batch jobs and scientific workloads for demand response [31, 32]. This gap is partially attributed to concerns about potential degradation in model quality, the diverse types of accelerators, and the complexities associated with migrating ML workloads.

Nevertheless, I believe that recent advancements make leveraging ML training workloads for demand response feasible.[3] For instance, Zeus provides a promising solution for automatic tuning of energy-efficient ML training without compromising model performance (**s2**). Similarly, Habitat (**S1**) and recent works on GPU energy modeling (e.g., [9, 19, 23]) demonstrate the predictability of GPU performance and power characteristics, critical for planned demand response. Moreover, substantial progress has been made in dynamic scheduling for model training at runtime [3]. For example, during the 51-day training of PaLM [5], model shards were frequently relocated at runtime for reasons like resource preemption and failure recovery. To address these gaps, I pose **RQ2**.

RQ2: How can we leverage ML training jobs for datacenter power objectives such as demand response and hotspot mitigation?

Challenges. Addressing **RQ2** entails extending the solution proposed for **RQ1** to a cluster scale and managing orchestration across multiple ML training jobs. The resulting framework must not only configure the model (e.g., batch size) and the accelerators (e.g., power cap) for each training job but also consider the cluster-wide implications of the optimization. This optimization involves the inherent energy-performance tradeoff (**s1**) – given a fixed cluster power budget, reducing the power cap naturally leads to energy saving but also performance degradation (e.g., a longer TTA), potentially introducing concerns related to fairness and adherence to users' SLAs. Moreover, since the energy-optimal setting is both model- and task-specific, building a dedicated power model and deploying a monitoring agent for each workload may be necessary. Lastly, evaluating the resulting framework also presents challenges, potentially requiring collecting production traces at a reasonable scale.

Research approach to RQ2. I plan to start with applying existing demand response methods designed for traditional batch jobs and scientific workloads to ML training jobs, conducting a comprehensive literature review and evaluating available artifacts. As an initial goal, I will attempt to schedule and configure training jobs to consume the same amount of energy. This naive policy should only consider homogeneous hardware (e.g., using the same type of GPUs) without runtime relocation of model partitions. Such experiments could provide me with insights into the impacts of this naive policy on training performance (e.g., TTA) and simplify validation and evaluation through trace-driven simulation.

Apparently, equalizing power budget over all training jobs is unfair, since different models and downstream tasks would have a different performance-energy Pareto frontier as described in previous sections. Consequently, the naive approach will likely to result in varying performance degradations across training jobs. Therefore, the subsequent step can aim to evenly distribute the performance degradation across all training jobs. One potential method is to employ runtime-based optimization (similar to Zeus and Habitat) to adjust the model and device configuration when performance degrades beyond the modeled behavior and *attempt to recover performance* by, for example, rapidly increasing its power budget in later stages. Another possible approach involves utilizing

[3] Although inference workloads typically amount to larger amounts of energy consumption [28], they are more sporadic and much less intense. Training jobs are more similar to traditional scientific workloads and, therefore, could potentially allow me to apply some existing methods designed for demand response programs.

detailed online/offline power profiles and leveraging complementary training jobs. For instance, it may be possible to reduce the power budget of a memory- or I/O-bound training job by 20% and simultaneously increase the power cap of a compute-bound job by about 20%.

Dynamic model relocation at runtime on heterogeneous hardware should only be considered if the aforementioned objectives are achieved and validated. Other potential future directions could include over-provisioning power budgets to further maximize utilization and colocating workloads on a single GPU node based on power profiles (e.g., distributing high-power jobs across the cluster to create balanced power mixing). However, these are relatively distant goals whose solutions depend on prior experiments.

REFERENCES

[1] Yash Akhauri and Mohamed S Abdelfattah. 2023. Multi-Predict: Few Shot Predictors For Efficient Neural Architecture Search. *arXiv preprint arXiv:2306.02459* (2023).

[2] Vaastav Anand, Zhiqiang Xie, Matheus Stolet, Roberta De Viti, Thomas Davidson, Reyhaneh Karimipour, Safya Alzayat, and Jonathan Mace. 2023. The odd one out: Energy is not like other metrics. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 71–77.

[3] Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Daniel Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, et al. 2022. Pathways: Asynchronous distributed dataflow for ml. *Proceedings of Machine Learning and Systems* 4 (2022), 430–449.

[4] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. 2019. *The datacenter as a computer: Designing warehouse-scale machines*. Springer Nature.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[6] Cody Coleman, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. 2019. Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark. *ACM SIGOPS Operating Systems Review* 53, 1 (2019), 14–25.

[7] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

[8] André DeHon. 2014. Wordwidth, instructions, looping, and virtualization: the role of sharing in absolute energy minimization. In *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*. 189–198.

