The Impact of GPU DVFS on the Energy and Performance of Deep Learning: an Empirical Study

Zhenheng Tang, Yuxin Wang, Qiang Wang, Xiaowen Chu
Department of Computer Science
Hong Kong Baptist University, Hong Kong
{zhtang,yxwang,qiangwang,chxw}@comp.hkbu.edu.hk

ABSTRACT

Over the past years, great progress has been made in improving the computing power of general-purpose graphics processing units (GPGPUs), which facilitates the prosperity of deep neural networks (DNNs) in multiple fields like computer vision and natural language processing. A typical DNN training process repeatedly updates tens of millions of parameters, which not only requires huge computing resources but also consumes significant energy. In order to train DNNs in a more energy-efficient way, we empirically investigate the impact of GPU Dynamic Voltage and Frequency Scaling (DVFS) on the energy consumption and performance of deep learning. Our experiments cover a wide range of GPU architectures, DVFS settings, and DNN configurations. We observe that, compared to the default core frequency settings of three tested GPUs, the optimal core frequency can help conserve 8.7%∼23.1% energy consumption for different DNN training cases. Regarding the inference, the benefits vary from 19.6%∼26.4%. Our findings suggest that GPU DVFS has great potentials to help develop energy efficient DNN training/inference schemes.

CCS CONCEPTS

• **Hardware → Power and energy**.

KEYWORDS

Graphics Processing Units, Dynamic Voltage and Frequency Scaling, Deep Convolutional Neural Network

ACM Reference Format:
Zhenheng Tang, Yuxin Wang, Qiang Wang, Xiaowen Chu. 2019. The Impact of GPU DVFS on the Energy and Performance of Deep Learning: an Empirical Study. In *e-Energy '19: Proceedings of the Tenth ACM International Conference on Future Energy Systems, June 25–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3307772.3328315

1 INTRODUCTION

Recent years witnessed the fast development of deep neural networks (DNN)[20] that can achieve the state-of-art performance in many challenging AI problems, such as image recognition [10, 14, 18, 41, 43], object detection [12, 27, 37] and natural language processing. However, this kind of successful applications heavily rely on the DNN training procedure, which requires a huge number of computational resources. Graphics processing units (GPUs) are currently the most widely used hardware to accelerate the training speed of DNNs. Different from the conventional CPUs, a high-end GPU board includes thousands of cores and a memory module with hundreds of Gigabytes of memory bandwidth.

While most of the previous work addressed the model accuracy and training performance [13, 15, 47], the energy consumption of those high-throughput GPU machines is usually overlooked. Large scale distributed systems [6, 8, 9, 13, 22, 24, 26] are being deployed to speed up the training of complex DNNs, but they also consume a significant amount of electricity. It becomes a critical issue to investigate the trade-off between training performance and energy consumption.

Dynamic voltage and frequency scaling (DVFS) is a widely used technique to balance the performance and power consumption of CPUs. In general, scaling up the CPU voltage/frequency can improve the performance but requires more power supply [2, 17, 30, 46]. Different from CPUs, GPUs have two sets of frequency domains, the core frequency (f^{core}) that controls the speed of ALU cores and other on-chip components, and the memory frequency (f^{mem}) that controls the SDRAM module. Since different GPU applications have different utilization of GPU cores and SDRAM [28, 31], raising the frequency of the components with low utilization may bring no performance improvement but consume higher power. Furthermore, because energy consumption depends on the system power and running time, it is a non-trivial problem to understand how GPU DVFS affects the energy consumption of DNN training.

In this study, we empirically evaluate the performance and energy consumption of DNNs training under different GPU DVFS settings and investigate the impact and energy conservation opportunities of GPU DVFS. Our experiments cover a wide range of GPU architecture generations, GPU DVFS settings, neural network configurations and convolution algorithms. Our major findings are listed as follows:

(1) Scaling up the GPU core frequency can improve the performance of DNN training and inference in varying degrees. Especially for the Turing GTX 2080Ti, the performance of different DNN training can achieve 17.4%∼38.2% improvements by applying a 50% higher core frequency than the default setting, while the performance of inference can be improved by 22.5%∼33.0%.
(2) We observe that the default frequency settings are usually not optimal for energy efficiency. For the Pascal P100 and Volta V100 GPUs, the energy scaling curves with increasing core frequency generally show a valley trend and there

e-Energy '19, June 25–28, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6671-7/19/06…$15.00
https://doi.org/10.1145/3307772.3328315

exists a sweet spot. Compared to the default setting, the optimal core frequencies discovered by our experiments achieve 23.1%, 14.5% and 8.7% average energy conservation for DNN training on three GPUs, respectively. For DNN inference, the average benefits are 26.4%, 22,3%, and 19.6%.

(3) Three convolution algorithms, GEMM, FFT and Winograd, have varying degrees of energy conservation when applying GPU DVFS techniques. Compared to the default setting, the optimal core frequency brings an average 14.5% energy savings for GEMM, 12.6% for FFT and 15.8% for Winograd.

The rest of this paper is organized as follows. Section 2 introduces the background knowledge and related work of DNNs and GPU DVFS. Section 3 describes our experimental design and setup. Section 4 demonstrates our experimental results and discusses the impact of GPU DVFS on the performance and energy consumption of different DNNs. Finally, Section 5 concludes our work and discusses some future research directions.

2 BACKGROUND AND RELATED WORK

2.1 Convolutional Neural Networks

DNNs have been rapidly developed as one of the most popular machine learning algorithms. Specifically, convolutional neural networks (CNNs) have achieved state-of-the-art performance in many AI applications. A typical CNN includes many convolutional layers [14, 18, 41]. Some studies [42] indicated that the computation of convolution layers usually dominate the training time. Besides, GPUs have been acknowledged as one of the most powerful devices to accelerate DNN training, whereas the downside is the huge energy consumption. It is important to develop not only fast but also energy efficient DNN training for GPUs.

There are three popular implementations of the convolution operation. The first approach is transforming the convolution to matrix multiplication [5], which can then benefit from the highly optimized GPU library. The second approach is based on Fourier transform [32], which transforms the convolution operation in the spatial domain to point-wise multiplications in the Fourier domain. The last one is the Winograd algorithm [19], which applies transforms to the input image and kernel to reduce the number of multiplications. NVIDIA's cuDNN library [7] implements all three algorithms. In addition to the exploration of the performance of different convolution algorithms in [25], it is also important to investigate their energy efficiency.

2.2 GPU DVFS

Recently NVIDIA has reinforced their GPUs with the extraordinary computational capability to meet the requirements of DNN training. For example, AutoML techniques often fully utilize hundreds or even thousands of GPUs to search for an efficient DNN structure with several weeks. E.g., Barret et. al [48] adopted 800 GPUs to search for an efficient RNN for language modeling on PTB dataset.

