Accepted manuscript: Krzywaniak, A., Czarnul, P., Proficz, J. (2022). GPU Power Capping for Energy-Performance Trade-Offs in Training of Deep Convolutional Neural Networks for Image Recognition. In: Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds) Computational Science – ICCS 2022. ICCS 2022. Lecture Notes in Computer Science, vol 13350. Springer, Cham. https://doi.org/10.1007/978-3-031-08751-6_48

GPU power capping for energy-performance trade-offs in training of Deep Convolutional Neural Networks for image recognition

Adam Krzywaniak[0000−0003−1904−2510], Pawel Czarnul[0000−0002−4918−9196] and Jerzy Proficz[0000−0003−2975−9339]

Faculty of Electronics, Telecommunications and Informatics
Centre of Informatics — Tricity Academic Supercomputer and networK (CI TASK)
Gdansk University of Technology, Narutowicza 11/12, 80-233 Poland
adam.krzywaniak@pg.edu.pl pczarnul@eti.pg.edu.pl j.proficz@task.gda.pl

Abstract. In the paper we present performance-energy trade-off investigation of training Deep Convolutional Neural Networks for image recognition. Several representative and widely adopted network models, such as Alexnet, VGG-19, Inception V3, Inception V4, Resnet50 and Resnet152 were tested using systems with Nvidia Quadro RTX 6000 as well as Nvidia V100 GPUs. Using GPU power capping we found other than default configurations minimizing three various metrics: energy (E), energy-delay product (EDP) as well as energy-delay sum (EDS) which resulted in considerable energy savings, with a low to medium performance loss for EDP and EDS. Specifically, for Quadro 6000 and minimization of E we obtained energy savings of 28.5%–32.5%, for EDP 25%–28% of energy was saved with average 4.5%–15.4% performance loss, for EDS (k=2) 22%–27% of energy was saved with 4.5%–13.8% performance loss. For V100 we found average energy savings of 24%–33%, for EDP energy savings of 23%–27% with corresponding performance loss of 13%–21% and for EDS (k=2) 23.5%–27.3% of energy was saved with performance loss of 4.5%–13.8%.

Keywords: energy-aware computing, high performance computing, green computing, machine learning

1 Introduction

Nowadays, energy consumption has become one of the key aspects, apart from execution time and scalability, for practically all types of parallel compute intensive applications, not only traditional high performance computing (HPC) applications executed in clusters [4] but also various workloads run in a cloud environment [11, 7]. This is also becoming a very interesting and important factor for the currently very popular and time consuming training of AI models. These are mostly executed on workstations and systems that feature powerful GPUs.

In this paper we tackle investigation of performance-energy configurations that stem from using GPU power capping for several popular Deep Convolutional Neural Networks used for image recognition, typically trained on both GPUs and multi- and many-core CPUs [14, 2, 9]. This power limiting method can be used for different purposes, such as performance maximization under a defined power budget or plain energy saving. Former research confirmed the method to be reliable and provide more predictable results in terms of performance losses in comparison with a DVFS technique [17]. The main contributions of this paper in the aforementioned context include:

- deriving configurations showing larger energy savings than performance loss (percentage wise) results using GPU power capping for specific models of deep Convolutional Neural Networks for image recognition, and
- investigation of differences in impact of power capping on various network models using various high performance GPUs including Nvidia V100 and Nvidia Quadro RTX 6000 GPUs.

The following section presents the related works in the GPU power capping subject, including general approach and energy-aware machine learning issues. Then, our testbed systems and used benchmarks are described. Afterwards, in Sec. 5, we described the performed experiments, monitoring methods and the results. Finally, we provided some conclusions and future works.

2 Related works

2.1 Power capping for GPU servers

GPU power capping, or limiting, related works are mainly grouped into power capping design and implementation techniques or methods of management of heterogeneous systems under a defined power limit. The former is dedicated for achieving defined power limitation (usually in Watts). The latter generalizes the hardware/software power limiting methods designed for one component into a system wide limitation of power or energy, with additional constraints usually related to the overall performance.

A typical example of the power capping solution is a GPU-CAPP micro-architectural technique [20] dedicated for power capping limitation over GPU, with an additional objective to accelerate computations of parallel workloads. The solution assumes a direct use of hardware on-chip and global voltage regulators, which led to speed up the computations in comparison to two different fixed frequency sets, using a Nvidia GTX480 GPU system.

Another approach to imply power limitation into GPU is using a dynamic voltage and frequency scaling (DVFS) technology. A solution [8] proposed by Huang et al. uses a global-based neural network for modeling Nvidia GPU behavior under DVFS limitations, based on task characteristics, for the presented test cases, the solution enables decreasing energy along with performance improvements.

In [15] Mclaughlin et al. proposed another power capping solution dedicated for a graph traversal breadth-first algorithm. It was based on two techniques: DVFS and core scaling, features provided by an AMD A10-5800K GPU card. The described and evaluated power management algorithm, designed for maximizing efficiency under a given power cap, showed promising results for a wide range of graph data.

In [21] Tsuzuku et al. proposed power capping method for CPU-GPU heterogeneous systems. They presented performance and power consumption models used for a static solution setting initial frequencies (with Nvidia DVFS support), and then during runtime their exact settings are tuned using dynamic approach.

