Evaluating the energy impact of device parameters for DNN inference on edge

Anurag Dutt*, Sri Pramodh Rachuri*, Ashley Lobo, Nazeer Shaik, Anshul Gandhi, Zhenhua Liu
Stony Brook University, Stony Brook, NY, 11794
{anurag.dutt, sripramodh.rachuri, ashley.lobo, nazeer.shaik, anshul.gandhi, zhenhua.liu}@stonybrook.edu

***Abstract*—With advancements in edge inference accelerating the shift from cloud to edge computing, there is a need for research on sustainable edge deployments of DNN workloads. However, the energy consumption of DNN workload execution is affected by numerous parameters and knobs. This paper studies the impact of hardware knobs (CPU and GPU frequency) across six different DNN inference workloads on two different Jetson edge devices. We find that default parameter settings need not be energy optimal; for example, tuning the CPU and GPU frequency can save as much as 19% energy over DVFS.**

***Index Terms*—Energy efficiency, edge computing, DNN inference, measurements.**

I. Introduction

In recent years, Deep Neural Networks (DNN), including Large Language Models (LLMs), have gained significant traction. While the training of these models has received attention, research on efficient deployment for *inference*, especially at the *edge*, is still ongoing [2], [11]. Edge computing is the layer closest to the users in IoT stack for real-time data processing, aimed at bringing computation closer to users. The edge devices are often deployed using batteries or solar panels for various critical applications in remote locations, which motivates our investigation in this paper on the *energy consumption* for these devices [8].

Efficient deployment of DNN on edge faces challenges due to the complex models straining the limited edge resources and the need to tune the various available hardware parameters. In fact, parameters like GPU frequency can impact energy, power, and inference time. This impact could also be *non-monotonic* making it difficult to find the optimal settings. Consequently, practitioners often settle for sub-optimal energy savings by not carefully tuning these parameters and resorting to existing tools, such as Dynamic Voltage and Frequency Scaling (DVFS), for configuring hardware knobs [18].

In this paper, we study the impact of hardware knobs on the energy consumption of DNN inference, specifically for edge devices. There has been some related work recently that looks at the energy efficiency of DNN inference. Holly *et al.* [4] profile Mobilenet-V2 as a function of hardware parameters (*e.g.*, number of cores); however, the workloads are limited to CNNs. DeepEdgeBench [1] analyzes the power consumption of running DNN models on different edge devices, but the authors do not investigate the impact of hardware parameter changes on energy. Likewise, there has been a lot of recent work on analyzing energy-efficient *training* of DNN workloads on edge (*e.g.*, Prashanthi *et al.* [14], Trainer [17], EfficientGrad [5]) and on servers (*e.g.*, Zeus [18]). However, insights from energy analysis for training need not translate to inference since training is more sensitive to memory, network delays, and parallelization.

TABLE I: Specifications for Jetson Nano and Xavier NX [9], [10].

Specification	Jetson Nano	Jetson Xavier NX
CPU	4-core ARM A57	8-core Nvidia Carmel
CPU Freq. range	102 MHz – 1.48 GHz	115 MHz – 1.9 GHz
CPU Freq. step	100 MHz (15 steps)	77 MHz (25 steps)
GPU	Nvidia Maxwell	NVIDIA Volta
CUDA Cores	128	384
Tensor Cores	-	48
Memory	4 GB LPDDR4	8 GB LPDDR4
GPU Freq. range	76 MHz – 921 MHz	114 MHz – 1.1 GHz
GPU Freq. steps	77 MHz (count 12)	90 MHz (count 15)
Throughput	472 GFLOPs	21 TOPs
Power Modes	5W, 10W	10W, 15W
Jetpack	4.6.3 [L4T 32.7.3]	4.5.1 [32.5.2]
Framework	PyTorch 1.10.0, Darknet (OpenCV 4.6.0)	PyTorch 1.10.0, Darknet (OpenCV 4.6.0)
Libraries	CUDA 10.2 + cuDNN 8.2.1	CUDA 10.2 + cuDNN 8.0.0
Operating Sys.	Ubuntu 18.04	Ubuntu 18.04

We conduct our empirical study using smart edge devices with compact ARM-based microprocessors with GPU acceleration. For our experiments, we use two devices: (1) Jetson Nano, an entry-level edge device, and (2) Jetson Xavier NX, a mid-tier, more powerful version, as shown in Table I. Both devices are capable of serving DNN models for a variety of practical application tasks. Our experimental results show that we can reduce inference energy by as much as 19% compared to DVFS by optimally tuning the CPU and GPU frequencies.

