Efficient GPU Power Management through Advanced Framework Utilizing Optimization Algorithms

Ramesha Rehman, Mashood Ul Haq Chishti,
Hamza Yamin

Abstract

The rapid rise in power usage by GPUs due to advances in machine and deep learning has led to an increase in power consumption of GPUs in Deep Learning workloads. To address this issue, a novel research project focuses on integrating Particle Swarm Optimization into a model training optimization framework to effectively reduce GPU power consumption during machine learning and deep learning training workloads. By utilizing the Particle Swarm Optimization (PSO)[1] algorithm within the proposed framework, we show the effectiveness of PSO in creating a more efficient power management strategy while also maintaining the performance. Upon evaluation of the proposed framework, it shows a reduction of 15.8% to 75.8% in power consumption across multiple workloads, with little to no performance loss.

Keywords: GPU, power reduction, machine learning, Particle Swarm Optimization.

MSC2020: 68-XX, 68Txx, 68T07.

1 Introduction

Over recent years, Graphics Processing Units (GPUs) have emerged as pivotal components across various domains such as gaming, computational graphics, machine learning, and scientific simulations [2]. The catalyst behind their prominence is their parallel processing capabilities, revolutionizing high-performance computing by enabling more

doi:10.56415/csjm.v32.08

efficient and rapid calculations. This progression in GPU technology has ushered in an era of heightened computing power, enabling the concurrent execution of numerous tasks. This parallelism contrasts with the sequential nature of central processing units (CPUs), conferring a significant advantage in handling computationally intensive applications. This multifaceted computation ability enhances speed and efficacy across diverse fields. However, the remarkable advancement in GPU computational capacity is accompanied by a significant escalation in energy consumption. The intrinsic parallel processing potential of GPUs, due to their multitude of processing cores, necessitates substantial electricity usage. The escalating power consumption of GPUs has raised pertinent concerns about environmental sustainability and energy conservation.

The elevated energy consumption of GPUs can be attributed to their specific design and architecture, which are optimized for parallel processing. Unlike CPUs, GPUs are engineered to execute tasks concurrently, resulting in a substantial core count to handle multiple calculations simultaneously. While this architectural choice boosts performance, it concurrently demands a greater power supply to accommodate the augmented workload. In response to these concerns, GPU manufacturers are actively exploring diverse avenues to mitigate power consumption and enhance energy efficiency [13]. This pursuit includes the development of more power-efficient architectures that optimize the balance between computational prowess and power usage. Additionally, software improvements play a pivotal role in enhancing energy efficiency. Software optimizations enable the effective distribution of computing tasks across available cores, thereby maximizing parallel processing capabilities and minimizing superfluous power consumption.

Clock gating [17] is a technique that selectively blocks the clock signal to individual GPU component when the component in question is not executing tasks. Power consumption can be substantially reduced by switching off the clock in dormant or seldom utilized components. Energy savings can be obtained by reducing wasted power consumption in GPU units that are not actively engaged in computations.

Another hardware-based technique called dynamic voltage and fre-

quency scaling (DVFS) [16] adjusts the GPU's operating voltage and frequency in accordance with workload demands. Energy usage can be optimized by dynamically adjusting these parameters to the ideal efficiency level [5]. In periods of reduced processing activity, DVFS permits the GPU to operate at lower voltages and frequencies, thus reducing power consumption while maintaining performance. While these hardware-driven optimization methods have shown some degree of effectiveness in lowering GPU power usage, there is still room for improvement. Researchers keep looking at new concepts and ways to improve energy efficiency.

This article delves into the potential of leveraging machine learning (ML) models to curtail GPU power consumption, concentrating on optimal batch size and power limit configurations [8], an aspect often overlooked in the pursuit of model performance enhancement. Notably, ML's transformative impact on energy efficiency has surpassed its immediate domain, with far-reaching consequences. The astronomical energy consumption exhibited by large-scale models like GPT-3, which consumes as much as 1,287 megawatt-hours (MWh) of electricity, exemplifies this critical issue. This consumption equates to the energy utilization of an average U.S. home over a span of 120 years. While commendable strides have been made to reduce operational power footprints, the unceasing growth of artificial intelligence processing requirements poses potential energy challenges.

