Exploiting non-conventional DVFS on GPUs: application to Deep Learning

Francisco Mendes
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Pedro Tomás
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Nuno Roma
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—The use of Graphics Processing Units (GPUs) to accelerate Deep Neural Networks (DNNs) training and inference is already widely adopted, allowing for a significant increase in the performance of these applications. However, this increase in performance comes at the cost of a consequent increase in energy consumption. While several solutions have been proposed to perform Voltage-Frequency (V-F) scaling on GPUs, these are still one-dimensional, by simply adjusting frequency while relying on default voltage settings. To overcome this, this paper introduces a methodology to fully characterize the impact of non-conventional Dynamic Voltage and Frequency Scaling (DVFS) in GPUs. The proposed approach was applied to an AMD Vega 10 Frontier Edition GPU. When applying this non-conventional DVFS scheme to DNNs, the obtained results show that it is possible to safely decrease the GPU voltage, allowing for a significant reduction of the energy consumption (up to 38%) and the Energy-Delay Product (EDP) (up to 41%) on the training of CNN models, with no degradation of the networks accuracy.**

***Index Terms*—GPU, DVFS, Undervoltage**

I. Introduction

In the last few years, Deep Neural Networks (DNNs) have had a significant impact in industry and society, by allowing for important breakthroughs in many application domains, such as computer vision, speech recognition, natural language processing, drug discovery, genomics, etc [1].

However, DNNs are usually characterized by significant computational burdens, particularly when considering the training of very deep and complex networks, and/or when dealing with high dimensional data, such as images and videos. For such purpose, researchers (and data scientists, in general) often rely on accelerators, such as Graphics Processing Units (GPUs), to cope with the associated computational burden and reduce the training time. As a result, GPUs are now commonly deployed on most supercomputers, data centers and other computational infrastructures related with the development of artificial intelligence algorithms.

Additionally, several software frameworks, algorithms and techniques have been proposed to manage and optimize the execution of DNNs on GPUs (e.g., Mittal [2]). However, most optimization techniques neglect the energy impact of the training phase, usually resulting in considerable costs.

To overcome this problem, researchers have also explored other solutions that allow mitigating the energy impact of neural network training. One particular and common approach relies on the use of low-precision arithmetic (e.g., Nabavinejad [3]), eventually trading network accuracy with increased processing performance and lower energy consumption.

Researchers have also looked at alternative approaches, such as by exploiting Dynamic Voltage and Frequency Scaling (DVFS) on both the inference and training phases. In fact, by carefully selecting the used voltage-frequency (V-F) levels, significant energy savings can be obtained, although depending on the considered DNN architecture and computing device [4]. This is achieved through a careful balance between the performance and power consumption of the different GPU components (particularly the core and global memory) such as to minimize stalls in the compute cores. In fact, not only can DVFS be used to decrease the power consumption, but it can also boost the system performance [4], by increasing the voltage and frequency levels (as long as the GPU total power envelope and thermal limits are not surpassed).

Nevertheless, most state-of-the-art works only consider tightly coupled V-F levels, often predefined by GPU manufacturers and neglecting the voltage margin that is usually introduced to guarantee fail-safe designs, as well as its variation with the kernel instruction sequence and the corresponding use of specific GPU components. Supported on this observation, this work starts by investigating such margins, by relying on a set of carefully crafted micro-benchmarks. Then we apply V-F scaling for the training and inference of state-of-the-art networks and conclude that significant energy and EDP savings can be attained by working at near-threshold voltage.

In accordance, the main contributions of this work are:

- Proposal of a new methodology and synthetic benchmark suit to fully characterize the impact of V-F scaling on modern GPU architectures;
- When applied to an AMD Vega 10 Frontier Edition, we show that it can be safely undervoltage (from the default setup), with the DRAM-Cache controller and the Arithmetic and Logic Unit (ALU) being the most sensitive components to voltage drops.
- Demonstration that non-conventional DVFS results in significant energy savings, and also on performance gains, as a side effect of the observed power savings.
- Evaluation of the computation errors due to undervoltage, showing that it can be safely applied to both the training and inference of DNNs without compromising the networks accuracy.

II. Related Work

Several authors have already exploited DVFS to reduce the GPU power consumption and attain energy savings. Different

approaches have been used, most commonly by relying on performance, power or energy consumption models (e.g., Guerreiro [5]–[7], Wang [8] and Fan [9], [10]). However, other alternative approaches have also been studied, such as standard machine learning techniques to predict when and how to adopt frequency scaling (e.g., the work of Guerreiro [11]).

Nevertheless, most of these solutions rely on tightly coupled voltage-frequency (V-F) levels, where the voltage level is predefined by the GPU manufacturer and it is based on the device working operating frequency, with an extra voltage guardband to take into account process and aging variations.

Nonetheless, some other works have already considered the investigation on how to minimize the voltage guardband to improve energy efficiency. Kalogirou et al. [12] explore such a concept to reduce the CPU consumption on cloud data centers. Papadimitriou et al. [13] comprehensively characterized the voltage guardbands in different ARM processors to predict the minimum working voltage using performance counters. Nakhaee et al., [14] exploit the properties of error resilient applications to operate CPUs with negative guardbands, i.e. with timing violations that introduce insignificant errors in the application results. However, although unveiling important results, these works are not directly applicable to GPUs.

On the other hand, Leng et al. [15] studied the undervoltage effect in NVIDIA GPUs to conclude that there exists a significant voltage guardband that is dependent on application kernels and that can result in up to 25% energy savings if reduced to a minimum. Similarly, Thomas et al. [16] studied the joint effect of process variations and voltage noise on GPU architectures, and developed a solution to dynamically reduce the voltage margins, achieving 15% energy savings. Similarly, Tan et al. [17] investigated the impact of reducing the voltage guardband at the register file and developed a solution to make the computation viable with unreliable register files in a low voltage operation. However, these works did not consider the impact of DVFS, as it will be addressed in this paper.