In [1] Ahmed et al. presented a power cap solution for CPU-GPU heterogeneous systems by defining a power cap allocation model, and implementing its simulator using discreet-event simulation engine. They performed trace-based experiments and crosschecked them with real parallel application executions on Nvidia based system. The results showed capability of the solution to decrease energy consumption of the performed computations.

In [3] Ciesielczyk et al. presented an approach limiting power usage for heterogeneous systems including GPU servers. They proposed an optimization model along with heuristic and exact solutions, where the test results using application benchmarks showed energy consumption reduction and minimal, negative impact on the performance.

In [16] Mishra et al. described how DVFS and other power-aware techniques, such as load balancing and task mapping, can influence the performance and power usage of CPU-GPU systems. They presented a collection of the up-to-date solutions and their comparative analysis. However, there was no power capping technique considered.

In [13] we performed an analysis of the GPU-based power capping for a collection of typical HPC benchmarks. We found out that using such a technique provides many possibilities of trade-off between performance and energy consumption, with different aspects of the overall evaluation, including solely energy and some in-between metrics. However, the presented results did not cover any machine learning related applications, which are addressed in this paper.

Power capping can be considered as a higher level mechanism than DVFS as it can employ core frequency scaling but potentially also shorter/longer term limits with peaks exceding the limits and thus is of interest to be explored.

2.2 Energy aware machine learning

Firstly, authors of paper [5] introduce several approaches and models of estimation of energy consumption, specifically as a key element to consider energy-aware processing in machine learning. Subsequently, they consider activities related to deep learning where energy estimation was adopted in the literature i.e. training and inference, as well as CPU and GPU based ones. Then energy estimation is applied to two real use cases. For data stream mining, HAT and VFDT algorithms obtain accuracy of over 97% for non-concept drift estimations and 56-65% for concept drift datasets. For energy estimation of inference using

a regression model they obtained accuracy of 73.7% for Inception-V3, 63% for MobileNet, 70% for DenseNet.

Furthermore, authors of [14] investigate power and energy of using CNN models on CPU and GPU systems. Specifically, thorough analysis includes tests on Xeon CPU, K20 GPU, Titan X GPU based systems, for various frameworks on GPUs (Caffee, Torch, MXNet,), various libraries on CPU (Atlas, OpenBLAS and MKL). Additionally, breakdown of energy and power per layers of a model has been provided, for various batch sizes, considering HyperThreading as well as various memory and core frequencies. Results could be used for setting configurations for minimization of energy on CPU and GPU systems. However, further exploration considering training times and accuracies for time limited training could be extensions of that work. On the other hand, in paper [2] the author proposed NeuralPower—a framework based on sparse polynomial regression aimed at prediction of power, runtime, and energy consumption of CNNs used on a GPU. Specifically, the author was able to obtain average accuracy of over 88% for runtime and power for tested CNN architectures.

In paper [22], authors proposed GPOEO, a new online energy optimization framework for iterative machine learning applications run on GPUs. The tool is able to detect iterative phases, measures performance and energy used. Firstly, in an offline stage, it collects performance metrics and energy for various frequencies of SM and memory and then in the online stage applies the selected frequencies to optimize a function of energy and time. Similarly, in paper [24], Zou et al. proposed a power limiting solution, using resource utilization as an indicator of iteration bounds. The proposed method used GPU DVFS mechanisms to set up a defined power limit with performance maximization or performance degradation level with minimizing the energy consumption. Our work is focused on research of the power limiting techniques introduced in the tested GPU cards. We demonstrated and evaluated these mechanisms against the ML applications, however, they are more general and are useful for other types of workloads.

In paper [10] authors optimize deep learning at a higher level i.e. provide an allocation method for a GPU cluster for deep learning jobs (training and inference) minimizing energy consumption and meeting performance requirements. The approach uses a mixed-integer nonlinear problem (MINLP) formulation. The solution is able to turn off the nodes that have no DL jobs. Experiments have been conducted using GTX 1000 and RTX 2000 series GPUs showing e.g. energy savings of 43% compared to PA-MBT and 15% compared to EPRONS approaches.

Another approach, related to optimization of selected steps of training CNNs meant for energy consumption reduction was proposed in [23]. Specifically, three types of optimizations were proposed: stochastic skipping mini-batches with 0.5 probability, selection of a different subset of CNN layers for updates, and computing the sign of a gradient without computing the full gradient. The authors have demonstrated results from an FPGA board e.g. for ResNet-74 trained using CIFAR-10, energy was saved of over 90% and 60%, with a top-1 accuracy loss of about 2% and 1.2%.

In paper [18], authors explored energy-aware optimization of running deep learning workloads using GPU power capping as well as frequency capping for reduction of Energy-to-Solution (ETS) and Energy-Delay-Product (EDP) on a system with POWER8 and one P100 GPU. The authors demonstrated savings of ETS up to 27% and half of the examples decreasing EDP.

3 Testbed systems

Table 1 presents testbed systems used for the research in this paper. We have performed the tests on two systems with modern Nvidia GPUs. Testbed 1 has 8 Nvidia Quadro RTX 6000 (Turing architecture) cards which were released in August 2018. Testbed 2 has 8 Nvidia Tesla V100-SXM2-16GB (Volta architecture) cards which were firstly released in June 2017. Both systems run on Ubuntu OS (20.4 and 18.4 respectively), both have CUDA 11 installed (11.5 and 11.2 respectively) and both has python 3.8 installed.