DVFS is one of the most typical energy conservation techniques for traditional CPUs. Some previous GPU DVFS works indicated that GPUs have more complex energy scaling behaviors, and focused on how to balance the performance and energy efficiency of GPUs [1, 2, 11, 17, 21, 30, 36, 44, 45]. Mei et al. [29] and Chau et al. [4] further adopted those DVFS-based energy conservation techniques to implement energy-efficient task scheduling for high-performance clusters. Recent papers [23, 38–40] focused on the performance of scalability of DNN training on different software and hardware environments. Li et al. [23] and Cai et al. [3] started to explore the energy characteristics of DNN processing on GPUs. We believe that it is essential to develop a deeper exploration of the impact of GPU DVFS on deep learning.

3 METHODOLOGY

To conduct a solid exploration of the impact of GPU DVFS on deep learning, we design comparative experiments to cover different facets, including GPU architecture, DVFS setting, the structure of DNNs, and convolution algorithms.

3.1 Hardware Setup

We perform our experiments on a single machine, which is equipped with an Intel i7 920 CPU and 8 GB main memory. We study three different GPUs, of which configurations are listed in Table 3 of the appendix. The default frequency settings are bolded. Due to the limited support offered by the GPU vendor, we can only control the frequencies while the NVIDIA driver will automatically adjust the voltage accordingly. We tune the GPU frequency setting with NVIDIA Inspector [35] and nvidia-smi [34].

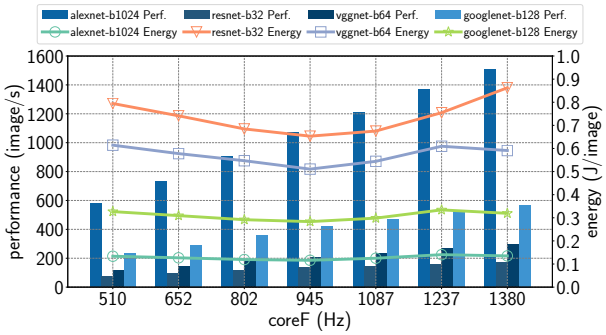
3.2 Network Setup

We explore the impact on both the training and inference procedures of DNN. Caffe [16] and TensorRT[1] are chosen as our training and inference implementations respectively. The CUDA version is 10.0 and the cuDNN version is 7.4.2 for both toolkits. We test four popular DNNs (i.e., AlexNet[18], VggNet-16[41], GoogleNet[43] and ResNet-50[14]), and their setups are listed in Table 4 of the appendix. Different batch sizes are tested for different DNNs according to the GPU memory availability. To explore the impact of DVFS on different convolution algorithms, we revise the Caffe source code to allow fixing the desired convolution algorithm. We test three algorithms, GEMM, FFT and Winograd. They are marked as ipc_gemm, fft_tile, winograd in the figures of Section 4 respectively.

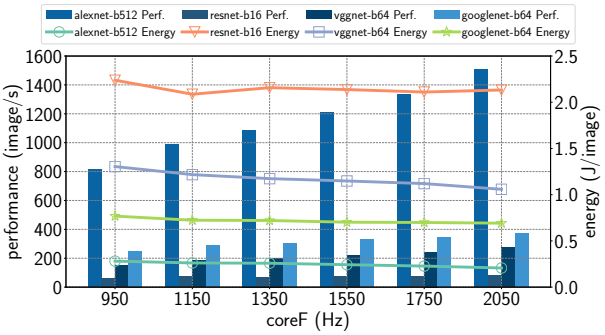
3.3 Performance and Power Measurements

For DNN training, we define the performance, denoted by Per, as the processing images per second. We repeat the experiments for 120 times and record the average time of one training iteration and Per can be obtained by dividing it by the batch size. The performance of inference is similar to training, except that it only records the time of forwarding. We measure the power consumption, denoted by Pow, by the NVIDIA management library (NVML)[33] API. We implement a thread to sample the instantaneous power data during the training/inference procedure and the sampling interval is 2 ms. Since the thread may record those power data sampled before or after GPU execution, we intercept those power data within DNN processing from the sampling results and take the average value. After obtaining all the performance and power data, we describe the energy consumption, denoted by E, with $\frac{Pow}{Per}$, which represents the average energy required by training/inferring a picture.

[1] https://developer.nvidia.com/tensorrt



(a) The performance and energy of V100



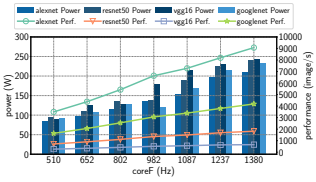
(b) The performance and energy of GTX 2080Ti

Figure 1: The impact of different core frequency settings on performance and energy consumptions of DNNs training.

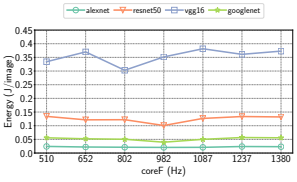
Notice that both core and memory frequency scaling are adopted to GTX 2080Ti. When exploring the effects of core frequency scaling, we calculate the geometric mean value among all the samples of each particular core frequency. The similar treatment is also used to explore the effects of memory frequency scaling and different convolution algorithms.

4 EXPERIMENTAL RESULTS

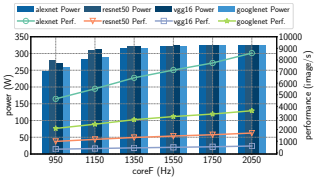
Due to the space limit, we only highlight some significant findings in the experimental results analysis. The complete experimental data can be found in the appendix.



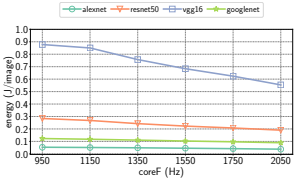
(a) The performance and power of V100



(b) The energy of V100



(c) The performance and power of GTX 2080Ti



(d) The energy of GTX 2080Ti

Figure 2: The impact of different core frequency settings on performance, power and energy consumption of DNN inference.

4.1 Impact of GPU DVFS on Performance and Energy Efficiency

Figure 1 shows the impact of different core frequency settings on the performance and energy consumption of DNN training. Some interesting phenomena are observed. First, scaling up the core frequency generally helps improve the training speed, especially for AlexNet and GoogleNet. The default core frequency of P100 and V100 are also the highest, which reasonably achieve the best performance. For GTX 2080Ti, scaling up the default 1350 MHz to 2050 MHz has 17.4%~38.2% performance improvements for different DNNs. Second, P100 and V100 generally perform a valley trend in the energy scaling curve with increasing core frequency. They achieve a sweet spot of the best energy efficiency in the middle core frequency level, while GTX 2080Ti seems to benefit more from a higher core frequency. The possible reason is that two Tesla GPUs have a dramatically increasing power consumption (refer to appendix) when the core frequency surpasses 1,000 MHz, while GTX 2080Ti does not have this issue.