Given the evolving landscape of GPUs in machine learning and deep neural networks, it is imperative to address the research and innovation gap concerning energy efficiency [14] [12]. This research project aims to fill this void by comprehensively investigating the intricate interplay between energy consumption and processing capacity in the context of GPUs. The endeavor seeks to redefine the role of GPUs in energy-efficient AI and inspire researchers and industry professionals alike. By cultivating an in-depth understanding of energy-efficient GPU utilization, we aspire to foster a collaborative endeavor that amalgamates technological innovation with sustainable resource management in the dynamic realm of machine learning and deep neural networks.

In response to the existing gap in model training optimization, our research introduces a framework [11] aimed at reducing the power con-

sumption of a workload [4] [9] [6]. This framework achieves optimization by adjusting the batch size and establishing an optimal power limit for a given workload [7]. To enhance batch size optimization, we employ a Multi Armed Bandit with GS-MOPSO strategy, allowing exploration of the search space to determine the most suitable batch size for the workload [10]. Simultaneously, power optimization is addressed through the utilization of a Just-in-Time profiler. This profiler leverages pre-calibrated configurations if optimization has been previously conducted for a similar workload. For novel workloads, the profiler measures throughput and average power consumption across various power limits, ultimately selecting the configuration that maximizes throughput while minimizing average power consumption. [15]

2 Methodology

2.1 Experimental Methodology

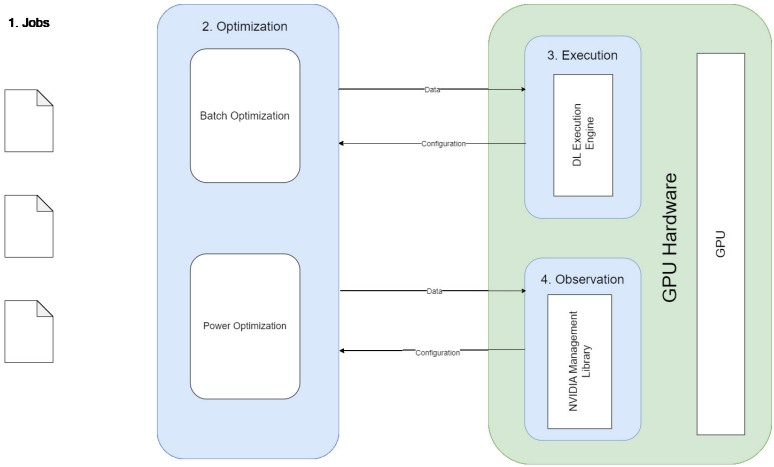


Figure 1. Proposed Methodology architecture

The proposed methodology, illustrated in Figure 1, aims to enhance efficiency through a multi-stage process. Tasks or jobs are channeled into the optimization framework (1), which unfolds as follows:

Optimization Stage: The initial phase employs Particle Swarm Optimization (PSO) to ascertain the optimal configuration for batch size and power limits (2). Subsequently, the training procedure commences employing the established configuration (3).

Monitoring and Feedback: Throughout the training process, the optimization framework gathers continuous statistics and information as feedback from the model's progression (4).

Our Proposed Framework works on step 2 (Optimization) of the proposed methodology, where it interacts with the GPU hardware to set the optimal configuration for the incoming jobs.

Adaptive Learning: Leveraging the collected feedback, the framework engages in adaptive learning, iteratively refining its configuration settings. This iterative loop persists until the predefined performance measure set by the user is attained or until the model's performance does not become more efficient within a specified time duration.

The iterative learning process empowers the framework to dynamically adapt its configuration based on feedback data, maintaining a trajectory towards the defined performance target or until saturation in model improvement is detected. Notably, the framework bifurcates the challenges associated with batch size and power limits, two parameters substantially influencing GPU performance and energy consumption. This strategic separation enables independent determination of the optimal power limit for any given batch size. Additionally, due to this decoupling, our exploration space is effectively confined to diverse batch sizes that harmonize with the optimal power limit. This focused exploration approach expedites the search for the optimum batch size, obviating the need to exhaustively assess every conceivable configuration.

2.2 Theoretical methodology

Our framework places a strong emphasis on the cost metric, which is a crucial component of our approach, hence we propose the following metric

$$Cost = \alpha.E_{tar}(s, w) + (1 - \alpha).P_{max}.T_{tar}(s, w). \quad (1)$$

Here α is the significance factor – if it is 0, then the framework

optimizes for time efficiency, and if it is 1, then it optimizes for energy efficiency; P_{max} is the maximum power allowed by the GPU; and E_{tar} and T_{tar} are energy consumed and time taken to reach the target metric with configuration batch size s and power limit w.