Table 1. Testbed configurations

Testbed system	Testbed 1: Quadro 6000	Testbed 2: V100
CPU model	2 x Intel® Xeon® Silver 4210 CPU @ 2.20GHz	2 x Intel® Xeon® CPU E5-2686 v4 @ 2.30GHz
CPU cores [physical / logical]	2 x [10 / 20]	2 x [16 / 32]
System memory size (RAM)	376 GB RAM	480 GB RAM
GPU model	Nvidia Quadro RTX 6000 (Turing)	Nvidia Tesla V100-SXM2-16GB (Volta)
GPU memory	24 GB GDDR6	16 GB HBM2
GPU default power limit	260 W	300 W
GPU available power limit range	100 W–260 W	150 W–300 W
Cuda cores	4608	5120
Core clock speed	1440 MHz	1370 MHz
Operating System	Ubuntu 20.04.3 LTS	Ubuntu 18.04.6 LTS
Python version	3.8.10	3.8.12
Cuda version	V11.5.119	V11.2.152
Tensorflow version	2.8.0	2.4.0

4 Deep CNN benchmarks used for experiments

In the experiments we have used six popular Convolutional Neural Networks (CNN) designed for image recognition. The CNNs we have chosen are: Alexnet (presented in 2012, 63e6 parameters), VGG-19 (2014, 143e6 parameters), Inception V3 (2015, 24e6 parameters), Inception V4 (2016, 43e6 parameters), Resnet50 (2015, 26e6 parameters) and Resnet152 (2015, 60e6 parameters). The models used by us as a representative set of CNN benchmarks for the experiments were trained with synthetic ImageNet dataset. We have reduced the number of synthetic data samples to 100,000 for V100 system and to 32,000 for Quadro 6000 system. Each benchmark was executed on a single GPU, with either the batch size 128 (for Alexnet, VGG-19, Inception V3, Resnet50) or batch size 64 (for Inception V4 and Resnet152) where the model reached the GPU memory limit. The code of the benchmarks were taken from the official Tensorflow benchmarks github available online[1].

[1] https://github.com/tensorflow/benchmarks/

5 Experiments and results

5.1 Power monitoring and controlling methodology

For the Nvidia GPUs power monitoring and controlling we have used the Application Programmable Interface (API) exposed in Nvidia Management Library (NVML). The API allows to read the current power usage in milliwatts using `nvmlDeviceGetPowerUsage` and – according to Nvidia's documentation – it is accurate to within ±5% of current power draw. The GPU power limit is set via `nvmlDeviceSetPowerManagementLimit` API call. The power consumption is read with the fixed $T = 0.5\,s$ period. The energy consumption E is calculated as a definite integral of current power P and the sampling period T products for the given interval identical with application total execution time. The application execution time is measured using C++ `std::chrono::high_resolution_clock`. The whole process of power limits exploration and evaluation of the performance-energy trade-offs was fully automated and enclosed within a software tool we call StaticEnergyProfiler (StEP). The tool was originally introduced as EnergyProfiler in [12] as an automatic tool for exploration of software power caps in Intel CPUs. Since then we have extended the tool with Nvidia GPU support and published the code with new naming convention in an open source repository with Software Power Limiting Tools (SPLiT) suite available online[2].

All the results presented in Section 5.3 and 5.4 were obtained with the StEP automatic tool. For each system we have evaluated the full available power limits range (100 W–260 W for Quadro 6000 system and 150 W–300 W for V100 system) with a 5 W step. For each power limit we have executed 5 test runs and we present an average result. Each of six benchmark CNNs was executed in a training mode for just a one epoch with reduced ImageNet synthetic dataset. We assume the potential real life use case of proposed StEP tool as a fast way of power limits exploration with performance-energy trade-offs evaluation as an initial step before launching the full CNN training for the target number of epochs and full dataset with a power limit selected by user.

It should be noted that our approach extends benchmarking of various models for various hyperparameters and compute devices such as in [9] by generating more configurations by power capping. However, this does not result in changes to the important model accuracy, precision or recall values.

5.2 Target metrics for bi-objective energy-performance optimization

Exploration of available power limits results in a series of the energy-performance results which may form a Pareto-optimal front. If we target for a bi-objective optimization which is considering both energy savings and performance loss any result included in a Pareto-optimal front might be chosen as a desired solution. In order to evaluate the results within Pareto-optimal front we may want to evaluate