In contrast, a more ambitious exploitation of voltage and frequency scaling is envisaged in this paper. Contrarily to a strict application of conventional DVFS techniques, the research that is herein presented considers a complete detachment of the voltage and frequency setups, in order to identify the most energy efficient operation in each application. To attain this objective, a comprehensive characterization of the several GPU components will be undertaken (namely the memories and execution units), by defining a convenient set of benchmarks that will allow an individual characterization of each component. The gathered information will then support the definition of non-conventional DVFS mechanisms that will allow a wholly decoupled scaling of the GPU voltage and frequency. A final case study will be presented, by applying these contributions to the optimization of a set of DNNs.

III. Architecture Characterization with Independent Voltage and Frequency Scaling

To evaluate and characterize the GPU architecture when subjected to non-conventional DVFS, a set of kernels were devised to evaluate the impact undervolting on the different GPU components. This allows determining the frequency-dependent minimum operating voltage that (still) leads to correct GPU operation (V_{min}) and to understand the impact of independent voltage scaling on performance and energy consumption. The kernels, presented in Table I, are described next, and are available as open-source[1].

TABLE I
Devised set of kernels to characterize GPU to Non-Conventional DVFS

Micro-kernels	Data Type	Objective
DRAM	FP32, INT32	Minimum Read & Write voltage, bit-flip, data-corruption
Cache L2	INT32	Minimum Read & Write voltage, data-corruption
Shared Memory	INT32	Minimum Read & Write voltage, data-corruption
ALU	FP64/32/16, INT64/32/16/8	Computation errors due to timing violations
SFU	FP64/32/16	Computation errors due to timing violations
Branch		Minimum voltage for correct schedualing operation
Mix (reduction)	FP64/32/16	Evaluates the simultaneous impact of stressing multiple GPU components

A. Characterization benchmarks

The devised benchmarks individually characterize the different components of the two GPU DVFS domains (*core* and *global memory*): DRAM, Shared Memory, Cache L2, and ALU. The ALU experiments cover both Multiply and Accumulate (MAC) and non-linear operations, as well as the impact of branches. Every benchmark was tested for multiple data types, by replacing a `DATA_TYPE` placeholder with standard integer, single and double precision types. However, due to space limitations, only the results that reveal a greater sensitivity to V_{min} will be herein reported, corresponding to DRAM, Cache and MAC.

Finally, it will be also evaluated a coarser and more representative kernel in many GPGPU applications - the reduction - which simultaneously stresses multiple architecture elements.

1) DRAM: This benchmark was devised (and validated through GPU counters) to determine the impact of memory under-clocking, by continuously fetching data from the DRAM (see Listing 1). For each data fetch, `OPS` controls the number of arithmetic operations to be performed before the data is placed on the DRAM again. A lower `OPS` value results in a more memory intensive kernel, eventually leading to a memory bound kernel. In contrast, a higher `OPS` results in a less memory intensive kernel and, since the memory accesses become more spaced in time, eventually results in a compute bounded kernel.

2) Cache: The *core* DVFS domain controls both the global L2 cache and the local L1 caches. The devised benchmark (see Listing 2) is very similar to Listing 1. However, it acts on the L2 cache and on the state machine responsible for communicating with the DRAM. The number of issued requests to the cache and to the DRAM-cache controller is the same independently of the `OPS` value. However, the amount of

[1]https://github.com/hpc-ulisboa/nonconventional-dvfs

```
void  DRAMcode(DATA_TYPE *IN0,*IN1,*OUT) {
    const int ite = (blockIdx.x * THREADS +
                threadIdx.x) % MEM_BLOCK;
    volatile DATA_TYPE r0;

    for (int i = 0; i < N; i++) {
        r0= IN0[i * C + ite] + IN1[i * C + ite];
        #pragma unroll
        for(int j = 0; j < OPS; j++)
            r0 += r0 * r0;
        OUT[threadId] = r0;
    }
}
```

Listing 1. DRAM Benchmark Code

```
void  CacheL2code(DATA_TYPE *IN, *OUT) {
    const int ite = blockIdx * THREADS + threadIdx;
    volatile DATA_TYPE r0;

    for (k=0; k<N; k++)
        for(j=0; j<COMP_ITE; j++) {
            r0= IN[ite];
            #pragma unroll
            for(m=0; m<OPS; m++)
                r0 += r0;
            OUT[ite] = r0;
        }
}
```

Listing 2. CacheL2 Benchmark Code

time between requests changes with `OPS`. As in the previous case, the benchmark was validated through GPU counters.

3) MAC: Listing 3 presents the devised benchmark to stress the ALU. A greater emphasis was devoted to the MAC operation, due to its prevalence in the Deep Learning (DL) domain. It is expected that some computational errors may occur when overly undervoltage is applied to this component, due to timing violations across the critical path. Another factor under test is the influence of dependencies in the code, as these may influence how the warps scheduler orders the threads for execution on the CUs. Since DL workloads are usually characterized by massive levels of parallelism, which translates to a high number of warps per block, the benchmark was devised to mimic this situation. The benchmark can be also used to study the influence of different dependencies that can exist in the application, by assigning a value between 0 and 5 to variable `d`. When `d=0`, no dependencies exist in the code. The setup with `d=1` represents the worst-case scenario, since introduces Read-after-Write (RaW) dependencies between all operations. This particular dependency setup was emphasized in the presented study, due to the variability of kernels executed by DL workloads. On the other hand, the setup with `d=3` was considered a general case, where some dependencies still exist in the code, but the scheduler can mask some of them.

4) Reduction: The `reduction` benchmark (Listing 4) reduces the length of a N-sized vector to $N/blockDim$, by performing an element wise sum. It makes use of the shared memory to enable inter-thread communication and improve performance. Hence, this benchmark stresses all elements of the architecture (DRAM, Cache, shared memory and ALU) and allows to assess a more complex use-case, where a single kernel stresses multiple architectural units.

```
void  ALUcode(DATA_TYPE *IN, *OUT) {
    const int ite = (blockIdx*THREADS+threadIdx)*4;

    volatile DATA_TYPE r0, r1, r2, r3, r4, r5;
    r0=IN[ite];  r1=IN[ite+1];  r2=IN[ite+2];
    r3=IN[ite+3];  r4=IN[ite];  r5=IN[ite+1];

    for(j=0; j<COMP_ITE; j++) {
        r0 += r0 * r{0-d}; r1 += r1 * r{1-d};
        r2 += r2 * r{2-d}; r3 += r3 * r{3-d};
        r4 += r4 * r{4-d}; r5 += r5 * r{5-d};
    }
    OUT[ite/4] = r0;
}
```