In conclusion, our cost metric is a flexible tool that enables you to customise the optimisation method for your framework in accordance with your unique goals. You may efficiently strike a balance between time and energy efficiency by modifying the importance factor, ensuring that your framework fits your intended objectives and limits.

2.2.1 Choosing Ideal Cost

Expanding the equation (1), we get

$$Cost = (\alpha.P_{avg}(s, w) + (1 - \alpha).P_{max}).T_{tar}(s, w). \quad (2)$$

In order to get the best cost, one must navigate a huge search space that is bounded by different batch sizes (s) and power restrictions (w). Additionally, because of the inherent variability in neural network training and the various hardware configurations of GPUs, it can be difficult to estimate both the average power consumption ($P_{avg}(s, w)$) and the time needed to obtain the goal metric ($T_{tar}(s, w)$).
We've put in place a dual approach to deal with these complications and uncertainties:

1. **Just-in-Time Profiler:** A Just-in-Time profiler has been incorporated into our system. During the execution of neural network training, this profiler is essential for dynamically measuring and monitoring the performance traits of various (s, w) configurations. We can make better judgments since it offers real-time information about the time and power needs [20], [19].

2. **Multi-Armed Bandit with GS-MOPSO:** We've used a Multi-Armed Bandit approach along with GS-MOPSO (Multi-Objective Particle Swarm Optimisation) to effectively explore the enormous search area and adapt to the stochastic nature of neural network training. Through careful consideration of the trade-offs between energy and time efficiency, this combination enables us

to deploy resources in a sensible manner. It continually modifies the (s, w) settings depending on performance data from the past, assisting us in more efficiently converging towards the ideal solution.

Essentially, our strategy makes use of real-time profiling and sophisticated optimisation techniques to address the problems brought on by the expansive and unpredictable (s, w) search space, as well as the inherent variability in GPU hardware and neural network training. This allows us to ultimately find the best cost-effective solutions.

2.2.2 Optimising power

Utilising the capabilities of our Just-in-Time profiler to increase power efficiency is fundamental to our optimisation approach. Within our system, when a job is started, the profiler activates and adheres to the following protocol:

1. **Batch Size Check:** The profiler first determines if the job's batch size has been calculated and optimised. It uses the pre-calibrated and optimised power limit for that specific batch size if optimisation has already been done for this batch size. With this strategy, we can rapidly and effectively distribute the right power resources for batches of known sizes.

2. **First Epoch Profiling:** When the batch size is used for the first time, our profiler intervenes to collect crucial information. During the first epoch of the work, it measures throughput and average power consumption (Pavg) for various power restrictions. We can create a baseline understanding of how various power restrictions effect performance for this particular batch size thanks to this thorough profiling.

2.2.3 Optimising batch size

We use Multi Armed Bandit with GS-MOPSO as shown in the algorithms below, to improve our batch size optimisation process. Together, these algorithms enable us to effectively calculate the optimal

batch size based on the provided power restriction and then to carry out the training task using the identified power limit (w) and batch size (s).

Algorithm 1.

Input: *Batch Sizes B, belief posterior* $mean_b$ *and* std_b^2
Output: *Batch size to run* b^*
Function: $Predict(B, mean_b, std_b^2)$*:*
Initialize random population P
Initialize Personal Best Set S_{pb} *and External set* S_e
For a=1:B do
    $S_{pb}\{a\}$=*P(a)*
S_e=*P*
For i=1:B do
    Assign rank to each particle in P_i *according to fast non-dominated sort*
    For k=1:B
        P_{best} = *first particle in sorted* $S_{pb}\{k\}$
        N_{best}= *particle closest to* k_{th} *particle in* S_e
        Update $P_i(k)$ *to* $P_{i+1}(k)$
        if $rand > Rank_i/maxRank$
            if $rand > i/B$
                Sample=$[P_{best}, N_{best}]$
                $P_{i+1}(k)$=$N(mean_b, std_b^2)$
            else
                Divide $[P, S_{pb}, S_e]$ *into N clusters using K-means*
                Identify the cluster, to which particle $P_i(k)$ *belongs and assign it to Sample*
                P_{i+1}=$N(mean_b, std_b^2)$
Update S_e
$b^* \longleftarrow argmin S_e$

Algorithm 2.