[2] https://projects.task.gda.pl/akrz/split/

each result using various target metrics. One of the simplest is consideration of only the total energy consumption where we seek for energy minimum:

$$M_E(E,t) = E \tag{1}$$

Another metric that we would like to evaluate is already used by us in [13] total energy and total execution time product which aims for bi-objective optimization. The energy-performance product is known in the literature as energy-delay product (EDP)[6]:

$$M_{EDP}(E,t) = Et \tag{2}$$

Finally, the third bi-objective oriented metric for energy aware optimization which we will evaluate in this paper is energy-delay sum (EDS) proposed by Roberts et al. in [19]:

$$M_{EDS}(E,t) = \alpha E + \beta t \tag{3}$$

EDS is a weighted sum of energy E and total execution time t. The metric assumes that we adjust the proposed weights α and β the way that we arbitrarily choose a theoretical acceptable time increase for the abstract scenario when the energy consumption would be 0. Since we need to know the theoretical time increase represented by k parameter we also need to know the reference time and energy result. Thus, the EDS metric may be only evaluated as a relative value based on some reference result. Taking the two points with a reference result and a theoretical result of $k \cdot t_{ref}$ time and zero energy we may construct an equation considering the metric as a linear function based on two points: $E - E_{ref} = \frac{0-E_{ref}}{k \cdot t_{ref} - t_{ref}} \cdot (t - t_{ref})$ out of which after transformation we obtain the formula:

$$1 = \frac{k-1}{k \cdot E_{ref}} \cdot E + \frac{1}{k \cdot t_{ref}} \cdot t \tag{4}$$

and we may read the $\alpha = \frac{k-1}{k \cdot E_{ref}}$ and $\beta = \frac{1}{k \cdot t_{ref}}$ weights where k is a theoretical accepted time increase.

Any result for which the equation 4 is true will indicate that the performance loss with given energy savings is proportionally equivalent to the reference result with respect to proportion defined by k parameter. Any result for which the EDS metric value is lower than 1 will be considered as better than the reference result in terms of EDS metric evaluation for particular k parameter value.

Figure 1 presents graphical visualization of three target metrics aforementioned above. The axes represent normalized energy and normalized execution time. The value of 1 on each axis represent the reference result obtained for the default system setup. Each data point represents a single energy-performance result obtained with some power limit applied. We present the E metric as a horizontal line with the value of 1 on the normalized energy axis. The EDP metric is a hyperbole which pass through point with (1,1) values. The EDS metric is a linear function which is presented by us for two values of k parameter ($k = 1.5$ and $k = 2.0$). Any result which is below the metric line will be accepted by this metric as a better than default result. The result point which euclidean distance from the metric line is the biggest will be considered as a result with minimal value for this metric.

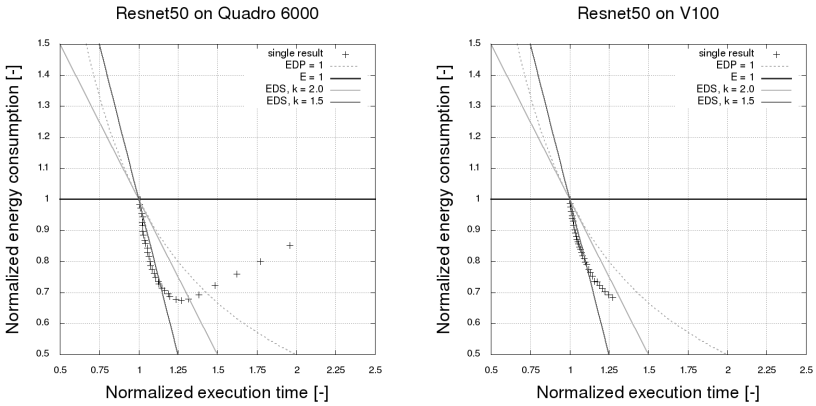

Fig. 1. Graphical visualization of selected target metrics for exemplary series of energy-performance results obtained for 1-epoch training of Resnet50 with different power limits executed on both Quadro 6000 (on the left) and V100 (on the right) systems.

5.3 Results obtained for Quadro 6000 system

Figure 2 presents the relative results obtained for all six CNN benchmarks executed on Quadro 6000 system. Table 2 presents specific absolute results with all the target metrics optimal results and their corresponding power limits obtained for the Quadro 6000 system. For all of the tested CNN benchmarks any power limit lower than default one results in a lower energy (E) value. The minimum of energy for each of six tested CNN benchmarks on Quadro 6000 system can be found somewhere within a range of power limits of 120 W–130 W. The typical obtained minimum of energy results in 28.5%–32.5% of energy saved.

The EDP metric, which represents the energy and time product, has a value less than 1 for any of tested CNN benchmarks typically in the range of power limits of 120 W–255 W. That means that for any power limit within that range we obtain the EDP value better than the default one. Typically the EDP minimum can be found within the power limits range of 140 W–170 W on the Quadro 6000 system. The aforementioned EDP minimum results mostly in 25%–28% of energy saved with an average 4.5%–15.4% of performance loss. Considering the EDP metric minimum we get the trade-off of much higher energy savings than the cost of performance loss we bear.