Listing 3. ALU Benchmark Code

```
void Reduction(DATA_TYPE* idata, T* odata){
    __shared__ DATA_TYPE s[THREADS];
    unsigned int i, k, t = threadIdx;
    unsigned int index = blockIdx*blockDim*N
                        + threadIdx;

    // cooperative load from global to shared memory
    s[t] = 0;
    for (i=0; i< 4; i++, index += blockDim.x)
        s[t] += idata[index];
    __syncthreads();

    // do reduction in shared memory
    if(t < 64) {
        s[t] += s[t+64]; __syncthreads(); }

    if(tid <32){
        s[t] += s[t+32];   s[t] += s[t+16];
        s[t] += s[t+8];    s[t] += s[t+4];
        s[t] += s[t+2];    s[t] += s[t+1];
    }

    // write result for this block to global mem
    if(t == 0) odata[blockIdx.x] = s[0];
}
```

Listing 4. Reduction Kernel Code

B. Non-Conventional DVFS Experimental Setup

The devised benchmarks were applied to characterize an AMD Vega 10 Frontier Edition GPU, whose specifications are presented in Table II. Despite some preconfigured Frequency-Voltage (F-V) setups, the GPU vendor rocm-smi[2] tool allows for an independent control over frequency and voltage. To allow a greater evaluation range, the GPU power cap was changed from the default 220W to 300W (matching the GPU thermal design). This GPU was installed on a machine equipped with an Intel i7 4770K CPU, with 32 GB of main memory.

The default frequencies of the GPU *Core* and *DRAM* domains, presented in Table II, were selected as the starting point for the non-conventional DVFS. For each frequency, the devised experiment started at the maximum voltage ($1200mV$) and a gradual undervoltage of the GPU V-F domain under test was applied with $50mV$ steps. For each step, the benchmarks were executed ten times to obtain the median value of the execution time and energy consumption.

[2]github.com/RadeonOpenCompute/ROC-smi

TABLE II
AMD VEGA 10 FRONTIER EDITION SPECIFICATIONS.

Architecture	GNC5
CUs & DRAM size	64 & 16 GB
Core voltage range [mV]	[900 - 1200]
DRAM voltage range [mV]	[800 - 1200]
Default Frequency-Voltage (F-V) setups	
Core F-V [MHz ; mV]	[995;900, 1140;950, 1350;1050, 1440;1100, 1530;1150, 1600;1200]
DRAM F-V [MHz ; mV]	[500;900, 800;950, 950;1000]

To avoid any bias incurred by the considered data values, all the tests were performed using randomly generated inputs. Integer values were obtained from a normal distribution across their complete 32-bits range. Floating-point operands were generated using an uniform distribution in the interval $[0.1\ ;\ 1]$. This ensures that operations are never applied to numbers with a significant different exponent value, thus avoiding rounding errors that would conduct to the discard of the operator with the lowest absolute value.

While performing the undervoltage, the GPU goes through three distinct stages. At the first stage (*working*), the GPU works regularly and no changes are detected in the application output. Then, by continuing the reduction of the GPU voltage, some *computational errors* are introduced and some application outputs change when compared with the default voltage setup. By continuing reducing the GPU voltage beyond this stage, the GPU enters into the *crash* state, becoming unusable.

To accurately determine the areas of interest (i.e., when infrequent computation errors occur) and to determine the crash point, the undervoltage step was reduced to $10mV$. Furthermore, when dealing with the *DRAM* V-F domain, the *Core* V-F domain was set to default values; for the *Core* V-F domain, the highest frequency and default voltage of the *DRAM* was selected. The GPU power consumption was measured using gpowerSAMPLER[3] [5], at every millisecond.

C. Characterization Results

For an easier understanding of the obtained results, the following charts only represent the data-points that correspond to voltages equal-to and lower-than the default voltage of each frequency level (no interesting data is found at higher voltage levels). Furthermore, the performance, energy consumption, and energy-delay product charts were normalized to the results achieved with the highest core frequency and default voltage (1600MHz and 1200mV), so a smaller number indicates an improvement in relation to that configuration.

1) DRAM: Fig. 1 illustrates the usable voltage range of the DRAM domain and the normalized performance and energy consumption when varying the `OPS` parameter between 0 and 50 operations (see Listings 1). The conducted experiment shows that no computation error or crashes happen for the default frequencies within the complete voltage range. The kernel runs successfully, with no perceptible change in the

[3]github.com/hpc-ulisboa/gpowerSAMPLER

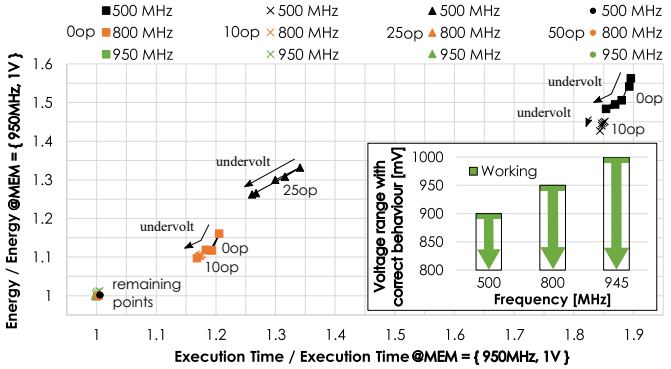


Fig. 1. DRAM DVFS - Normalized energy consumption and execution time and usable DRAM voltage for each frequency configuration.

output. The experiment also shows that for all `OPS` values (0 to 50), the highest DRAM frequency delivers not only the best performance but also the lowest energy consumption. Moreover, undervolting the DRAM at that frequency did not result in a relevant reduction in the total GPU energy consumption, leading us to conclude that the weight of *DRAM* in the GPU energy consumption is not significant in relation to the *Core* energy consumption. In accordance, the highest *DRAM* frequency will be hence-forwardly considered, guaranteeing the maximum performance, and leaving the voltage control for the automatic DVFS system.