Different DNNs also demonstrate different performance and energy characteristics. The four DNNs have different numbers of convolution layers. It is reasonable that AlexNet always achieves the best performance and the best energy efficiency, while ResNet-50 has the lowest throughput and needs the largest energy consumption for each image processing. Notice that ResNet-50 shows the best convergence and classification accuracy among four DNNs. The progress of DNNs needs the support of GPU computing energy.

Table 1: Energy Conservation on DNN training/inference by the optimal core frequency: different CNNs

network	DNN training			DNN inference		
	P100	V100	2080Ti	P100	V100	2080Ti
AlexNet	25.7%	7.7%	20.2%	28.7%	17.9%	21.3%
VggNet-16	19.1%	17.9%	9.2%	25.7%	18.9%	26.9%
GoogleNet	24.3%	7.0%	2.3%	28.2%	27.8%	10.9%
ResNet-50	23.1%	25.3%	3.2%	23.1%	24.7%	19.5%

Figure 2 illustrates the impact of different core frequency settings on DNN inference. It is observed that the benefits of GPU DVFS is similar to DNN training. Higher core frequency leads to better image processing throughput. For GTX 2080Ti, scaling up the default 1350 MHz to 2050 MHz has 22.5%~33.0% performance improvements for different DNNs. The energy curves of P100 and V100 achieve the best energy efficiency in the middle frequency zone, while GTX 2080Ti benefits more from a high frequency.

It is also interesting to explore the energy saving by applying the optimal frequency setting compared to the default one. Table 1 concludes the results. Compared to the default setting, the optimal core frequency found in our experiments helps achieve remarkable

energy conservation for DNNs training (23.1% for P100, 14.5% for V100 and 8.7% for GTX 2080Ti on average). For DNNs inference, the average benefits are 26.4%, 22,3%, 19.6% for three GPUs.

4.2 Impact of GPU DVFS on Convolution Algorithms

Figure 3 illustrates the impact of GPU DVFS on the performance and energy consumption of DNNs training when applying different convolution algorithms. For P100 and V100, the performance of three algorithms performs a similar linearly increasing trend with scaling up the core frequency. The energy consumption curves of them also show a valley trend and have a sweet spot on the middle-level core frequency. It can be interpreted by the fact that the power consumption of P100 and V100 have a larger jump when the core frequency surpasses 1,000 MHz. Different from P100 and V100, the performance of ipc_gemm on GTX 2080Ti shows a higher acceleration rate than fft_tile and Winograd.

Table 2: Energy Conservation on DNN training by the optimal core frequency: different convolution algorithms

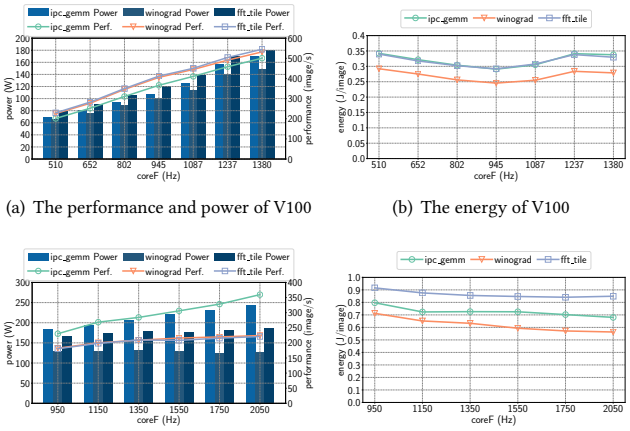
Algorithm	P100	V100	2080Ti
GEMM	23.3%	14.0%	6.3%
FFT	23.1%	11.5%	3.1%
Winograd	25.1%	11.3%	11.0%

Besides, we notice that the power consumption of fft_tile and winograd have negligible changes when adjusting the core frequency. We also have explored the effects of memory frequency scaling and found it not significant. Thus, it is possible that the current implementations of those two convolution algorithms on Turing GPUs still cannot fully utilize the computational resources. Table 2 concludes the energy conservation results of applying the optimal core frequency on three convolution algorithms, compared to the default setting. The average energy conservation is 14.5% for GEMM, 12.6% for FFT and 15.8% for Winograd respectively.

4.3 Discussion

We have investigated the benefits brought by GPU DVFS for the performance improvement and energy conservation of DNN training/inference. First, the performance improvement brought by scaling up the core frequency depends on the GPU core utilization of the software. On the one hand, DNN training includes the data loading step that is not operated on GPUs. Whether the data loading latency can be well hidden significantly affects the GPU core utilization. On the other hand, the GPU kernels of tackling DNN training/inference mainly determine the GPU core utilization. Notice that for GTX 2080Ti, scaling up the default core frequency by 1.5× has 1.17~1.38× performance improvement for DNN training and 1.22~1.33× for DNN inference. Although Caffe, cuDNN and TensorRT have highly optimized implementations for DNN training/inference, the performance gap still exists.

Second, whether GPU DVFS helps conserve the energy consumption of DNN training/inference depends on the changing curves of both performance and power with the increase of the core/memory frequency. For example, as shown in Figure 2(a), the performance of



(a) The performance and power of V100

(b) The energy of V100

(c) The performance and power of GTX 2080Ti

(d) The energy of GTX 2080Ti

Figure 3: The impact of different core frequency settings on performance, power and energy consumption of different convolution algorithms.

alexnet-b1024 on V100 is improved with an approximately equal ratio to the increase of the core frequency (nearly 83% from f^{core}=510 MHz to f^{core}=945 MHz). Meanwhile, as shown in Figure 4(a) in the appendix, the average power of V100 is only increased by 66%. Thus, the energy curve has the lowest value at f^{core}=945 MHz. However, when f^{core} ≥1,000 MHz, the power consumption has a big jump since a higher core voltage is needed to support that frequency range, and then the energy consumption goes up again.

5 CONCLUSION

In this paper, we investigate the impact of GPU DVFS on energy consumption and performance of DNN training and inference. Our experiments cover a wide range of GPU architectures, DVFS settings and CNNs. The results show that the optimal core frequency can not only help improve the DNN performance by up to 33% but also conserve up to 23.1% energy consumption of DNN training and 26.4% of DNN inference. The observations suggest that GPU DVFS has great potentials to help develop energy efficient DNN processing schemes without significant performance degradation.

There are two directions of our future explorations on energy efficient DNN training/inference. First, notice that the same voltage and frequency is applied throughout the feed-forwarding and back-propagation procedures. But different layers may have different energy conservation benefits from different DVFS settings. It is interesting to explore a layer-wise DVFS scheme for DNN training/inference to further reduce energy consumption. Second, considering a scheduling system for multiple DNN training tasks, GPU DVFS can perform as an effective technique to improve the system-wide throughput and decrease energy consumption.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their thorough and insightful comments and suggestions. The research was supported by Hong Kong RGC GRF grant HKBU 12200418.