[9] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Parteek Sharma, Fan Chen, and Lei Jiang. 2023. LLMCarbon: Modeling the end-to-end Carbon Footprint of Large Language Models. *arXiv preprint arXiv:2309.14393* (2023).

[10] X Yu Geoffrey, Yubo Gao, Pavel Golikov, and Gennady Pekhimenko. 2021. Habitat: A {Runtime-Based} computational performance predictor for deep neural network training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 503–521.

[11] Udit Gupta, Samuel Hsia, Jeff Zhang, Mark Wilkening, Javin Pombra, Hsien-Hsin Sean Lee, Gu-Yeon Wei, Carole-Jean Wu, and David Brooks. 2021. RecPipe: Co-designing models and hardware to jointly optimize recommendation quality and performance. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 870–884.

[12] Hongyu Hè, Michal Friedman, and Theodoros Rekatsinas. 2023. EnergAt: Fine-Grained Energy Attribution for Multi-Tenancy. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. 1–8.

[13] Peter H Hochschild, Paul Turner, Jeffrey C Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E Culler, and Amin Vahdat. 2021. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. 9–16.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[15] Samuel Hsia, Udit Gupta, Bilge Acun, Newsha Ardalani, Pan Zhong, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2023. MP-Rec: Hardware-Software Co-design to Enable Multi-path Recommendation. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 449–465.

[16] IEA. 2023. Demand Response. https://www.iea.org/energy-system/energy-efficiency-and-demand/demand-response.

[17] Gamze Islamoglu, Moritz Scherer, Gianna Paulin, Tim Fischer, Victor JB Jung, Angelo Garofalo, and Luca Benini. 2023. Ita: An energy-efficient attention and softmax accelerator for quantized transformers. In *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.

[18] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–14.

[19] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G Rogers, Tor M Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A power modeling framework for modern GPUs. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 738–753.

[20] Mahmoud Khairy, Zhesheng Shen, Tor M Aamodt, and Timothy G Rogers. 2020. Accel-Sim: An extensible simulation framework for validated GPU modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 473–486.

[21] Tanvir Ahmed Khan, Akshitha Sriraman, Joseph Devietti, Gilles Pokam, Heiner Litz, and Baris Kasikci. 2020. I-spy: Context-driven conditional instruction prefetching with coalescing. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 146–159.

[22] Tanvir Ahmed Khan, Muhammed Ugur, Krishnendra Nathella, Dam Sunwoo, Heiner Litz, Daniel A Jiménez, and Baris Kasikci. 2022. Whisper: Profile-guided branch misprediction elimination for data center applications. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 19–34.

[23] Jonathan Lew, Deval A Shah, Suchita Pati, Shaylin Cattell, Mengchi Zhang, Amruth Sandhupatla, Christopher Ng, Negar Goli, Matthew D Sinclair, Timothy G Rogers, et al. 2019. Analyzing machine learning workloads using a detailed GPU simulator. In *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 151–152.

[24] Zhenhua Liu, Iris Liu, Steven Low, and Adam Wierman. 2014. Pricing data center demand response. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014), 111–123.

[25] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350* (2021).

[26] Matteo Risso, Alessio Burrello, Giuseppe Maria Sarda, Luca Benini, Enrico Macii, Massimo Poncino, Marian Verhelst, and Daniele Jahier Pagliari. 2023. Precision-aware Latency and Energy Balancing on Multi-Accelerator Platforms for DNN Inference. *arXiv preprint arXiv:2306.05060* (2023).

[27] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[28] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

[29] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and Optimizing {GPU} Energy Consumption of {DNN} Training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 119–139.

[30] Luca Zanatta, Alfio Di Mauro, Francesco Barchi, Andrea Bartolini, Luca Benini, and Andrea Acquaviva. 2023. Directly-trained Spiking Neural Networks for Deep Reinforcement Learning: Energy efficient implementation of event-based obstacle avoidance on a neuromorphic accelerator. *Neurocomputing* 562 (2023), 126885.

[31] Yijia Zhang, Daniel C Wilson, Ioannis Ch Paschalidis, and Ayse K Coskun. 2021. A data center demand response policy for real-world workload scenarios in HPC. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 282–287.

[32] Yijia Zhang, Daniel Curtis Wilson, Ioannis Ch Paschalidis, and Ayse K Coskun. 2021. HPC data center participation in demand response: an adaptive policy with QoS assurance. *IEEE transactions on sustainable computing* 7, 1 (2021), 157–171.

[33] Noa Zilberman, Eve M Schooler, Uri Cummings, Rajit Manohar, Dawn Nafus, Robert Soulé, and Rick Taylor. 2023. Toward carbon-aware networking. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 15–20.