The EDS metric was evaluated for two values of the k parameter. For $k = 1.5$ the values of a metric which are better than the reference result are mostly found when power limit is set in the range of 160 W–255 W. For $k = 2.0$ the range is wider and starts at 140 W. Typically, the minimum of EDS(k=1.5) is found in the power limits range of 190 W–215 W and results on average in 12%–26% of energy savings with only 3.9%–6.8% of performance loss. For EDS(k=2.0) the typical power limits range for finding a minimum of this metric is within 140 W–180 W what results on average in 22%–27% of energy savings with 4.5%–13.8% of performance loss. Minimization of EDS metric for both k parameter values

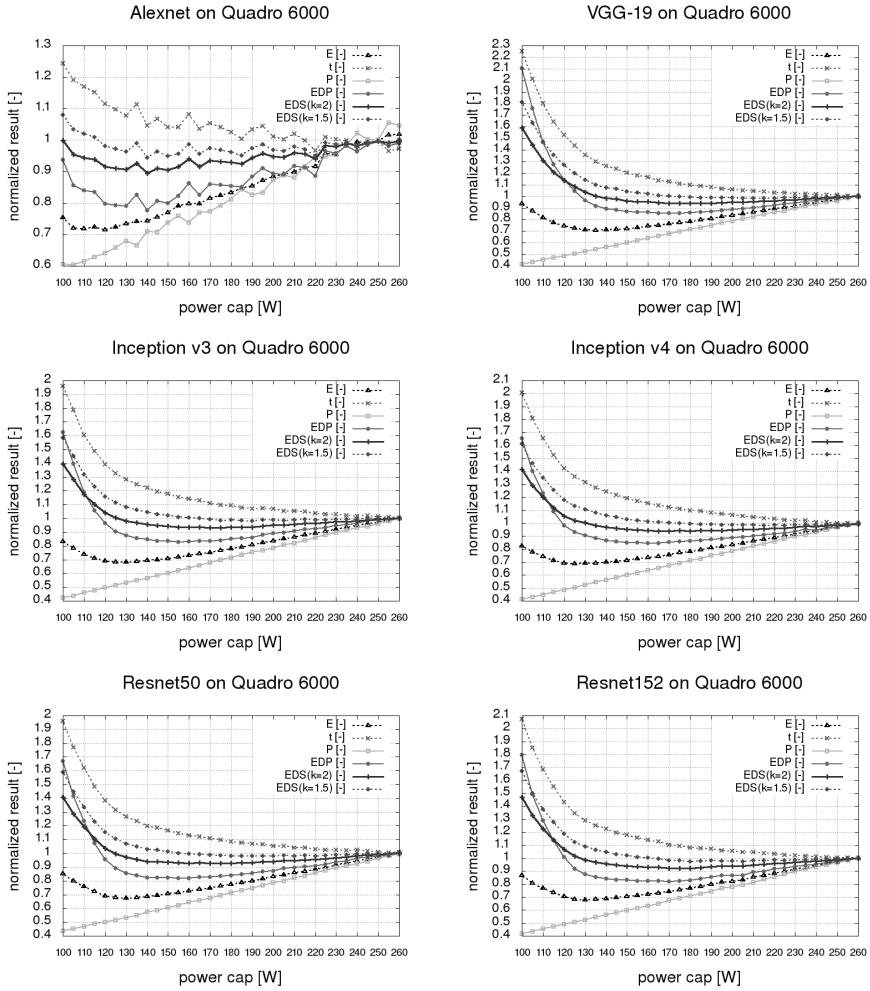

Fig. 2. Normalized results (average power, total energy consumption, total execution time, EDP and EDS metric) for all six CNN benchmarks (Alexnet, VGG-19, Inception 3, Inception 4, Resnet50, Resnet152) obtained for different power caps in range 100 W–260 W applied to Quadro 6000 system.

allows then for significant energy savings with minor or negligible performance loss.

5.4 Results obtained for V100 system

Figure 3 presents relative results obtained for all six CNN benchmarks executed on V100 system. Table 3 presents specific absolute results with all the target metrics optimal results and their corresponding power limits obtained for the V100 system.

Table 2. Results of minimization of selected three target metrics (E, EDP, EDS) for all six CNN benchmarks (Alexnet, VGG-19, Inception 3, Inception 4, Resnet50, Resnet152) obtained for Quadro 6000 system with synthetic ImageNet dataset reduced to 32,000 of samples.

CNN benchmark	Target Metric	Power cap [W]	Average Power [W]	Total Energy [kJ]	Total Energy vs default [%]	Total Time [s]	Total Time vs default [%]
Alexnet	default	260	116.459	3.288	-	28.2	-
	min E	120	71.9	2.336	-28.5	32.5	+11.5
	min EDP	140	79.6	2.427	-25.7	30.5	+4.5
	min EDS(k=2.0)	140	79.6	2.427	-25.7	30.5	+4.5
	min EDS(k=1.5)	140	79.6	2.427	-25.7	30.5	+4.5
VGG-19	default	260	233.6	51.516	-	220.5	-
	min E	135	126.9	36.342	-29.9	286.4	+29.9
	min EDP	165	153.8	38.646	-25.0	251.2	+13.9
	min EDS(k=2.0)	175	162.5	39.734	-22.9	244.5	+10.9
	min EDS(k=1.5)	205	188.5	43.686	-15.2	231.8	+5.1
Inception3	default	260	224.4	41.382	-	184.4	-
	min E	125	115.0	28.157	-32.0	244.9	+32.8
	min EDP	155	139.8	29.720	-28.2	212.6	+15.3
	min EDS(k=2.0)	175	156.6	31.668	-23.5	202.2	+9.7
	min EDS(k=1.5)	190	169.3	33.346	-19.4	197.0	+6.8
Inception4	default	260	235.9	82.316	-	348.9	-
	min E	125	119.1	56.450	-31.4	474.0	+35.8
	min EDP	160	149.9	60.360	-26.7	402.6	+15.4
	min EDS(k=2.0)	165	154.1	61.202	-25.7	397.3	+13.8
	min EDS(k=1.5)	215	198.4	71.921	-12.6	362.4	+3.9
Resnet50	default	260	215.5	27.068	-	125.4	-
	min E	130	114.8	18.264	-32.5	159.1	+26.8
	min EDP	155	134.5	19.323	-28.6	143.7	+14.5
	min EDS(k=2.0)	160	139.1	19.683	-27.3	141.2	+12.8
	min EDS(k=1.5)	200	169.9	22.445	-17.1	132.1	+5.3
Resnet152	default	260	230.4	64.522	-	280.2	-
	min E	130	121.1	43.839	-32.1	362.0	+29.2
	min EDP	170	155.1	47.958	-25.7	309.2	+10.4
	min EDS(k=2.0)	180	163.6	49.471	-23.4	302.4	+7.9
	min EDS(k=1.5)	205	183.3	53.645	-16.9	292.7	+4.5