REFERENCES

[1] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres. 2014. Power and performance characterization and modeling of gpu-accelerated systems. In *IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS), Phoenix, AZ, USA, May, 2014.*

[2] R. A. Bridges, N. Imam, and T. M. Mintz. 2016. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods. *ACM CSUR* 49, 3 (2016), 41.

[3] E. Cai, D. Juan, D. Stamoulis, and D. Marculescu. 2017. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. In *Proceedings of The 9th Asian Conference on Machine Learning (ACML), Seoul, Korea, November, 2017.*

[4] V. Chau, X. Chu, H. Liu, and Y. Leung. 2017. Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy '17), Shatin, Hong Kong, China, May, 2017.*

[5] K. Chellapilla, S. Puri, and P. Simard. 2006. High Performance Convolutional Neural Networks for Document Processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France, October, 2006.*

[6] C. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan. 2018. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI), New Orleans, Louisiana, USA, February, 2018.*

[7] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014).

[8] D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar, B. Kaul, and P. Dubey. 2016. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709* (2016).

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, and Q. V. Le. 2012. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems (NeurIPS), Lake Tahoe, Nevada, United States, December, 2012.*

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, Florida, USA, June 2009,.*

[11] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong. 2013. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *2013 42nd International Conference on Parallel Processing, Lyon, France, October, 2013.*

[12] R. Girshick. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, December, 2015.*

[13] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[14] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June, 2016.*

[15] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, and L. Yu. 2018. Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes. In *NeurIPS Workshop on Systems for ML and Open Source Software, Montréal, Canada, December, 2018.*

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, November, 2014.*

[17] Y. Jiao, H. Lin, P. Balaji, and W. Feng. 2010. Power and performance characterization of computational kernels on the gpu. In *In International Conference on Green Computing and Communications, Hangzhou, China, December, 2010.*

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems (NeurIPS), Miami, Florida, USA, June, 2009.*

[19] A. Lavin and S. Gray. 2016. Fast Algorithms for Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June, 2016.*

[20] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[21] J. Lee, V. Sathisha, M. J. Schulte, K. Compton, and N. Kim. 2011. Improving Throughput of Power-Constrained GPUs Using Dynamic Voltage/Frequency and Core Scaling. In *2011 International Conference on PACT, Galveston, TX, USA, October, 2011.*

[22] S. Lee, J. K. Kim, X. Zheng, Q. Ho, G. A. Gibson, and E. P. Xing. 2014. On Model Parallelization and Scheduling Strategies for Distributed Machine Learning. In *Advances in Neural Information Processing Systems (NeurIPS), Montreal, Quebec, Canada, December, 2014.*

[23] D. Li, X. Chen, M. Becchi, and Z. Zong. 2016. Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs. In *2016 IEEE International Conferences on Sustainable Computing and Communications (SustainCom), Atlanta, GA, USA, October, 2016.*

[24] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October, 2014.*

[25] X. Li, G. Zhang, H. H. Huang, Z. Wang, and W. Zheng. 2016. Performance Analysis of GPU-Based Convolutional Neural Networks. In *2016 45th International Conference on Parallel Processing (ICPP), Philadelphia, PA, USA, August, 2016.*

[26] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *6th International Conference on Learning Representations (ICLR), BC, Canada, April 30 - May 3, 2018.*

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. 2016. Ssd: Single shot multibox detector. In *14th European conference on computer vision (ECCV), Amsterdam, The Netherlands, October, 2016.*

[28] X. Mei and X. Chu. 2017. Dissecting GPU Memory Hierarchy Through Microbenchmarking. *IEEE TPDS* 28, 1 (Jan 2017), 72–86.

[29] X. Mei, X. Chu, H. Liu, Y. Leung, and Z. Li. 2017. Energy efficient real-time task scheduling on CPU-GPU hybrid clusters. In *2017 IEEE Conference on Computer Communications (INFOCOM), Atlanta, GA, USA, May, 2017.*

[30] X. Mei, Q. Wang, and X. Chu. 2017. A survey and measurement study of GPU DVFS on energy conservation. *Digital Communications and Networks* 3, 2 (2017), 89 – 100.

[31] X. Mei, K. Zhao, C. Liu, and X. Chu. 2014. Benchmarking the memory hierarchy of modern GPUs. In *Network and Parallel Computing - 11th IFIP, Ilan, Taiwan, September, 2014.*

[32] M. Michaël, M. Henaff, and Y. LeCun. 2014. Fast training of convolutional networks through ffts. In *2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, April, 2014.*

[33] NVIDIA. 2018. NVIDIA Management Library . [Online] https://developer.nvidia.com/nvidia-management-library-nvml.

[34] NVIDIA. 2018. NVIDIA System Management Interface (nvidia-smi). [Online] https://developer.nvidia.com/nvidia-system-management-interface.

[35] Orbmu2k. 2016. NVIDIA Inspector. [Online] http://blog.orbmu2k.de/tools/nvidia-inspector-tool.

[36] I. Paul, W. Huang, M. Arora, and S. Yalamanchili. 2015. Harmonia: Balancing compute and memory power in high-performance GPUs. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, OR, USA, June, 2015.*

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June, 2016.*

[38] S. Shams, R. Platania, K. Lee, and S. Park. 2017. Evaluation of Deep Learning Frameworks Over Different HPC Architectures. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, June, 2017.*

[39] S. Shi, Q. Wang, and X. Chu. 2018. Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs. In *4th Intl Conf on Big Data Intelligence and Computing(DataCom), Athens, Greece, August , 2018.*

[40] S. Shi, Q. Wang, P. Xu, and X. Chu. 2016. Benchmarking State-of-the-Art Deep Learning Software Tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China, November, 2016.*

[41] K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, May, 2015,.*

[42] V. Sze, Y. Chen, T. Yang, and J. S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* 105, 12 (Dec 2017), 2295–2329.

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going Deeper With Convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, June, 2015.*

[44] Q. Wang and X. Chu. 2017. GPGPU Power Estimation with Core and Memory Frequency Scaling. *SIGMETRICS Perform. Eval. Rev.* 45, 2 (Oct. 2017), 73–78.

[45] Q. Wang and X. Chu. 2018. GPGPU Performance Estimation with Core and Memory Frequency Scaling. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, December, 2018.*

[46] Q. Wang, P. Xu, Y. Zhang, and X. Chu. 2017. EPPMiner: An Extended Benchmark Suite for Energy, Power and Performance Characterization of Heterogeneous Architecture. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy'17), Shatin, Hong Kong, China, May, 2017.*

[47] Y. You, Z. Zhang, C. Hsieh, J. Demmel, and K. Keutzer. 2018. ImageNet Training in Minutes. In *Proceedings of the 47th International Conference on Parallel Processing (ICPP), Eugene, OR, USA, August, 2018.*

[48] B. Zoph and Q. V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations (ICLR), Toulon, France, April , 2017.*

A APPENDIX

In the appendix, we present the complete experimental results of four DNNs on three GPU cards. The concrete hardware configurations of three GPUs are listed in Table 3. The experiments cover a wide range of frequency options. Due to the limited support offered by the GPU vendor, we can only control the frequencies while the NVIDIA driver will automatically adjust the voltage accordingly.