Input: *Batch Size b and Total cost* C_t,
Previous cost C_p
belief posterior $mean_b$ *and* std_b^2
Output: *updated belief posterior* $mean_b$ *and* std_b^2
Function: $Observe(b, C_t, C_p, mean_b, std_b^2)$*:*
$C_p \longleftarrow C_p \bigcup \{C_t\}$
$std^2 \longleftarrow Variance(C_t)$
$std_b^2 \longleftarrow (\frac{1}{std_0^2} + \frac{|C_p|}{std^2})$
$mean_b \longleftarrow std_b^2(\frac{mean_0}{std_0^2} + \frac{\sum C_p}{std^2})$

Algorithm 3.

Input: *Batch Size b*
belief posterior $mean_0$ *and* std_0^2
while $t<T$ *do*
$b^* \longleftarrow Predict(B, mean_b, std_b^2 \forall b \in B)$
Run job with b^* *and add to cost C*
$mean_b$,$std_b^2 \longleftarrow Observe(b, C, mean_0, std_0^2)$
$t \longleftarrow t + 1$

Our novel optimisation framework stands at the forefront in enhancing computational efficiency in GPU-centric deep learning tasks. It successfully integrates three essential algorithms to dynamically modify batch sizes and minimize power usage. The sophisticated Algorithm 1 (Predict) at the heart of this technique incorporates the Multi Armed Bandit with GS-MOPSO strategy. Using belief posterior parameters like meanb and std2b, this strategy systematically explores and determines the ideal batch size. By proactively adjusting to the changing subtleties of the workload, it makes sure that the batch sizes have been carefully selected to match the deep learning model's exact specifications.

After the first prediction, Algorithm 2 (Observe) plays an essential role in coordinating an adaptive learning process at the end of every task run. This technique improves the model's understanding of the computation environment by iteratively updating the belief posterior

with observed batch sizes, total cost (Ct), and the cumulative history of past costs (Cp). This continuous learning loop increases our optimization framework's versatility so that it can dynamically adjust to changing computing needs. Lastly, Algorithm 3 handles the whole iterative procedure, including belief posterior updates, task execution, and prediction in a smooth manner across a predefined number of iterations, T. The integrated approach's comprehensive nature highlights its effectiveness in managing power consumption and computing efficiency while also adeptly navigating the dynamic nature of deep learning workloads. With the help of these methods, our system shows itself to be a comprehensive and flexible approach, well suited to maximize GPU-driven deep learning's performance in the always changing field of computational difficulties.

We can speed up the batch size selection process and make sure that it perfectly matches the determined power limit(w) by combining these approaches. With this strategy, we may dynamically adjust and optimise the batch size(s) in response to shifting circumstances and limitations. As a consequence, we are able to control and carry out the training workloads with the best possible balancing of batch size and power limit.

3 Experiment/Results

In our evaluation, we considered prominent Deep Neural Network (DNN) models such as DeepSpeech2, ResNet-50, and NeuMF and others as shown in Table 1. Our investigation revolved around comparing training time and energy utilization across three methodologies: the default strategy, grid search exploration, and our novel approach on system specification as shown in Table 2. The outcomes of our research unveiled significant insights, outlined as follows:

3.1 Energy Consumption Reduction:

Figure 2 depicts the graphical representation of performance for the most recent five iterations of our approach, grid search, and the default baseline. Our methodology showcases a remarkable reduction in energy consumption, ranging from 15.3% to an impressive 75.8%.

Table 1. Models and datasets used in our evaluation of the respective configurations

Task	Dataset	Model	Optimizer	Default batch size	Target Metric
Speech Recognition	LibriSpeech	DeepSpeech2	Adam	176	WER: 55
Question answers	Sentiment140	BERT(QA)	Adam	44	F1:88.0
Sentiment Analysis	ImageNet	BERT(SA)	Adam	122	Accuracy:86%
Image Classification	CIFAR-100	RESNET-50, ShuffleNet-v2	Adadelta	232, 1000	Accuracy: 68%, 65%
Recommendation	MovieLens-1M	NeuMF	Adam	1000	NDCG: 0.52

This is further supported with Figure 5 and Table 3. This substantial decrease in energy usage is achieved while imposing minimal impact on training time.