Similarly to the Quadro 6000 system, on the V100 system we can also observe a wide range of performance-energy trade-offs which may be obtained when applying power limits. What is different between V100 and Quadro 6000 is that V100 has much higher lowest power limit value available and therefore in most of tested CNN benchmark cases the minimal value for the energy (E) metric is obtained for the lowest available power limit which is 150 W. The minimal energy (E) value for the V100 system allows for average energy savings of 24%–33%.

The EDP bi-objective metric which represents the energy and time product has its minimum for the most of tested CNN benchmarks when the power limit is set within range 170 W–180 W. We observe for EDP metric minimum the average energy savings of 23%–27% with corresponding performance loss of 13%–21%. This shows that for the V100 system the EDP metric minimum allows again for obtaining interesting and satisfactory trade-offs when proportionally more energy may be saved than we loose on execution time increase.

The EDS metric minimum for the V100 system was also evaluated for two values of the k parameter. For $k = 1.5$ most of the tests with selected CNN benchmarks show that the results better than the reference one which means results with the value of EDS metric less than 1 were found for the V100 system mostly when power limit was set within 210 W–295 W. The optimal solutions (min of EDS) for $k = 1.5$ were found for power limits set within the range of 190 W–215 W. We observed significant energy savings of 12.6%–25.7% with only a minor performance loss of 3.9%–6.8%. For $k = 2.0$ the range of power limits

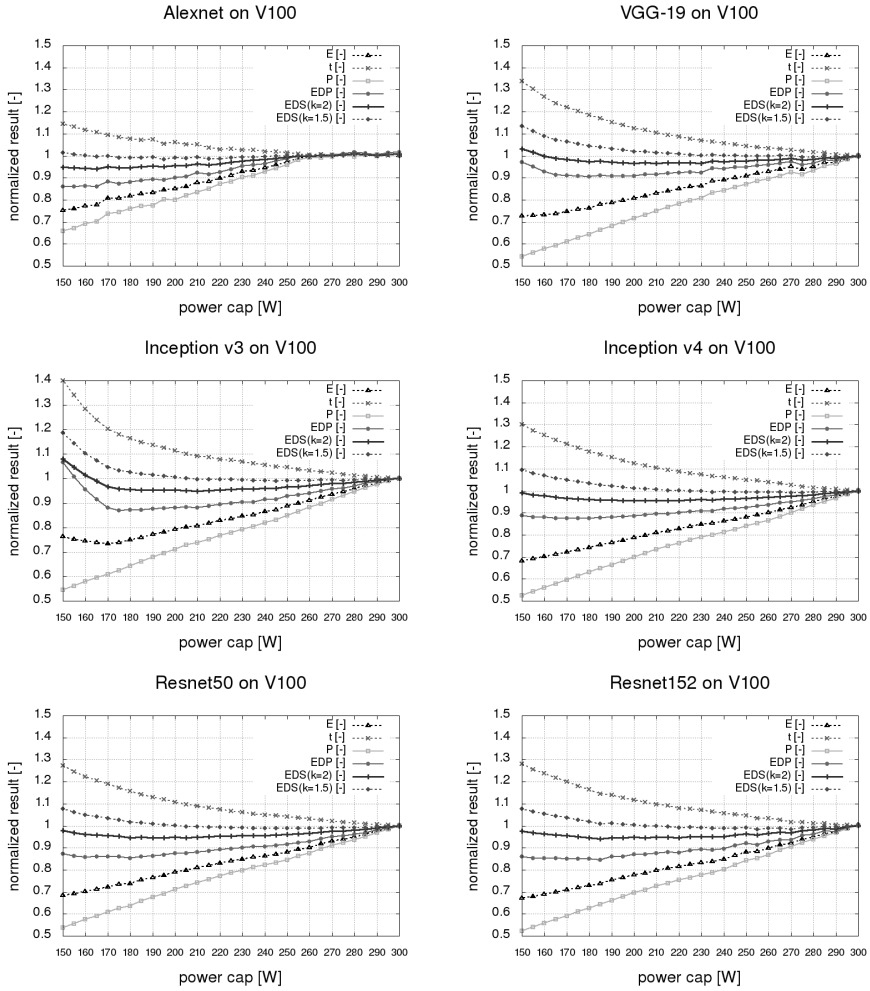

Fig. 3. Normalized results (average power, total energy consumption, total execution time, EDP and EDS metric) for all six CNN benchmarks (Alexnet, VGG-19, Inception 3, Inception 4, Resnet50, Resnet152) obtained for different power caps in range 100 W–260 W applied to V100 system.

with better than default results was covered by the range of 150 W–295 W. The minimal EDS(k=2.0) metric values for most of the tested CNNs were obtained for power limits set within the range of 160 W–180 W. The EDS(k=2.0) metric allowed for finding the solutions where we can save 23.5%–27.3% with performance loss of 4.5%–13.8%. Thus, the EDS metric minimized for V100 again allowed for significant energy savings with minor performance loss.