Table 3: Target GPU specifications

Device	Tesla P100	Tesla V100	GTX 2080Ti
Architecture	Pascal	Volta	Turing
SMs/SM Cores	16/128	28/128	72/64
Global mem.	16 GB	16 GB	12 GB
Core freq. (MHz)	[544, 683, 810, 936, 1,063, 1,202, **1,328**]	[510, 652, 802, 945, 1,087, 1,237, **1,380**]	[950, 1,150, **1,350**, 1,550, 1,750, 1,950]
Memory freq. (MHz)	715	877	[5,800, 6,300, **6,800**, 7,300]

We test four popular DNNs (i.e., AlexNet[18], VggNet-16[41], GoogleNet[43] and ResNet-50[14]), and their setups are listed in Table 4. Different batch sizes are tested for different DNNs according to the GPU memory availability.

Table 4: The experimental setup of neural networks

Network	# of layers	Parameters	Batch size
AlexNet	8	~60 millions	128/256/512/1024
VggNet-16	16	~138 millions	16/32/64
GoogleNet	22	~53 millions	16/32/64/128
ResNet-50	50	~24 millions	8/16/32

We demonstrate the experimental results grouped by three different convolution algorithms: GEMM, Winograd, and FFT. We present how the performance and power change with respect to the GPU core and memory frequencies on different DNNs. Each figure includes the results of different batch sizes. Due to the limitation of GPU memory size, some large batch sizes are not supported. Notice that GTX2080Ti supports both core and memory frequency scaling. To give a comprehensive result of core frequency scaling, we calculate the geometric mean of the data of each core frequency across all memory frequency sets. For memory frequency scaling, we calculate the geometric mean of the data of each memory frequency across all memory frequency sets.

A.1 Using implicit GEMM algorithm

Figures 4 and 5 demonstrate the results of GEMM algorithm on two Tesla GPUs, P100 and V100. When f^{core} is less than 1,000 MHz, the performance mostly has a faster-growing trend than the power consumption with the increase of the core frequency. However, when f^{core} surpasses 1,000 MHz, the power consumption has a sudden jump, which raises up the total energy consumption again. Thus, the core frequency that achieves the best energy efficiency usually lies in the middle interval. Besides, it is observed that the performance of AlexNet and VggNet-16 rarely changes with different batch sizes, while GoogleNet and ResNet-50 gain higher image processing throughputs with larger batch sizes. Notice that GoogleNet and ResNet-50 have more layers than the other two. Larger batch sizes help GoogleNet and ResNet-50 achieve higher GPU utilization.

Figures 6 and 7 demonstrate the results of GEMM algorithm on GTX 2080Ti. Different from two Tesla GPUs, the performance of GTX 2080Ti always has a faster-growing trend than the power consumption, which results in that the best energy efficiency is mostly achieved at the highest core frequency. On the contrary, increasing the memory frequency hardly affects the performance of DNN training, but leads to higher power consumption. Thus, applying a low memory frequency surprisingly helps conserve energy.

A.2 Using Winograd algorithm

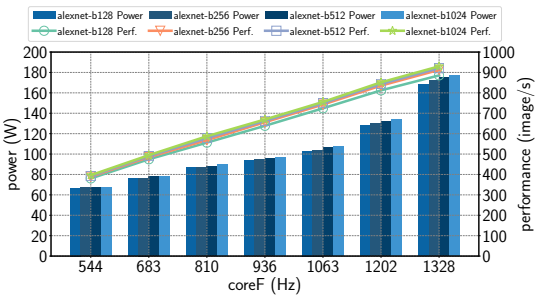
Figures 8 and 9 demonstrate the results of Winograd algorithm on two Tesla GPUs, P100 and V100. Notice that Winograd requires a larger GPU memory to tackle convolution than GEMM does. Only a few batch sizes of four DNNs are supported on GPUs. Similar to GEMM, the core frequency that achieves the best energy efficiency usually lies in the middle interval. Besides, GoogleNet and ResNet-50 achieve higher image processing throughputs with larger batch sizes. Compared to GEMM, Winograd achieves better performance while keeping nearly the same power consumption. Thus, Winograd has better energy efficiency than GEMM, which also meets the results in Figure 3.

Figures 10 and 11 demonstrate the results of Winograd algorithm on GTX 2080Ti. Scaling up the core frequency leads to different performance improvement for different DNNs and even different batch sizes. The power consumption of AlexNet remains nearly the same when applying different batch sizes, while that of GoogleNet and ResNet-50 becomes larger with the increase of the batch size. Similar to GEMM, increasing the memory frequency hardly helps conserve energy since the performance cannot benefit from it.

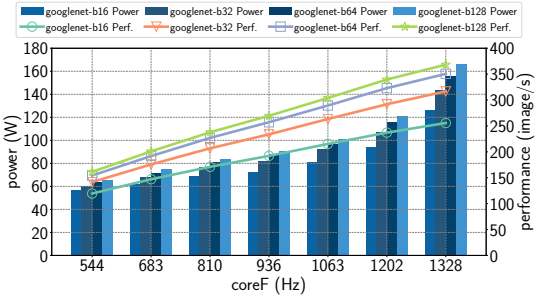
A.3 Using FFT algorithm

Figures 12 and 13 demonstrate the results of FFT algorithm on two Tesla GPUs, P100 and V100. FFT requires the largest GPU memory to tackle convolution among three algorithms. The tested GPUs can only process small batch sizes when applying FFT for DNN training. Similar to the previous two algorithms, the performance grows faster than the power for most cases with the increase of the core frequency. GoogleNet also achieves a higher image processing throughput with a larger batch size.

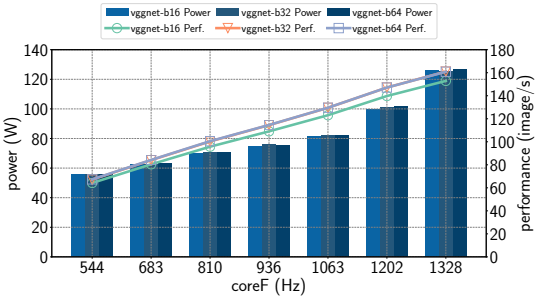
Figures 14 and 15 demonstrate the results of FFT algorithm on GTX 2080Ti. Unfortunately, no matter for the performance and the power consumption, scaling up both the core and memory frequencies brings few benefits. It seems that the current implementation of FFT-based convolution on Turing GPUs still cannot fully utilize the computational resources. Besides, FFT generally works slower than the other two algorithms when the convolutional kernel size is small. Since the convolution layers of four tested DNNs features small kernel sizes, it is difficult for FFT to beat GEMM and Winograd in DNN applications.