3.2 Balanced Time-Consumption Trade-offs:

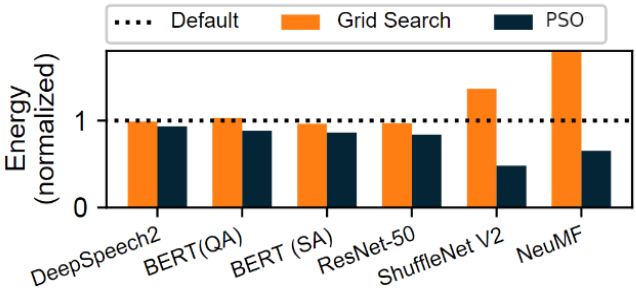
Within the context of training time, our strategy demonstrates a balanced trade-off with energy efficiency. Figure 2b elucidates this aspect by illustrating the duration of the last five rounds for both our methodology and grid search, compared to the default baseline. Notably, our methodology achieves a training time reduction of up to 60.1% for specific workloads, alongside a modest 12.8% increase for a distinct group of workloads. This finding underscores the delicate equilibrium between optimizing time and enhancing energy efficiency.

Table 2. GPU and System Specifications used in evaluation of our framework.

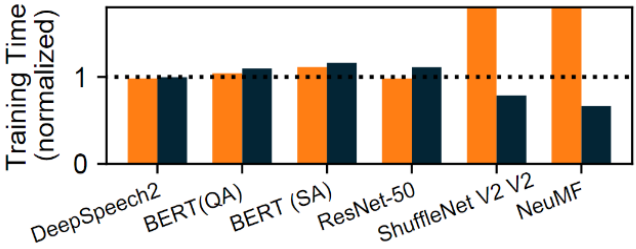
GPU Specifications	System Specification
Model	A40
CPU	AMD EPYC 7513
VRAM	48GB
RAM	512GB DDR4
Architecture	Ampere
Disk	960 GB NVMe SSD
Model	V100
CPU	AMD EPYC 7542
VRAM	32GB
RAM	512GB DDR4
Architecture	Volta
Disk	2TB HDD
Model	RTX 6000
CPU	Xeon Gold 6126
VRAM	24GB
RAM	192GB DDR4
Architecture	Turing
Disk	256GB SSD
Model	P100
CPU	Xeon E5-2670 v3
VRAM	16GB
RAM	128 GB DDR4
Architecture	Pascal
Disk	1TB HDD

Table 3. Improvements of different methodologies employed on DL training workloads w.r.t. Default NVIDIA strategy [18]

Method Employed	Least	Best
DVFS [16]	8.7%	23.1%
Clock Gating [17]	11.2%	60%
GPOEO [4]	8%	29.5%
Our Framework with PSO	15.3%	75.8%



(a) *Energy Consumption*

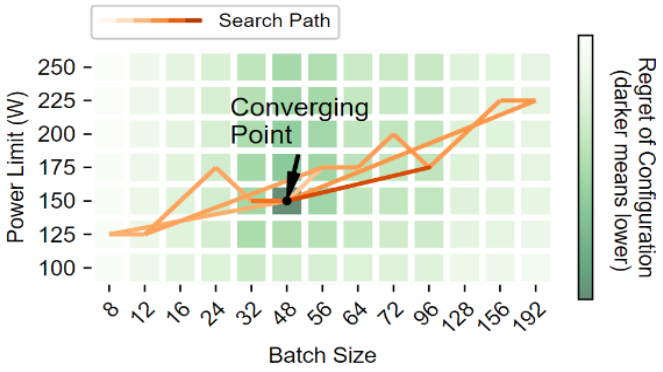


(b) *Training Time*

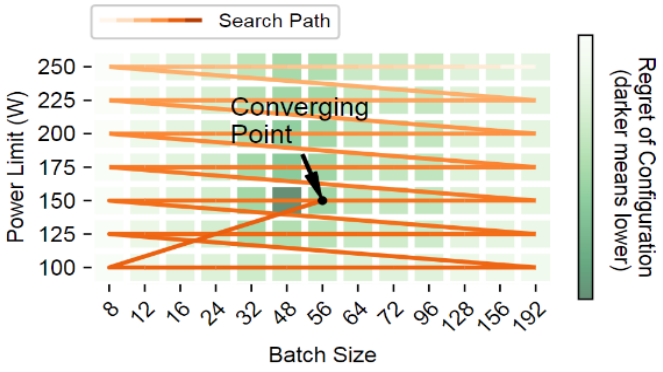
Figure 2. Energy consumption and training time of last 5 iterations of Grid Search and Our framework with PSO w.r.t. baseline NVIDIA strategy