II. Experiment Design

Both Jetson Nano and Xavier NX provide multiple onboard sensors measuring power for different components and can be accessed through an `I2C` interface. We log power consumed by the entire module every 100ms with an overhead of less than 0.5%. We found that more frequent polling (*e.g.*, 10ms or 1ms) can result in higher energy overheads of more than 2%.

This work was supported by NSF grants 2214980, 2106434, and 1750109.
*First two authors contributed equally to this work.

TABLE II: DNN workload specifications

Model	Layers	Params	Ops (GFLOPs)	Batch Size	Input
AlexNet	8	61M	0.727	4, 8, 16, 32, 64	Tensor (3,224,224)
ResNet-18	18	11M	2	4, 8, 16, 32, 64	Tensor (3,224,224)
MobileNet-V2	53	3.4M	0.57	4, 8, 12	Tensor (3,224,224)
YOLOv4-Tiny	29	6.1M	6.9	4, 8, 16, 32, 64	Tensor (3,416,416)
BERT-Tiny	4	4.4M	0.0353	4, 8, 16, 32, 64	String (512 words, 1.1kb)
DistilBERT	6	43.2M	4.3	4, 8, 16	String (512 words, 1.1kb)

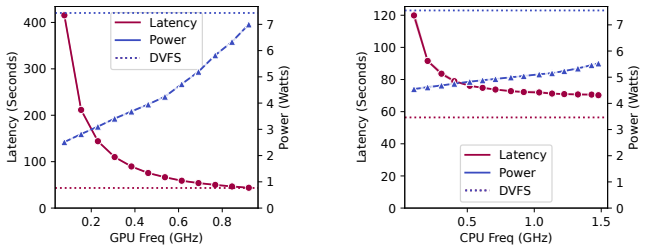
Both devices have three main hardware knobs that can be changed during runtime—CPU clock frequency, GPU clock frequency, and the number of cores. We focus on the first two since the number of cores did not impact the power consumption in our experiments as the DNN models are implemented in Python, which is effectively single-threaded due its Global Interpreter Lock (GIL) mechanism. Both devices offer a large range of CPU and GPU frequencies. Jetson Nano has 180 possible GPU and CPU frequency combinations and Xavier NX has 375 possible configurations. We use `sysfs` APIs to set static CPU and GPU frequencies. Baselines were established using the default DVFS module.

We executed the DNN inference process and logged the workload execution checkpoints and their requisite timestamps while recording the power metrics on a separate thread. Merging these logs and correlating them with inference events (*e.g.*, model initialization, inference start) allowed us to compute the power, energy, and execution time for each inference workload. During testing, we disconnected all peripherals from the device and killed unnecessary background tasks.

Each experiment consisted of running a DNN inference workload under a specific CPU and GPU frequency setting. For our workloads, we used AlexNet [7], ResNet-18 [3], and MobileNet-v2 [12] for image classification, YOLOv4-Tiny [6] for object detection, and DistilBERT [13] and BERT-Tiny [15] for natural language classification. The workload in every experiment was kept constant at 3,200 inferences. Each experiment was repeated 10 times and average values were reported; the variation between runs was low (less than 5%). The workload parameters have been summarized in Table II.