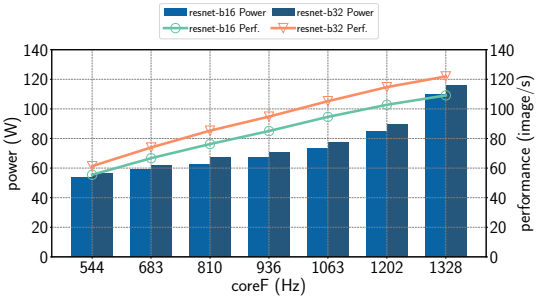
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

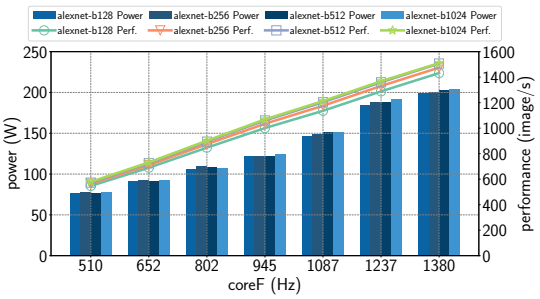


(c) power and performance of training VggNet-16

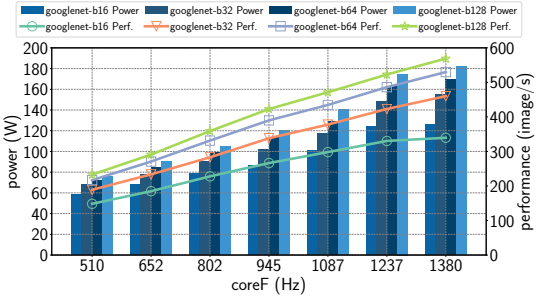


(d) power and performance of training ResNet-50

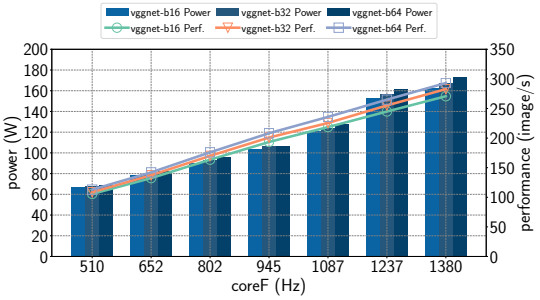
Figure 4: training using implicit GEMM on P100 with increase of core frequency



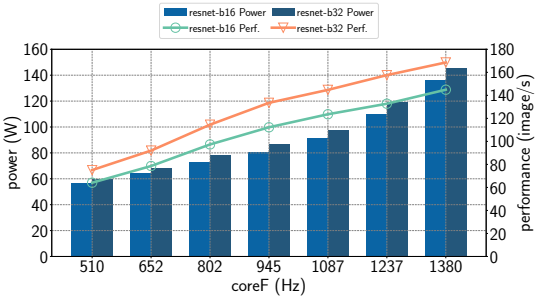
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

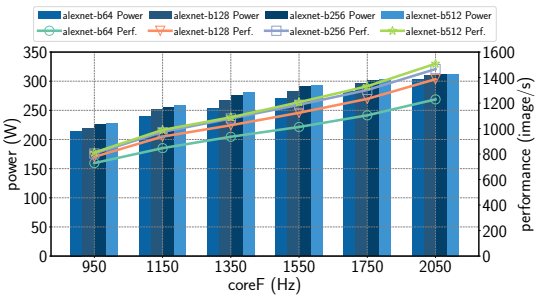


(c) power and performance of training VggNet-16

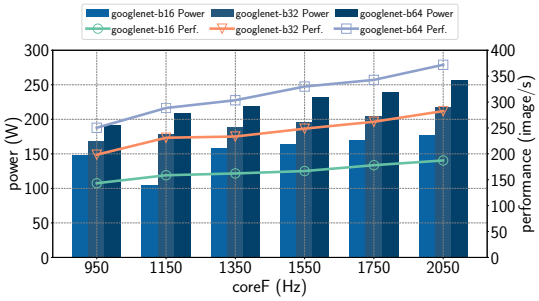


(d) power and performance of training ResNet-50

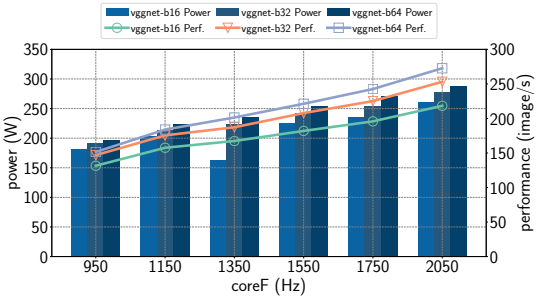
Figure 5: training using implicit GEMM on V100 with increase of core frequency



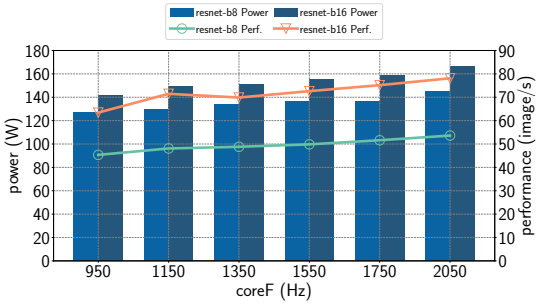
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

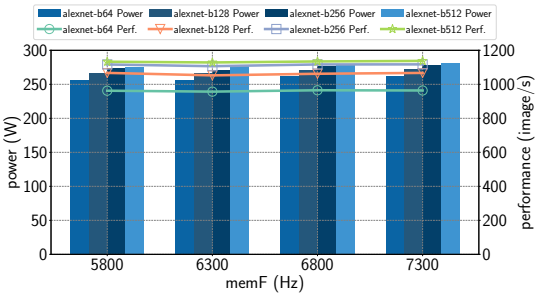


(c) power and performance of training VggNet-16

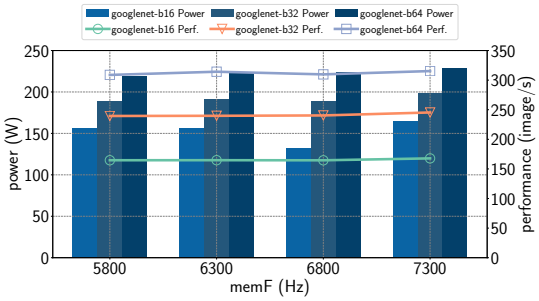


(d) power and performance of training ResNet-50

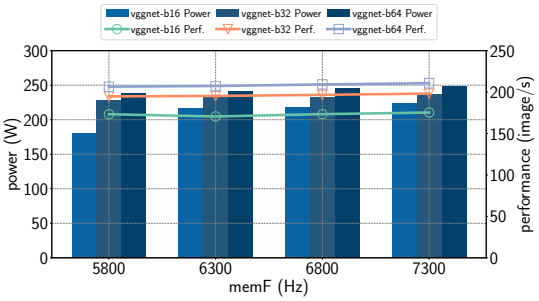
Figure 6: training using implicit GEMM on GTX2080Ti with increase of core frequency