3.3 Progressive Regret and Resource Efficiency:

While our methodology and grid search exhibit comparable performance, our approach showcases superior resource efficiency for convergence. As illustrated in Figure 3, the increasing regret trajectories for DeepSpeech2 and ResNet-50 models underscore our methodology's ability to attain similar outcomes with significantly fewer resources compared to grid search, this is also supported by Figure 6.



(a) PSO



(b) Grid Search

Figure 3. Progressive Regret of our framework with both Grid Search and PSO

3.4 GPU Model-Independent Energy Savings:

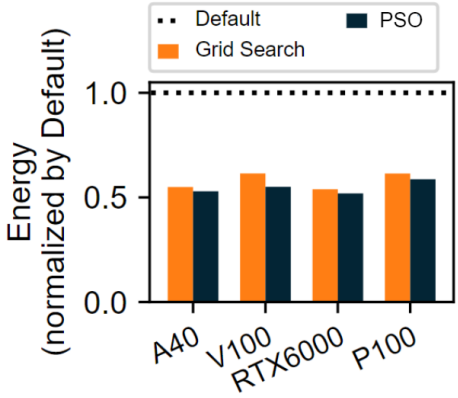


Figure 4. Energy consumption for DL workloads over multiple GPUs using our framework with Grid Search and PSO w.r.t default NVIDIA strategy

An essential discovery from our study is the independence of our methodology's effectiveness from GPU models as shown in Figure 4. Across four NVIDIA GPU generations, our approach consistently achieves substantial energy consumption reductions. This energy-saving capability exceeds 50% compared to the default baseline across multiple GPU models.

In summation, our comprehensive analysis demonstrates the efficacy of our approach in mitigating energy consumption without imposing considerable time penalties. This balanced strategy emerges as a viable avenue to enhance the training of Machine Learning and Deep Neural Network models across diverse workloads and GPU architectures.

4 Conclusion

In summation, our approach has showcased its efficacy through notable reductions in energy consumption, all while maintaining the unimpaired performance of GPUs. This achievement in diminishing GPU energy usage is attributed to the framework's adeptness in delicately balancing energy preservation and optimization of training time.

The promising outcomes of our technological endeavor underscore its potential to instigate a paradigm shift in the domain of energy-efficient Deep Neural Network (DNN) training. The approach's precision in quantifying the intricate interplay between training time and energy consumption has yielded substantial advancements in enhancing GPU energy efficiency.

Anticipating the future, we envision the widespread adoption of this methodology across diverse sectors, transcending its current scope. Innovations such as TinyML [3] open new vistas, wherein the implementation of our strategy in compact devices like mobile devices and embedded systems holds immense promise. Foreseeing a consequential enhancement in the efficiency of such devices through our methodology, we expect this trend to pave the way toward a technological landscape characterized by sustainability and heightened energy consciousness.

References

[1] A. A. Nagra, F. Han, Q. -H. Ling, M. Abubaker, F. Ahmad, S. Mehta, and A. T. Apasiba, "Hybrid self-inertia weight adaptive particle swarm optimization with local search using C4.5 decision tree classifier for feature selection problems," *Connection Science*, vol. 32, no. 1, pp. 16–36, DOI: 10.1080/09540091.2019.1609419.

[2] A. A. Nagra, I. Mubarik, M. M. Asif, K. Masood, M. A. A. Ghamdi, and S. H Almotiri, "Hybrid GA-SVM Approach for Post-operative Life Expectancy Prediction in Lung Cancer Patients," *Appl. Sci*, vol. 12, Article No. 10927, 2022.

[3] M. Shafique, A. Marchisio, R. V. Wicaksana Putra, and M. A. Hanif, "Towards Energy-Efficient and Secure Edge AI: A Cross-Layer Framework ICCAD Special Session Paper," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9, DOI: 10.1109/ICCAD51958.2021.9643539.