III. Evaluation Results

Frequency Scaling Sweep and DVFS: We start by analyzing the impact of GPU frequency scaling on power and inference time, as shown in Figure 1(a) for an AlexNet inference workload on Jetson Nano for a batch size 16. We see that power consumption increases almost linearly with GPU frequency. However, the *decrease in inference latency starts to plateau out at higher GPU frequencies* as GPU is no longer the bottleneck and the performance is limited by other components such as memory and I/O bandwidth; similar effects have been noted in prior work on servers [16]. Both power and latency vary greatly as GPU frequency changes from lowest to highest



(a) Changing GPU frequency (b) Changing CPU frequency

Fig. 1: Comparison of inference latency and power consumed under different CPU/GPU frequencies when running AlexNet on Jetson Nano.

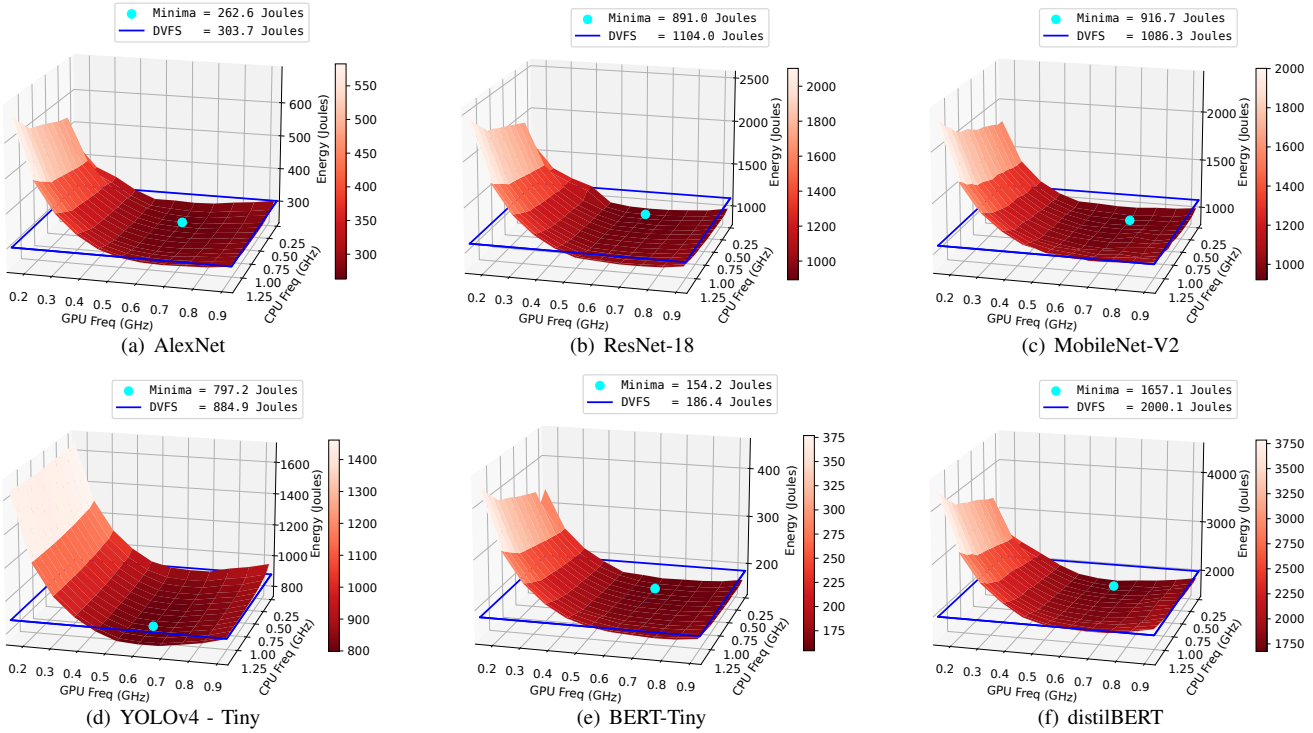
(CPU frequency was fixed at 1132.8 MHz); the span of power consumed is 4.7W, whereas the span of latency is 532s.

Figure 1(b) shows a similar result, but for changing CPU frequencies; here, we fix the GPU frequency at 0.6912 GHz. While the trend is somewhat similar, we see that the span of power (1W) and the span of latency (60s) are much narrower, indicating that ***CPU frequency has a smaller impact compared to GPU frequency***. This is because computationally intensive operations, such as tensor operations, are done by the GPU, whereas the CPU is responsible for less intensive tasks such as data preprocessing, initialization, and control flow.