Unlike Quadro 6000, for V100 the E metric has its optimal value (except for Inception v3) for minimal power limit. The reason for that seem to be much higher minimal power limit available on V100 (150 W) compared to Quadro 6000

Table 3. Results of optimization of selected four target metrics (E, EDP, EDS(k=2.0), EDS(k=1.5)) for all six CNN benchmarks (Alexnet, VGG-19, Inception 3, Inception 4, Resnet50, Resnet152) obtained for V100 system with synthetic ImageNet dataset reduced to 100,000 of samples.

CNN benchmark	Target Metric	Power cap [W]	Average Power [W]	Total Energy [kJ]	Total Energy vs default [%]	Total Time [s]	Total Time vs default [%]
Alexnet	default	300	181.4	7.206	-	39.7	-
	min E	150	119.4	5.425	-24.7	45.4	+14.4
	min EDP	155	121.9	5.477	-24.0	44.9	+13.1
	min EDS(k=2.0)	165	127.5	5.603	-22.2	43.9	+10.6
	min EDS(k=1.5)	195	145.5	6.091	-15.5	41.9	+5.4
VGG-19	default	300	264.9	130.369	-	492.5	-
	min E	150	143.8	94.703	-27.4	658.6	+33.8
	min EDP	180	170.5	99.494	-23.7	583.5	+18.6
	min EDS(k=2.0)	200	189.9	105.181	-19.3	553.8	+12.5
	min EDS(k=1.5)	275	242.9	122.194	-6.3	503.1	+2.2
Inception3	default	300	265.8	104.941	-	394.8	-
	min E	170	160.9	75.608	-26.7	470.0	+20.2
	min EDP	170	160.9	75.608	-26.7	470.0	+20.2
	min EDS(k=2.0)	210	195.0	83.257	-19.4	427.0	+9.2
	min EDS(k=1.5)	245	219.7	90.106	-12.7	410.2	+4.9
Inception4	default	300	273.1	230.8	-	845.1	-
	min E	150	143.4	157.5	-31.8	1098.5	+30.0
	min EDP	170	162.6	166.602	-27.8	1024.4	+21.2
	min EDS(k=2.0)	200	190.9	181.532	-21.4	950.7	+12.5
	min EDS(k=1.5)	270	245.9	213.249	-7.6	867.4	+2.6
Resnet50	default	300	262.3	69.886	-	266.4	-
	min E	150	141.2	47.866	-31.5	339.0	+27.3
	min EDP	180	166.7	51.431	-26.4	308.4	+15.8
	min EDS(k=2.0)	190	177.6	53.430	-23.5	300.8	+12.9
	min EDS(k=1.5)	260	229.5	63.002	-9.9	274.5	+3.0
Resnet152	default	300	273.7	186.948	-	683.0	-
	min E	150	143.2	125.352	-33.0	875.3	+28.2
	min EDP	185	176.3	137.879	-26.2	782.1	+14.5
	min EDS(k=2.0)	185	176.3	137.879	-26.2	782.1	+14.5
	min EDS(k=1.5)	255	233.7	165.032	-11.7	706.2	+3.4

(100 W). If the V100 offered a wider range of available power limits, with the lower possible minimum, we could probably observe slightly lower minimal energy consumption with a bigger performance penalty. Then, the E characteristics would probably have their own clear minimal values for the corresponding power limits, that could be different from the imposed by the manufacturer minimal power caps.

6 Conclusions and future work

In this paper, we investigated performance-energy trade-offs for a collection of popular machine learning architectures, supported by power limiting (a.k.a. capping) features of two Nvidia GPU cards: V100 and Quadro 6000. The performed benchmarks cover training of Deep Convolutional Neural Networks, such as Alexnet, Resnet or Inception, which were used used for image recognition.

The results proved that power capping and its underlying implementation limiting computational use of GPU resources can result in limited performance and consequently prolonged execution time but for some configurations allows even larger, percentage wise, reduction of energy used. Power limiting can imply much lower energy consumption (up to 33% for V100 and Quadro 6000), along with a low to medium performance penalty. Moreover, usage of well defined metrics enabled this bi-objective optimization to support the selection of the desired configuration.

The future works will cover the following areas:

- automation of the selection of a power-related configuration, based on a chosen metric, supporting the dynamic adaptation to the application behavior,
- CPU/GPU resource allocation, depending on the performance and energy requirements defined for a whole HPC system,
- modeling and simulation of the GPU/CPU behavior for selection of a static power-related configuration.

Basing on our research results, we are convinced that the power-performance optimization can significantly decrease the carbon footprint and energy cost of the machine learning solutions.