2) Cache: Fig. 2 presents the usable voltage interval and the normalized energy consumption and execution time for different V-F setups. To guarantee the best clarity of the following pictures, the considered voltage values were only annotated in Fig. 2 - all the following charts will adopt the same voltage levels. For frequencies below 1530MHz, no computation errors or crashes were observed. Only for frequencies as high as 1530 and 1600 MHz performing undervoltage resulted in the program crashing. A critical observation is that no computation errors occur, meaning that this architectural component either works normally or makes the GPU immediately to crash. This phenomenon is of significant importance to determine the root cause of failures when analyzing the CNN layers (see section IV). Furthermore, an increase of `OPS` allows for a higher amount of undervolt. Since this change only affects the stress over the DRAM-Cache controller (the number of cache accesses and hit-rate maintains the same), it can be concluded that it is the Cache-DRAM controller that limits the undervoltage range.

In what concerns the energy and performance variations, performing voltage scaling at the default frequency (dashed line) allows a reduction of energy consumption as high as 46.1%. However, this results in a performance degradation of 60.9%. The advantage of performing non-conventional DVFS becomes apparent by allowing for energy reduction without any performance degradation. In this case, it is possible to run the GPU Core at 1600 MHz and 1000 mV, thus achieving an energy reduction of 35.7% with no performance degradation.

Fig. 3 presents the obtained Energy-Delay Product (EDP)

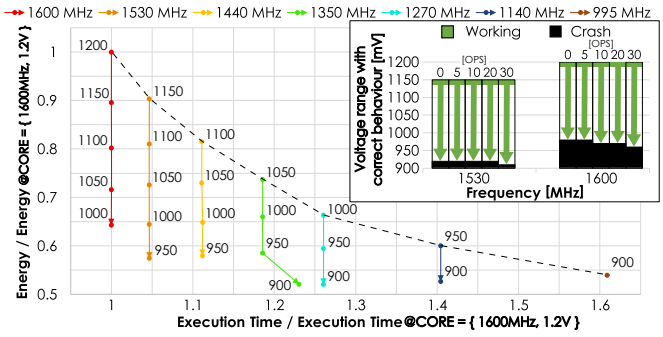


Fig. 2. CacheL2 - Normalized energy and performance variations and usable voltage range for `OPS=0`, when applying V-F scaling to the GPU Core. The values shown in the figure represent the applied core voltage and the dashed line connects the default F-V configurations.

Frequency [MHz]

Voltage [mV]	995	1140	1270	1350	1440	1530	1600
1200	1.75	1.49	1.30	1.22	1.12	1.06	1.00
1150	1.56	1.33	1.18	1.09	1.01	0.95	0.90
1100	1.40	1.19	1.05	0.98	0.91	0.85	0.80
1050	1.24	1.06	0.94	0.87	0.81	0.76	0.72
1000	1.10	0.95	0.84	0.78	0.72	0.67	0.64
950	0.98	0.84	0.75	0.69	0.64	**0.60**	
900	0.87	0.74	0.66	0.64			

Underlined results correspond to default V-F pairs.

Fig. 3. CacheL2 - Obtained Energy-Delay Product (EDP) for `OPS=0`, when applying V-F scaling to the GPU Core.

for the L2 Cache benchmark. This component favors the higher frequencies and minimum voltages, to achieve the lowest EDP product.

3) ALU: Fig. 4 represents the usable undervoltage range for the ALU benchmark. Since most DL frameworks adopt single-precision floating-point numbers by default, the presented results of the benchmark refer to this data type. For frequencies below 1440 MHz, the benchmark successfully runs for all voltage values. For higher frequencies, it is observed that after a certain amount of undervoltage, computation errors start appearing. The GPU crashes if a further undervoltage level is applied. It is also observable that the voltage margin increases with the operating frequency, from around 170mV for 1440MHz to around 210mV for 1600MHz, and that the existence of dependencies in the code reduces the voltage margin. However, when compared with the setup with no dependencies, the undervoltage range of the benchmark configuration that represent the general case (`d=3` - see Listings 3) is only reduced by 10mV.

An interesting phenomenon is observed in the energy-execution time plot for the highest frequencies. Performing undervoltage not only reduces energy consumption (as expected), but it also allows for faster execution time. An explanation can be found by analyzing the power consumption during the benchmark execution. For the default voltage, the power surpasses the power cap, which activates the GPU protection mechanisms, halting the execution until the power is reduced. By applying an undervoltage, the power significantly decreases (as $P_{Static} \propto V$ and $P_{Dynamic} \propto V^2$, see [5]) and allows a sustained maintenance of the desire DVFS configuration.

Finally, Fig. 5 presents the obtained EDP chart, where the configuration of 1440MHz and 950mV achieves the best

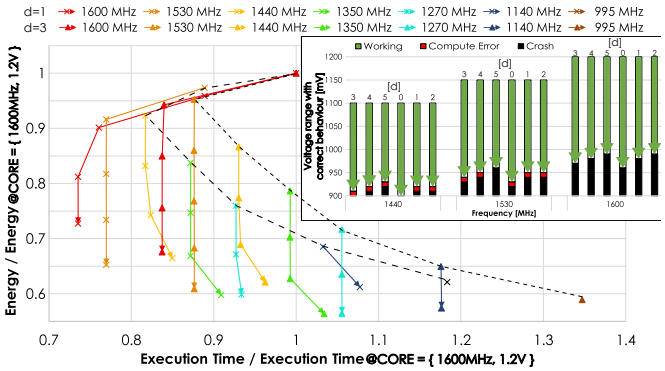


Fig. 4. ALU - Normalized energy and performance chart and usable voltage range when applying V-F scaling to the GPU Core, for the benchmark setup with different values of `d` (see Listing 3). The dashed lines connect the results for default F-V configurations.

d=1

Frequency [MHz]

Voltage [mV]	995	1140	1270	1350	1440	1530	1600
1200	1.45	1.24	1.11	1.07	1.03	1.03	1.00
1150	1.31	1.11	0.98	0.91	0.88	0.87	0.85
1100	1.17	0.99	0.88	0.82	0.75	0.71	0.69
1050	1.05	0.89	0.79	0.73	0.68	0.63	0.60
1000	0.93	0.80	0.70	0.65	0.61	0.57	0.53
950	0.83	0.71	0.62	0.58	0.56	**0.50**	
900	0.74	0.66	0.56	0.54			

d=3

Frequency [MHz]

Voltage [mV]	995	1140	1270	1350	1440	1530	1600
1200	1.56	1.33	1.16	1.09	1.02	1.00	1.00
1150	1.41	1.19	1.05	0.98	0.90	0.83	0.79
1100	1.26	1.07	0.94	0.87	0.81	0.75	0.71
1050	1.13	0.96	0.84	0.78	0.72	0.67	0.63
1000	1.01	0.86	0.75	0.70	0.64	0.60	0.57
950	0.90	0.76	0.67	0.62	0.60	**0.53**	
900	0.79	0.68	0.60	0.58			

Underlined results correspond to default V-F pairs.