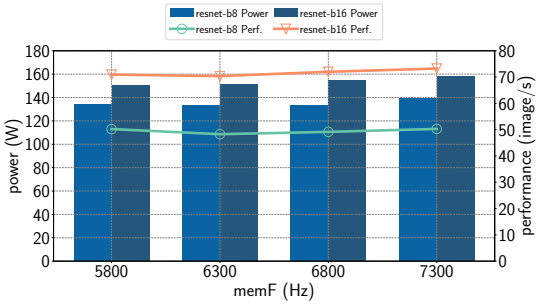
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

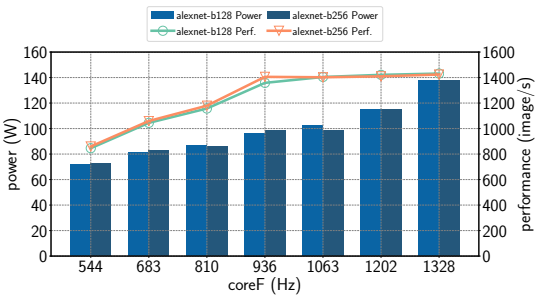


(c) power and performance of training VggNet-16

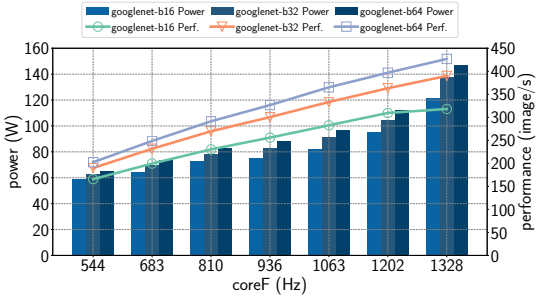


(d) power and performance of training ResNet-50

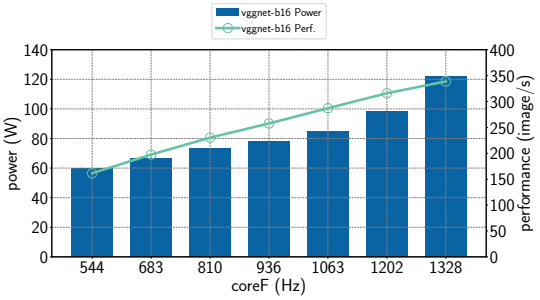
Figure 7: training using implicit GEMM on GTX2080Ti with increase of memory frequency



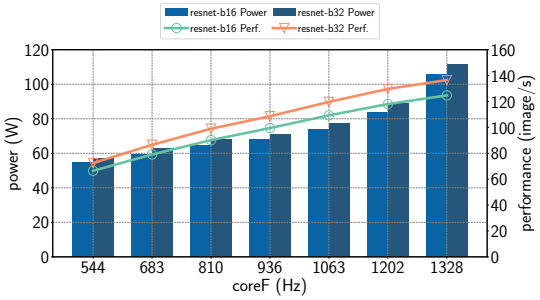
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

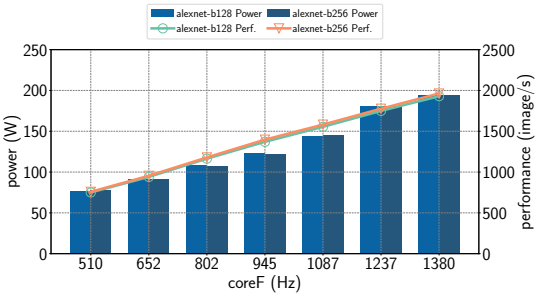


(c) power and performance of training VggNet-16

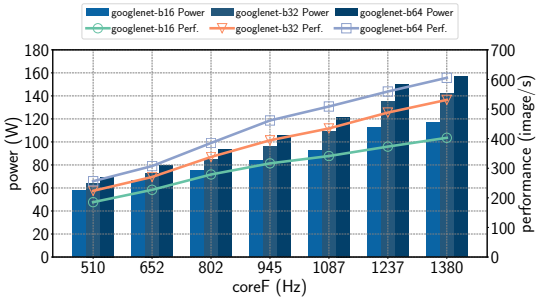


(d) power and performance of training ResNet-50

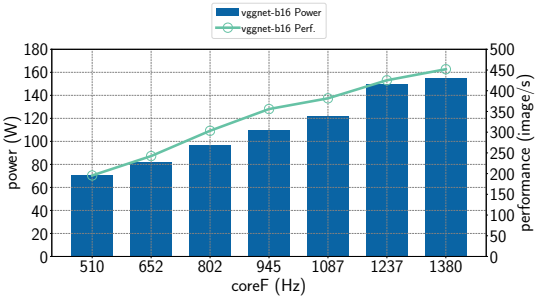
Figure 8: training using Winograd on P100 with increase of core frequency



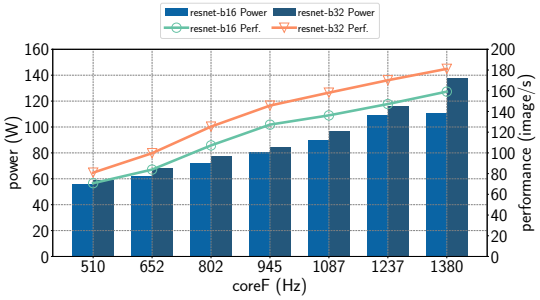
(a) power and performance of training AlexNet