Next, we evaluate the impact of DVFS on power and latency. The dashed lines in Figure 1 show the behavior when using DVFS (defaults are 'nvhost_podgov' for GPU and 'schedulutil' for CPU) instead of manually setting frequencies. We see that DVFS affords low latency but incurs very high power consumption, suggesting that *default DVFS opts for higher frequencies*. This was confirmed by profiling, which revealed that DVFS operated at the highest CPU and GPU frequencies 89% and 83% of the time, respectively.

We also experimented with the other DVFS governors available (*e.g.*, 'powersave', 'performance', 'ondemand'). We found that 'powersave' consumes <0.01% more energy than the default governors, primarily because although the average power is reduced by 30%, the execution time of the workload is increased by 43%. Likewise, we found that all other DVFS governors perform similarly to the default governor, with the difference in energy consumption being within 1%. With CPU DVFS, the best among other governors had 2.9% lower power but with 2.6% higher inference latency.

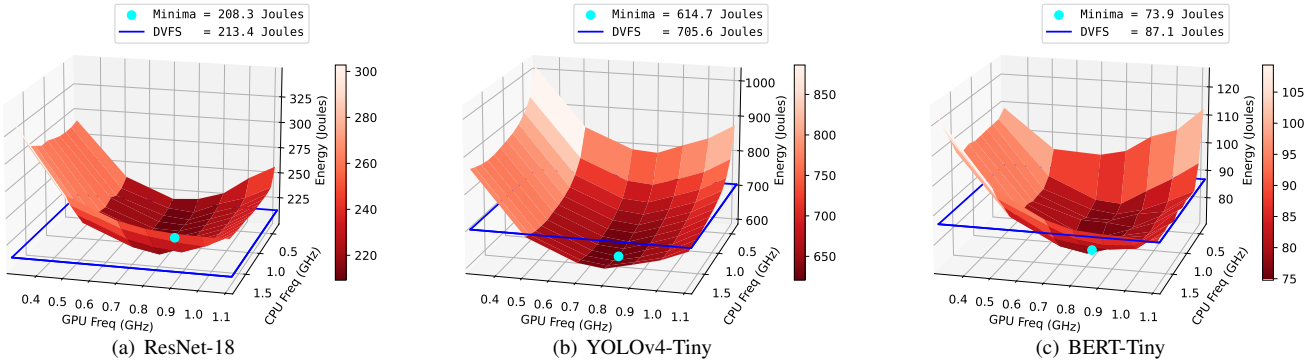
Energy Topology under Jetson Nano: To study the impact of energy, we plot the energy consumed for inference as a function of CPU and GPU frequency for the 6 DNN workloads for Jetson Nano in Figure 2. The batch size is fixed at 16 for all workloads except MobileNet-v2, for which we use a batch size of 8 due to memory constraints. We see that energy does not change much with CPU frequency for a given GPU frequency. However, for a given CPU frequency, the ***energy consumption does change substantially with GPU frequency***. While not always visible, the *impact of GPU frequency on energy is not monotonic*; there exist some moderately high GPU frequencies at which the energy is minimized, as shown by the cyan dot in the plots.



(a) AlexNet (b) ResNet-18 (c) MobileNet-V2

(d) YOLOv4 - Tiny (e) BERT-Tiny (f) distilBERT

Fig. 2: Inference energy consumption as a function of CPU and GPU frequency under Jetson Nano; the color shade denotes the energy consumption, also shown on z-axis. Also highlighted is the point at which energy is minimized. The blue line represents the energy consumed under DVFS.



(a) ResNet-18 (b) YOLOv4-Tiny (c) BERT-Tiny

Fig. 3: Inference energy consumption as a function of CPU and GPU frequency under Xavier NX; the color shade denotes the energy consumption, also shown on the z-axis. Also highlighted is the point at which energy is minimized. The blue line represents the energy consumed under DVFS.