Fig. 5. ALU - Obtained Energy-Delay Product (EDP) when applying V-F scaling to the GPU core for the benchmark setup with `d=1` (left) and `d=3` (right) (see Listing 3).

energy-efficiency, yielding an energy consumption reduction of 39.1% and an execution time improvement of 12.4%.

4) Reduction: The bars in the right side of Fig. 6 present the usable voltage range for the `reduction` benchmark. Due to the high pressure exercised on cache by this benchmark, the minimum usable voltage coincides with the one measured for the Cache benchmark. Due to the low utilization ratio of the arithmetic units (in comparison with the memories), performing undervoltage did not change the achieved performance for a given frequency (similar to the result of Fig. 2).

5) General Comments and Remarks: Fig. 6 presents a comprehensive comparison of the valid voltage ranges for all the considered architectural components of the GPU. The general motive is that the CacheL2 and the ALU are the two components that compromise the undervoltage capabilities. CacheL2 affects kernels that are more memory intensive, while the ALU limits those that are more compute-intensive. These conclusions help us to understand and predict the behavior of more elaborate DL applications (see section IV). This figure also presents corresponding results for branch instructions, which allowed to conclude that branch miss-prediction does not negatively impact the minimum voltage.

IV. CNN Layer Characterization with Independent Voltage and Frequency Scaling

As it was referred in Section II, Tang [4] has recently studied the impact of frequency scaling on the performance and energy consumption of DNNs executed in GPUs. By extending this study with the capability to also apply undervoltage

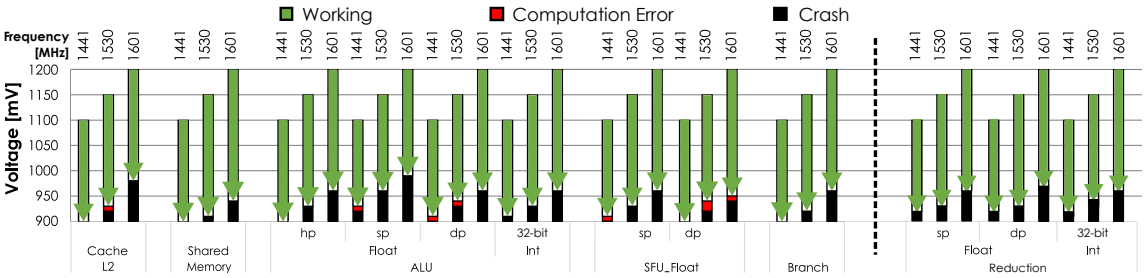


Fig. 6. Comparison of usable GPU core voltage ranges for all the considered architectural components of the GPU.

techniques, a broader range of DVFS configurations is herein envisaged to provide even greater benefits.

Another important characteristic of DNNs is their tolerance to a certain degree of computation errors [18], without any significant change in the training and inference results. As a consequence, it is important to complement the characterization that was performed in the previous section with the voltage scaling effects in DNNs training and inference phases and, in particular, with its influence on the computation errors that might occur when exploring the existing voltage margins.

Looking at the particular case of the Convolutional Neural Network (CNN), its feature extraction ability is supported on the convolution operator. Nonetheless, CNNs also include other types of layers, such as fully connected and pooling layers. However, convolution and fully connected layers take up to 97% of the GPU energy consumption [19], which makes them particularly suited to exploit energy saving mechanisms.

Hence, to extend the presented benchmarking results to a more ambitious analyses based on the use of high-level deep learning frameworks (e.g., PyTorch, TensorFlow), the default mathematical libraries provided by the considered GPU manufacturer (AMD) were extensively evaluated. This section reports the main achieved conclusions for the convolution operator and fully connected layers.

A. Convolution Layer

The MIOpen library[4] provides multiple convolution implementations, being the *Direct*, *GEMM* and *Winograd* the ones that are more often used. At the beginning of the execution, this library performs one convolution operation with each of these algorithms. Then, the algorithm that takes the smallest execution time is chosen and used to perform the remaining convolutions of the current layer.

To conduct this analysis, a set of convolution layer configurations was selected from the DeepBench benchmark[5], in order to understand how each algorithm is affected by DVFS, both in its inference and training phases.

Fig. 7 presents the set of valid voltage ranges that were obtained for the inference and training phases of the convolution layer. When comparing these results with those that were obtained in the previous section, it can be observed that some computation errors (and even some GPU crashes)

[4]github.com/ROCmSoftwarePlatform/MIOpen

[5]github.com/baidu-research/DeepBench

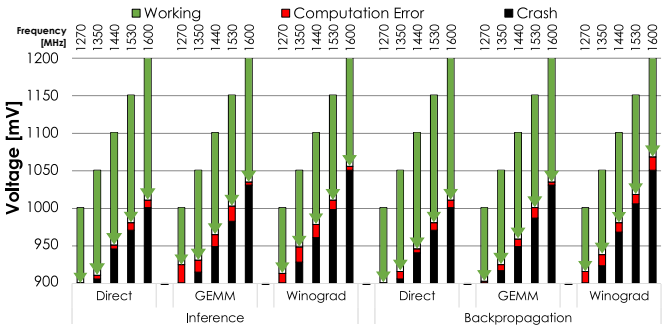


Fig. 7. Convolution Layer - Usable voltage range when applying V-F scaling to the GPU core.

were detected at lower voltages. In fact, since this operation is more complex and requires the utilization of multiple architectural components, the undervoltage limit is more likely to be violated by the voltage drops induced by the activation and deactivation of the GPU architectural components [16]. This phenomenon will make certain parts of the GPU not to work properly (even momentarily), producing an increased rate of computation errors and a increased crash threshold voltage.