References

1. Ahmed, K., Tasnim, S., Yoshii, K.: Energy-efficient heterogeneous computing of parallel applications via power capping. In: 2020 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 1237–1242. IEEE (dec 2020)
2. Cai, E.: Learning-based power and runtime modeling for convolutional neural networks, data Analysis Project, Carnegie Mellon University
3. Ciesielczyk, T., Cabrera, A., Oleksiak, A., Piątek, W., Waligóra, G., Almeida, F., Blanco, V.: An approach to reduce energy consumption and performance losses on heterogeneous servers using power capping. Journal of Scheduling **24**(5), 489–505 (oct 2021)
4. Czarnul, P., Proficz, J., Krzywaniak, A.: Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. Sci. Program. **2019**, 8348791:1–8348791:19 (2019)
5. García-Martín, E., Rodrigues, C.F., Riley, G., Grahn, H.: Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing **134**, 75–88 (2019)
6. Gonzalez, R., Horowitz, M.: Energy dissipation in general purpose microprocessors. IEEE Journal of Solid-State Circuits **31**(9), 1277–1284 (1996)
7. Hogade, N., Pasricha, S., Siegel, H.J.: Energy and network aware workload management for geographically distributed data centers. CoRR **abs/2106.00066** (2021)
8. Huang, Y., Guo, B., Shen, Y.: GPU energy consumption optimization with a global-based neural network method. IEEE Access **7**, 64303–64314 (2019)
9. Jabłońska, K., Czarnul, P.: Benchmarking deep neural network training using multi- and many-core processors. In: Saeed, K., Dvorský, J. (eds.) Computer Information Systems and Industrial Management. pp. 230–242. Springer International Publishing, Cham (2020)
10. Kang, D.K., Lee, K.B., Kim, Y.C.: Cost efficient GPU cluster management for training and inference of deep learning. Energies **15**(2) (2022)
11. Khan, T., Tian, W., Ilager, S., Buyya, R.: Workload forecasting and energy state estimation in cloud data centres: Ml-centric approach. Future Generation Computer Systems **128**, 320–332 (2022)
12. Krzywaniak, A., Czarnul, P., Proficz, J.: Extended investigation of performance-energy trade-offs under power capping in hpc environments. In: 2019 International Conference on High Performance Computing and Simulation (HPCS). pp. 440–447 (2019)

13. Krzywaniak, A., Czarnul, P.: Performance/energy aware optimization of parallel applications on GPUs under power capping. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12044 LNCS, pp. 123–133 (June 2020)
14. Li, D., Chen, X., Becchi, M., Zong, Z.: Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In: 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom). pp. 477–484 (2016)
15. Mclaughlin, A., Paul, I., Greathouse, J.L.: A power characterization and management of GPU graph traversal (2014)
16. Mishra, A., Khare, N.: Analysis of DVFS techniques for improving the GPU energy efficiency. Open Journal of Energy Efficiency **04**(04), 77–86 (2015)
17. Patki, T., Frye, Z., Bhatia, H., Di Natale, F., Glosli, J., Ingolfsson, H., Rountree, B.: Comparing GPU power and frequency capping: A case study with the MuMMI workflow. In: 2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS). pp. 31–39. IEEE (nov 2019). https://doi.org/10.1109/WORKS49585.2019.00009, https://ieeexplore.ieee.org/document/8943552/
18. Pérez, M.M., Seiler, N., Bederián, C.S., Wolovick, N., Vega, A.: Power Efficiency Analysis of a Deep Learning Workload on an IBM "Minsky" Platform, Communications in Computer and Information Science, vol. 979, chap. High Performance Computing - 5th Latin American Conference, CARLA 2018, Bucaramanga, Colombia, September 26-28, 2018, Revised Selected Papers, p. 255–262. Springer
19. Roberts, S.I., Wright, S.A., Fahmy, S.A., Jarvis, S.A.: Metrics for energy-aware software optimisation. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 10266 LNCS, pp. 413–430. Springer Verlag (2017)
20. Straube, K., Lowe-Power, J., Nitta, C., Farrens, M., Akella, V.: Improving provisioned power efficiency in HPC systems with GPU-CAPP. In: 2018 IEEE 25th International Conference on High Performance Computing (HiPC). pp. 112–122. IEEE (dec 2018)
21. Tsuzuku, K., Endo, T.: Power capping of CPU-GPU heterogeneous systems using power and performance models. In: Proceedings of the 4th International Conference on Smart Cities and Green ICT Systems. pp. 226–233. SCITEPRESS - Science and and Technology Publications (2015)
22. Wang, F., Zhang, W., Lai, S., Hao, M., Wang, Z.: Dynamic gpu energy optimization for machine learning training workloads. IEEE Transactions on Parallel and Distributed Systems p. 1–1 (2022)
23. Wang, Y., Jiang, Z., Chen, X., Xu, P., Zhao, Y., Lin, Y., Wang, Z.: E2-train: Energy-efficient deep network training with data-, model-, and algorithm-level saving. CoRR **abs/1910.13349** (2019)
24. Zou, P., Li, A., Barker, K., Ge, R.: Indicator-directed dynamic power management for iterative workloads on GPU-accelerated systems. Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020 pp. 559–568 (2020). https://doi.org/10.1109/CCGrid49817.2020.00-37