[4] F. Wang, W. Zhang, S. Lai, M. Hao, and Z. Wang, "Dynamic GPU Energy Optimization for Machine Learning Training Workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2943–2954, 1 Nov. 2022, DOI: 10.1109/TPDS.2021.3137867.

[5] S. Ilager, R. Muralidhar, K. Rammohanrao, and R. Buyya, "A Data-Driven Frequency Scaling Approach for Deadline-aware Energy Efficient Scheduling on Graphics Processing Units (GPUs)," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud, and Internet Computing (CCGRID)*, 2020, pp. 579–588, DOI: 10.1109/CCGrid49817.2020.00-35.

[6] M. Endrei, C. Jin, M. N. Dinh, D. Abramson, H. Poxon, L. DeRose, and B. R. de Supinski, "Statistical and machine learning models for optimizing energy in parallel applications," *The International Journal of High-Performance Computing Applications*, vol. 33, no. 6, pp. 1079–-1097, 2019.

[7] S. Ramesh, S. Perarnau, S. Bhalachandra, A. D. Malony, P. Beckman, *et al.*, "Understanding the impact of dynamic power capping on application progress", in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE*, 2019, pp. 793–-804.

[8] Y. Wang, W. Zhang, M. Hao, and Z. Wang, "Online power management for multi-cores: A reinforcement learning based approach," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[9] K. Fan, B. Cosenza, and B. Juurlink, "Predictable GPUs frequency scaling for energy and performance," in *Proceedings of the 48th International Conference on Parallel Processing*, 2019, pp. 1–-10.

[10] Y. Wen, Z. Wang, and M. F. O'Boyle, "Smart multi-task scheduling for OpenCL programs on CPU/GPU heterogeneous platforms," in *2014 21st International conference on high performance computing (HiPC). IEEE*, 2014, pp. 1–-10.

[11] V. Kandiah *et al.* "AccelWattch: A power modeling framework for modern gpus," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, DOI: 10.1145/3466752.3480063.

[12] Keskar, N. Shirish, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," arXiv preprint arXiv:1609.04836, 2016.

[13] M. Hodak, M. Gorkovenko, A. Dholakia, "Towards power efficiency in deep learning on Data Center Hardware," in *2019 IEEE International Conference on Big Data (Big Data) [Preprint]*, DOI: 10.1109/bigdata47090.2019.9005632.

[14] J. Chen *et al.*, "Closing the generalization gap of adaptive gradient methods in training deep neural networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence [Preprint]*, DOI: 10.24963/ijcai.2020/452.

[15] S. Hong and H. Kim, "An integrated GPU power and Performance Model", in *Proceedings of the 37th annual international symposium on Computer architecture [Preprint]*, DOI: 10.1145/1815961.1815998.

[16] Z. Tang *et al.*, "The impact of GPU DVFS on the energy and performance of Deep Learning," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems [Preprint]*, DOI: 10.1145/1815961.1815998.

[17] J. Leng *et al.*, "GPUWattch," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 487–-498, 2013, DOI: 10.1145/2508148.2485964. (in Romanian)

[18] Z. Jia, M. Maggioni, B. Staiger, and D. Paolo Scarpazza, "Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking," *CoRR* abs/1804.06826, April, 2018.

[19] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA '10)*, 2010, pp. 280–289.

[20] J. Guerreiro, A. Ilic, N. Roma, and P. Tomas, "GPGPU Power Modeling for Multi-domain Voltage-Frequency Scaling," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018.