When comparing the three convolution algorithms, it is observed that *Direct* allows for the greatest amount of undervoltage, followed by *GEMM* and *Winograd*. The *Direct* algorithm is the simpler of the three, with no data transformation and movement being necessary for its execution. In contrast, *GEMM* and *Winograd* need some data pre-processing before the convolution is performed. This extra step relies on the activation of more GPU components, making these algorithms more prone to induce voltage drops.

When comparing the training and inference phases, it is observed that they present similar undervoltage capabilities (for all algorithms), with the crash point diverging only around 10mV. However, the training algorithm is more prone to the introduction of computation errors.

Fig. 8 illustrates the impact of non-conventional DVFS on the energy consumption, execution time and EDP. When working with the default voltage level of each frequency (dashed lines), the *Direct* and *GEMM* algorithms exhibit a valley in their performance chart, with the frequency of 1530 MHz providing the best performance. In contrast, the *Winograd* achieves its best performance with the lowest frequencies. Upon the introduction of independent voltage scaling, it is

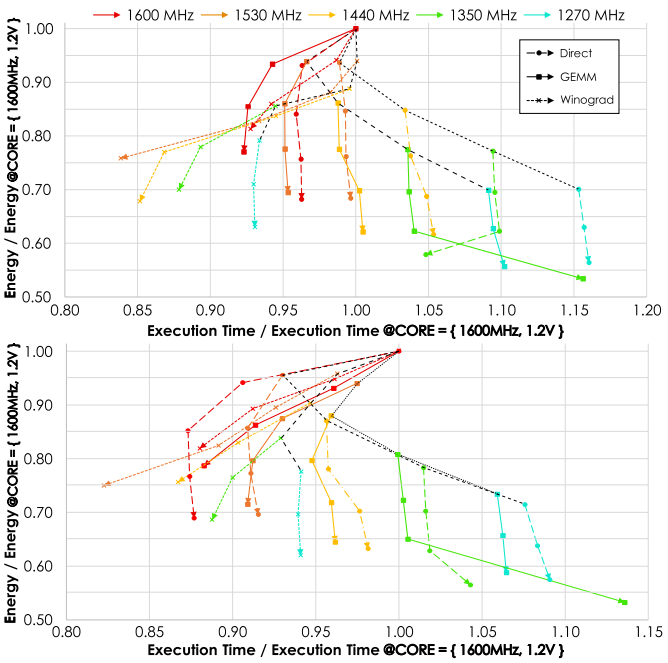


Fig. 8. Convolution Layer - Normalized energy and performance chart when applying V-F scaling to the GPU core during inference (top) and training (bottom) phases.

Inference

Voltage [mV] \ Direct, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.21	1.14	1.09	1.02	1.00
1150	1.09	1.03	0.97	0.93	0.90
1100	0.99	0.93	0.88	0.84	0.81
1050	0.90	0.84	0.79	0.76	0.73
1000	0.81	0.76	0.72	0.68	0.66
950	0.73	0.68	0.65		
900	0.65	**0.61**			

Voltage [mV] \ GEMM, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.15	1.13	1.04	1.02	1.00
1150	1.05	1.02	0.94	0.91	0.88
1100	0.94	0.90	0.85	0.82	0.79
1050	0.85	0.80	0.77	0.74	0.71
1000	0.76	0.72	0.70	0.66	
950	0.69	0.65	0.62		
900	**0.61**	0.62			

Voltage [mV] \ Winograd, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.14	1.10	1.05	1.02	1.00
1150	1.05	1.02	0.97	0.94	0.93
1100	0.95	0.92	0.88	0.87	0.81
1050	0.84	0.81	0.79	0.77	0.75
1000	0.74	0.70	0.67	0.64	
950	0.66	0.61	**0.58**		
900	0.59				

Training

Voltage [mV] \ Direct, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.15	1.09	1.03	1.02	1.00
1150	1.05	0.98	0.92	0.89	0.85
1100	0.94	0.88	0.83	0.78	0.74
1050	0.85	0.79	0.75	0.70	0.67
1000	0.77	0.71	0.69	0.64	0.60
950	0.69	0.64	0.62		
900	0.63	**0.59**			

Voltage [mV] \ GEMM, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.16	1.11	1.06	1.02	1.00
1150	1.05	0.99	0.95	0.92	0.89
1100	0.95	0.90	0.84	0.81	0.79
1050	0.86	0.81	0.75	0.73	0.69
1000	0.78	0.72	0.69	0.65	
950	0.70	0.65	0.62		
900	0.63	**0.61**			

Voltage [mV] \ Winograd, Frequency [MHz]	1270	1350	1440	1530	1600
1200	1.13	1.09	1.05	1.01	1.00
1150	1.03	0.99	0.95	0.92	0.91
1100	0.92	0.89	0.86	0.83	0.81
1050	0.82	0.78	0.75	0.74	0.72
1000	0.73	0.69	0.66	0.62	
950	0.65	0.61			
900	**0.58**				

Underlined results correspond to default V-F pairs.

Fig. 9. Convolution Layer - Energy Delay Product (EDP) when applying V-F scaling to the GPU core during inference (top) and training (bottom) phases. The dashed lines connect the results for default F-V configurations.

possible to improve the execution time and energy consumption (in comparison with the default voltage) by 8% and 23%, 19% and 8%, and 14% and 24% for the *Direct*, *GEMM* and *Winograd* algorithms, respectively. The EDP charts depicted in Fig. 9 indicate that the most energy efficient configuration for the three algorithms is at the lowest frequencies and maximum undervoltage possible. At these configurations, although the execution time is reduced by 16% (for the *Direct* and *GEMM* algorithms), it is still possible to achieve a reduction in energy consumption of up to 46%. The use of the most efficient configuration for the *Winograd* algorithm improves both the execution time and the energy by 15% and 32%, respectively.