(b) power and performance of training GoogleNet

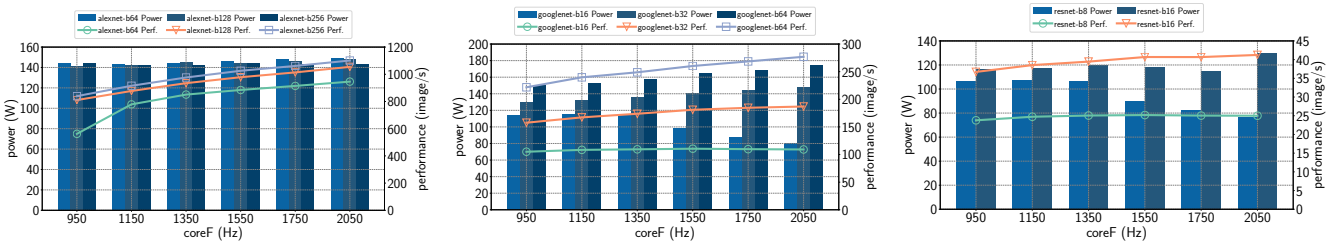


(c) power and performance of training VggNet-16



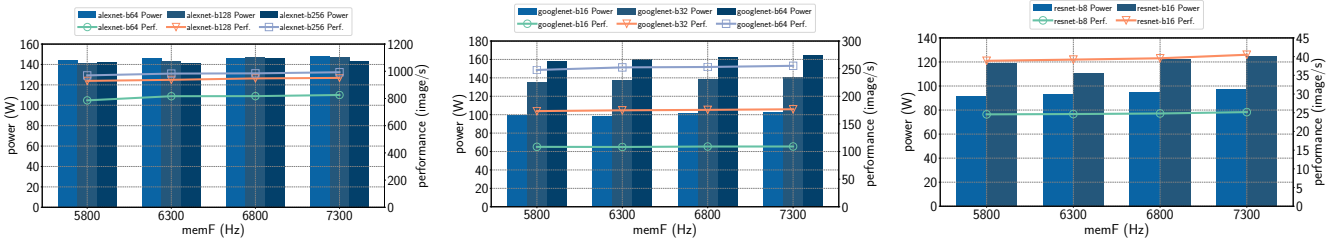
(d) power and performance of training ResNet-50

Figure 9: training using Winograd on V100 with increase of core frequency



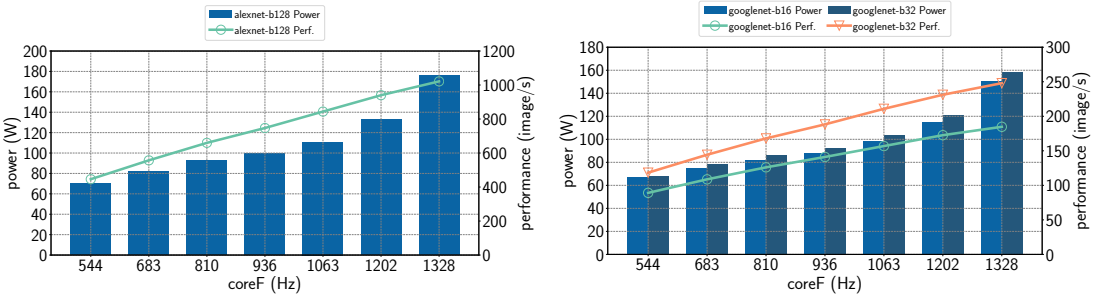
(a) power and performance of training AlexNet (b) power and performance of training GoogleNet (c) power and performance of training ResNet-50

Figure 10: training using Winograd on GTX2080Ti with increase of core frequency



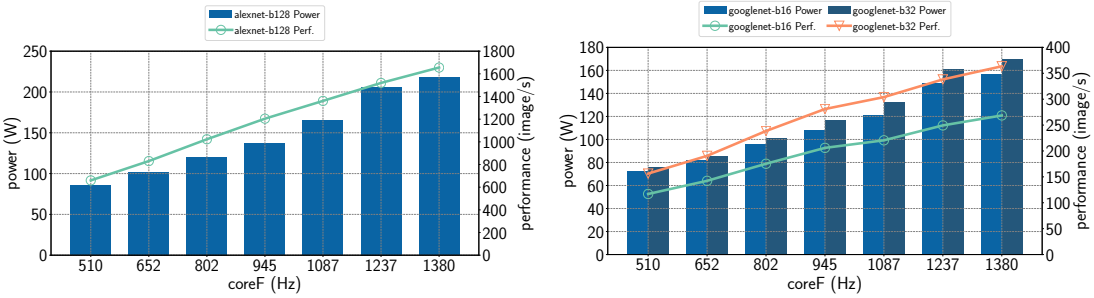
(a) power and performance of training AlexNet (b) power and performance of training GoogleNet (c) power and performance of training ResNet-50

Figure 11: training using Winograd on GTX2080Ti with increase of memory frequency



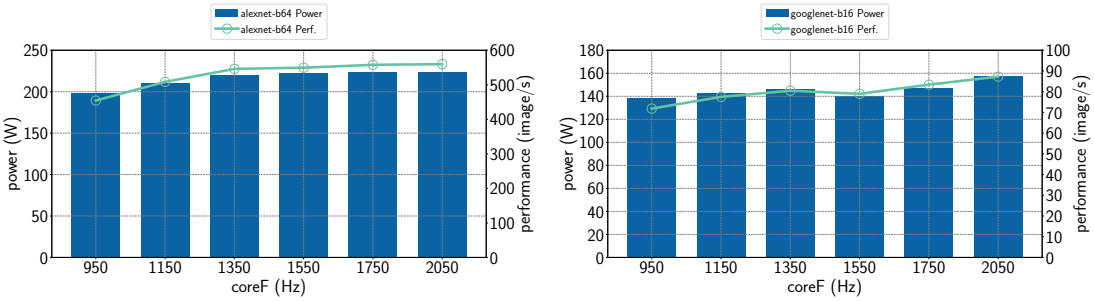
(a) power and performance of training AlexNet (b) power and performance of training GoogleNet

Figure 12: training using FFT on P100 with increase of core frequency



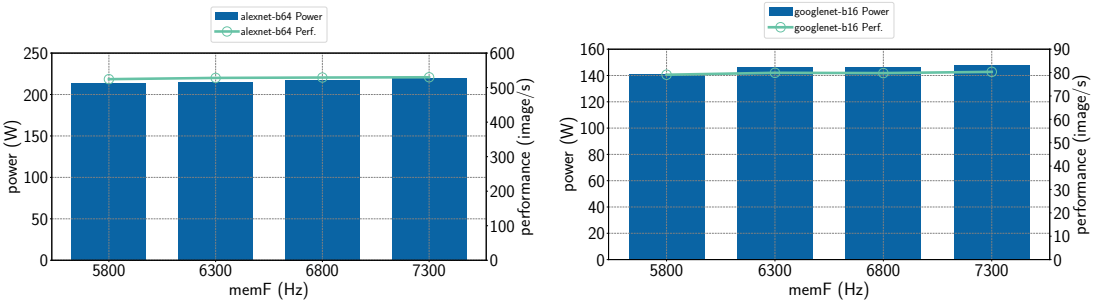
(a) power and performance of training AlexNet (b) power and performance of training GoogleNet

Figure 13: training using FFT on V100 with increase of core frequency



(a) power and performance of training AlexNet (b) power and performance of training GoogleNet

Figure 14: training using FFT on GTX2080Ti with increase of core frequency



(a) power and performance of training AlexNet (b) power and performance of training GoogleNet

Figure 15: training using FFT on GTX2080Ti with increase of memory frequency