Supplementary Information

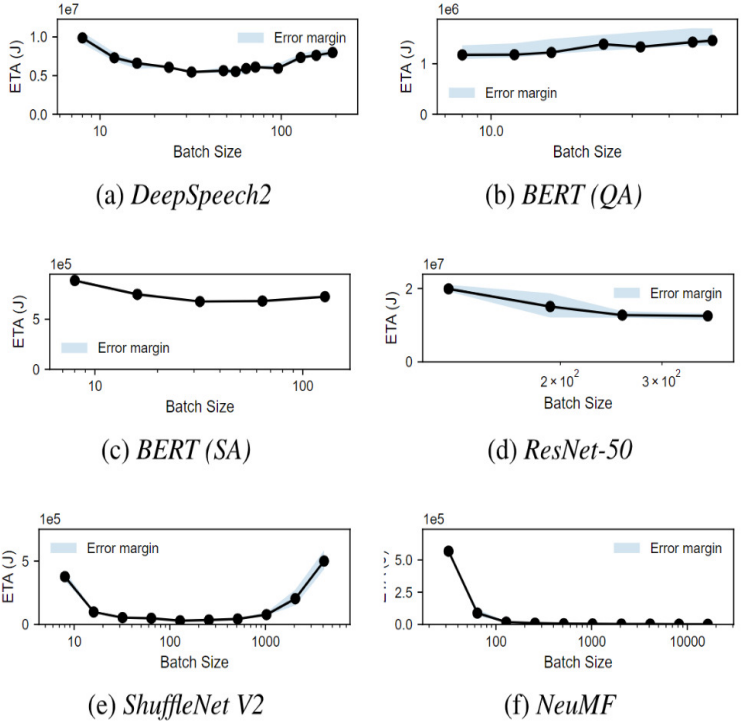


Figure 5. Energy consumed w.r.t. batch size of training workloads. The blue shade represents error margins on multiple runs.

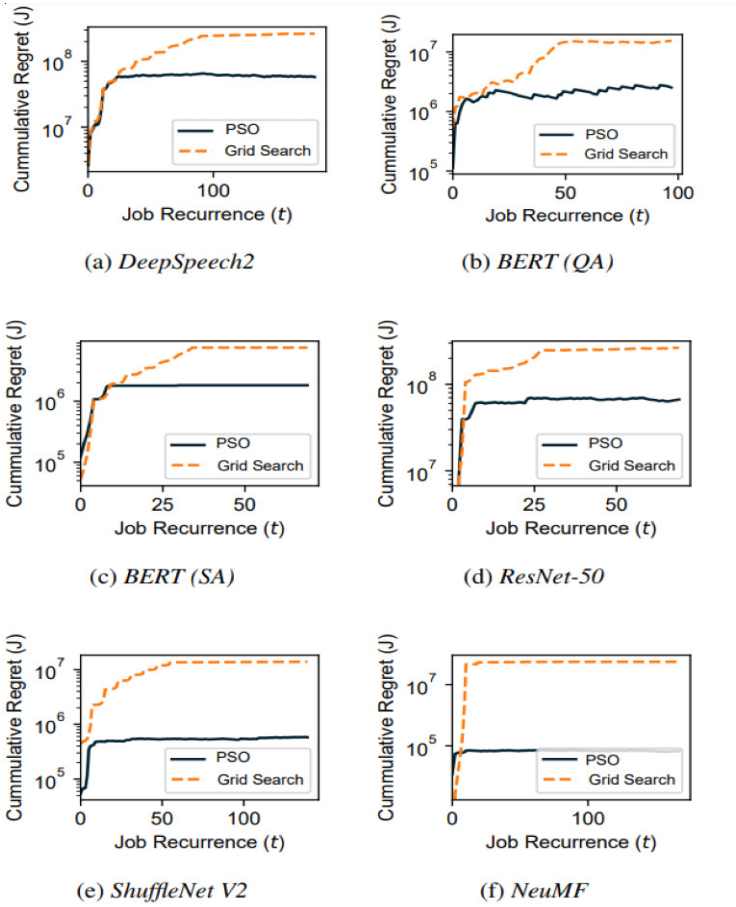


Figure 6. Progressive regret of PSO and Grid search on all training workloads

Ramesha Rehman[1], Mashood Ul Haq Chishti[2], Hamza Yamin[3]

Received August 30, 2023
Revised January 28, 2024
Accepted January 31, 2024

[1,2,3]Lahore Garrison University
Phase 6, Block C, Defence Housing Authority,
Lahore, Punjab, Pakistan 54810

Ramesha Rehman
ORCID: https://orcid.org/0009-0009-3322-117X
E–mail: rameshareh man@lgu.edu.pk

Mashood Ul Haq Chishti
ORCID: https://orcid.org/0009-0000-4024-2360
E–mail: fa19-bscs-048@lgu.edu.pk

Hamza Yamin
ORCID: https://orcid.org/0009-0008-9556-2730
E–mail: fa19-bscs-084@lgu.edu.pk