B. Fully Connected Layer

The RocBlas library[6] provides a single API for matrix multiplication, the underlying mathematical operation of the fully

[6]github.com/ROCmSoftwarePlatform/rocBLAS

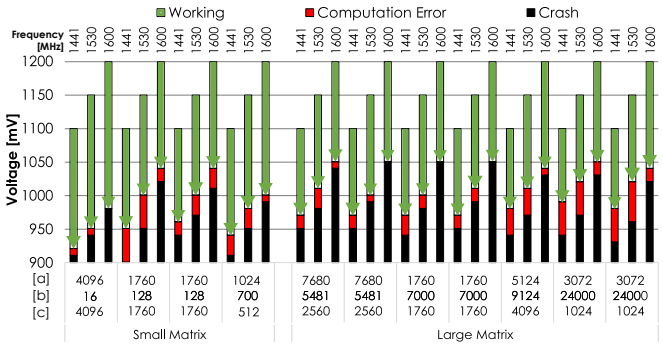


Fig. 10. Fully Connected Layer - Usable GPU core voltage range. Values represent matrix sizes, example $A_{a\times b} \cdot B_{b\times c}$

connected layer. By analyzing the performance counters and the kernels called by the library, it is possible to understand that multiplication is performed in one of two ways depending on the size of the matrices. According to [20], small matrices are first loaded to cache and all the operations are performed in this device, making the operation compute bounded. For large matrices, the multiplication starts and it is executed wherever the necessary data is available on the memory and local caches. The threshold size corresponds to the size of the L1 Cache.

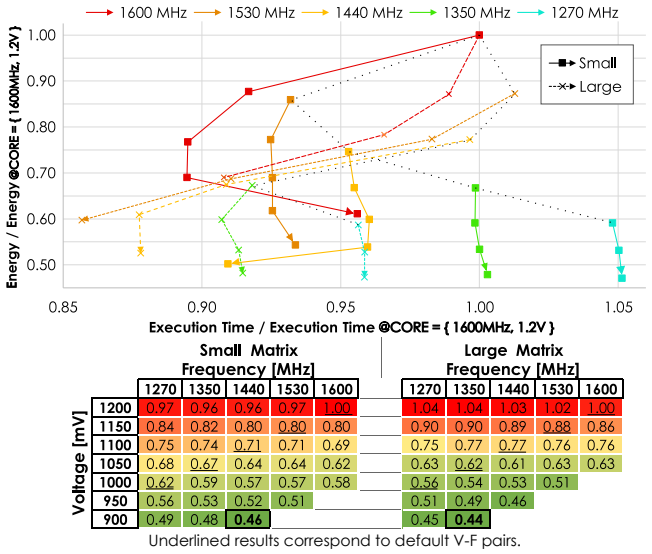
As a result, non-conventional DVFS will impact these two implementations of the matrix multiplication in different ways. For small matrices, it is the ALU that will limit the undervoltage. Consequently, it is expected that a valley-like shape is observable in the performance chart after the application of frequency scaling (see section III-C3). On the other hand, for large matrix sizes, the cache will be stressed the most, with constant requests on the DRAM-Cache controller limiting the undervoltage. Consequently, the results will be similar to those that were observed in section III-C2.

Fig. 10 and 11 illustrate the results of the experiment and confirm the prediction: for small matrix sizes it is possible to perform a higher degree of undervoltage. Furthermore, it is possible to have an improvement in the execution time and energy consumption for both cases. The EDP chart indicates the same energy efficiency configuration for both cases, which results in an average reduction of 52% in energy consumption and 8% of improvement of the execution time.

C. Error Analysis

To evaluate the occurrence of computation errors due to the utilization of non-conventional DVFS, each benchmark was executed both with the default "automatic" parameterization and with the V-F pair under testing, with the same input data. A *warmup_kernel* was also executed before each of these two runs, to fill the cache with random data.

For the architecture characterization benchmarks (see section III), a computation error was asserted whenever any of the output vectors differs between the executions. For the characterization of the CNN layers (see section IV), a different error metric was adopted due to the utilization of software libraries (versus custom kernels) operating over floating-point



Voltage [mV]	Small Matrix Frequency [MHz] 1270	1350	1440	1530	1600	Large Matrix Frequency [MHz] 1270	1350	1440	1530	1600
1200	0.97	0.96	0.96	0.97	1.00	1.04	1.04	1.03	1.02	1.00
1150	0.84	0.82	0.80	0.80	0.80	0.90	0.90	0.89	0.88	0.86
1100	0.75	0.74	0.71	0.71	0.69	0.75	0.77	0.77	0.76	0.76
1050	0.68	0.67	0.64	0.64	0.62	0.63	0.62	0.61	0.63	0.63
1000	0.62	0.59	0.57	0.57	0.58	0.56	0.54	0.53	0.51	
950	0.56	0.53	0.52	0.51		0.51	0.49	0.46		
900	0.49	0.48	**0.46**			0.45	**0.44**			

Fig. 11. Fully Connected Layer - Normalized energy and performance chart (top - the dashed lines connect the results for default F-V configurations), and obtained EDP (bottom) for small (left) and large (right) matrices when applying V-F scaling to the GPU core.

numbers (as before, generated from an uniform distribution in the interval $[0.1\ ;\ 1]$ to ensure that operations are never applied to numbers with significantly different exponent values). These libraries can launch the kernels in a different order, changing the order of operations, with a possible impact in the final result. In fact, by conducting experiments on the default voltage, it was observed that the order of kernel execution resulted in a relative output difference not greater than 10^{-6}. In accordance, a computation error was asserted whenever the relative difference in each position of the output vectors was greater than or equal to 10^{-5}.

1) Convolution Layer: Fig. 12 depicts the distribution of the output results of the convolution layer for the three considered convolution algorithms (at both inference and training phases) for the minimum usable voltage values (i.e., before GPU crash) across all considered core frequency values. The obtained results emphasize the little effect of the applied undervoltage on the computed values. Most of the output results are still fully accurate and only a small portion of the results present deviations. In fact, it should be emphasized that not only is the fraction of non-accurate results very small, but the normalized relative error of those non-accurate results has a very low

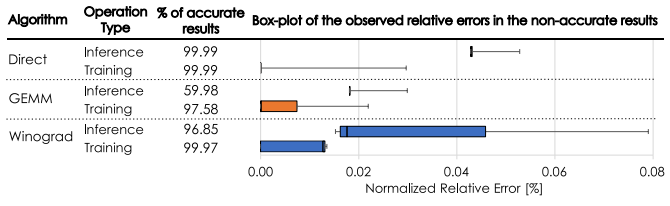


Fig. 12. Convolution Layer - Average percentage of accurate results and relative error distribution of non-accurate outputs for the minimum usable core voltage across all considered core frequency values.