The minimum energy obtained by sweeping over all the CPU and GPU frequencies does lower energy significantly when compared to DVFS, as noted in the legend for each subfigure. The percentage reductions in energy afforded by the minima over DVFS for Figures 2(a)–2(f) are 13.5%, 19.3%, 15.6%, 9.9%, 17.3%, and 17.2%, respectively. However, we found that the latency under the minima configuration is typically 28%–35% higher than that achieved under DVFS. The minima usually occurred at a GPU frequency of 614.4 MHz, with the optimal CPU frequency varying across the workloads.

Energy Topology under Jetson Xavier NX: Figure 3 shows a similar energy vs. frequencies plot for three workloads with batch size 16 but under Xavier NX. We see similar trends here as with the Nano (thus omitting figures for other workloads), with the minima affording an energy reduction of 2.5%, 12.9%, and 15.1% for ResNet-18, YOLOv4-Tiny, and BERT-Tiny, respectively; the reduction afforded for other models was in the 13–15% range. The energy minima usually occur at a GPU frequency of 803.25MHz.

Comparing the energy topologies of Nano (Figure 2) and Xavier NX (Figure 3), we clearly see that the ***energy consumption of Xavier NX is significantly lower***, often by at least 2×, and sometimes as much 4×. Given the newer generation of Xavier NX, this is not wholly unexpected. We also observe a greater increase in energy (compared to the minima) at higher frequencies for Xavier NX, resulting in a steeper trough-like surface; by contrast, the energy values appear to plateau and only slightly increase at higher frequencies under Nano.

REFERENCES

[1] S. Baller, A. Jindal, M. Chadha, and M. Gerndt. Deepedgebench: Benchmarking deep neural networks on edge devices. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*, pages 20–30, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[2] Linux Foundation. Sharpening the edge: Overview of the lf edge taxonomy and framework, Jul 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Stephan Holly, Alexander Wendt, and Martin Lechner. Profiling Energy Consumption of Deep Neural Networks on NVIDIA Jetson Nano. In *Proceedings of the 11th International Green and Sustainable Computing Workshops (IGSC)*, pages 1–6, 2020.

[5] Ziyang Hong and C. Patrick Yue. Efficient-grad: Efficient training deep convolutional neural networks on edge devices with gradient optimizations. 21(2), feb 2022.

[6] Zicong Jiang, Liquan Zhao, Shuaiyang Li, and Yanfei Jia. Real-time object detection method based on improved yolov4-tiny, 2020.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[8] Yu-Jen Ku and Sujit Dey. Sustainable vehicular edge computing using local and solar-powered roadside unit resources. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–7, 2019.

[9] NVIDIA. NVIDIA Jetson Nano System-on-Module. https://developer.download.nvidia.com/assets/embedded/secure/jetson/Nano/docs.

[10] NVIDIA. NVIDIA Jetson Xavier NX System-on-Module. https://en.miivii.com/uploads/file/20200511/NV_Jetson_Xavier_NX_DataSheet_v0.1.pdf.

[11] Sri Pramodh Rachuri, Francesco Bronzino, and Shubham Jain. Decentralized modular architecture for live video analytics at the edge. In *Proceedings of the 3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges*, HotEdgeVideo '21, page 13–18, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[14] Prashanthi S.K, Sai Anuroop Kesanapalli, and Yogesh Simmhan. Characterizing the performance of accelerated jetson edge devices for training deep learning models. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(3), dec 2022.

[15] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models, 2019.

[16] Qiang Wang and Xiaowen Chu. Gpgpu performance estimation with core and memory frequency scaling. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2865–2881, 2020.

[17] Yang Wang, Yubin Qin, Dazheng Deng, Jingchuan Wei, Tianbao Chen, Xinhan Lin, Leibo Liu, Shaojun Wei, and Shouyi Yin. Trainer: An energy-efficient edge-device training processor supporting dynamic weight pruning. *IEEE Journal of Solid-State Circuits*, 57(10):3164–3178, 2022.

[18] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and optimizing GPU energy consumption of DNN training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 119–139, Boston, MA, April 2023. USENIX Association.