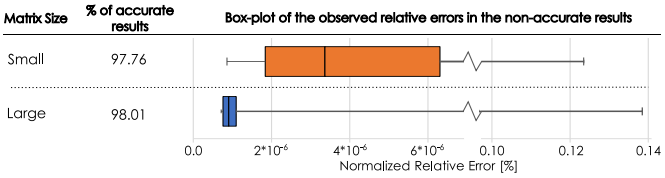


Fig. 13. Fully Connected - Average percentage of accurate results and relative error distribution of non-accurate outputs for the minimum usable core voltage across all considered core frequency values.

magnitude. A particular observation is worth noting about the results of the inference phase with the *GEMM* algorithm. Although the amount of non-accurate results is greater than in the other configurations, the magnitude of the normalized relative error is much smaller.

2) Fully Connected: Fig. 13 represents the same evaluation for the Fully Connected Layer, for both small and large matrices. Even at these extreme configurations, it is observed that most results are still computed with full accuracy (98% of the cases), with a normalized relative error as low as 1.37×10^{-3} (on the remaining 2% of the cases).

D. Complete CNN training with non-conventional V-F

The previously obtained results evidence that the small number and relative amount of computation errors introduced by these under-voltage conditions is well cooped with the operations that are conducted in DNN layers. However, the same evaluation urged to be done for the whole network. Hence, four complete well-known CNN models (AlexNet, LeNet, VGG11 and WideResNet) were trained under the proposed non-conventional DVFS. For each case, the previously performed error analyses was applied to evaluate the best V-F configurations and to assess the induced errors.

Table III presents the median classification accuracy over ten runs. Results show that, when compared with the default setup (i.e., no undervoltage), the introduced computation errors do not induce any significant change in the network's final training accuracy. Also, Table IV presents the best configurations considering maximum undervoltage at highest frequency, and V-F configuration minimizing EDP.

TABLE III
COMPARING CNN TRAINING ACCURACY WITH THE APPLICATION OF DIFFERENT UNDERVOLTAGE LEVELS.

Amount of undervolt [mV]	AlexNet [%]	LeNet [%]	VGG11 [%]	WideResNet [%]
0	76.59	59.84	86.14	80.32
50	76.48	60.08	86.14	80.39
100	76.60	59.94	86.04	80.23
150	76.61	60.12	86.39	80.08
Number of trained epochs	50	100	30	30

Finally, Table V emphasizes and summarizes the main achievements of this research, by comparing the CNN execution at the default V-F setup, with frequency scaling using default voltage values, and with non-conventional DVFS. It considers two configurations: (i) at the highest frequency, and (ii) at minimum EDP. As it can be observed, by exploring non-conventional DVFS, it is possible to significantly improve

TABLE IV
NON-CONVENTIONAL V-F SETTINGS PER CNN MODEL.

Configuration	Frequency and Voltage configuration [MHz - V]			
	AlexNet	LeNet	VGG11	WideResNet
Default GPU setup	1600 - 1.2	1600 - 1.2	1600 - 1.2	1600 - 1.2
Standard DVFS*	1270 - 1.0	1270 - 1.0	1270 - 1.0	1270 - 1.0
Proposed @ highest freq.	1600 - 1.1	1600 - 1.05	1600 - 1.1	1600 - 1.05
Proposed @ best EDP	1530 - 1.0	1270 - 1.0	1440 - 1.0	1530 - 1.0

* DVFS setup that optimizes EDP using manufacturer voltage values.

TABLE V
EVALUATION OF PERFORMANCE, ENERGY, AND EDP WHEN APPLYING NON-CONVENTIONAL DVFS IN THE TRAINING OF NEURAL NETWORKS.

Metric	Selected configuration	Improvement *vs* default V-F			
		AlexNet	LeNet	VGG11	WideResNet
Energy	At highest frequency	24%	20%	22%	23%
	At best EDP	38%	38%	33%	38%
Training time	At highest frequency	1%	2%	0%	6%
	At best EDP	3%	-3%	-2%	0%
EDP	At highest frequency	21%	22%	22%	23%
	At best EDP	38%	41%	32%	36%
		Improvement *vs* F scaling with default V-F pairs			
Energy	At best EDP	-2%	0%	-1%	-2%
Training time	At best EDP	8%	0%	6%	10%
EDP	At best EDP	3%	0%	3%	6%

a positive value indicates an improvement vs the default V-F configuration of the GPU.

all three metrics (energy, training time, and EDP) without compromising the resulting accuracy of the whole CNN. The main benefit of our approach is allowing the utilization of higher or even the highest frequency (maximizing performance) while having similar energy consumption to using the lower frequency/energy savings performance levels.

V. CONCLUSION

The presented research shows that there is a great benefit in performing non-conventional DVFS while running CNNs (and DNNs in general). The conducted GPU architecture characterization allowed to understand that the L2 Cache and the ALU are the most sensitive components when performing undervoltage. It was also observed that it is safe to undervolt the considered GPU between 150 mV and 200 mV without significantly constraining the accuracy of results. On the other hand, this allows for significant energy gains and, in some cases, it even improves the attained performance. Applying non-conventional DVFS to the convolution and fully connected CNN layers reduces the EDP by 50%, at a cost of introducing a small amount of computation errors. Nevertheless, it was shown that the application of non-conventional DVFS to the training of complete CNN models does not significantly affect the final test accuracy. The obtained results also indicate that it is possible to improve the GPU EDP while training complete CNN models by an average of 36.7%. Overall, this paper shows how to charactetize a GPU architecture sensitivity to undervoltage, in order to reduce the GPU energy consumption without degrading the attained results.

ACKNOWLEDGMENTS

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects UIDB/CEC/50021/2020, PTDC/EEI-HAC/30485/2017 and PCIF/MPG/0051/2018.

REFERENCES

[1] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[2] S. Mittal and S. Vaishay, "A survey of techniques for optimizing deep learning on GPUs," *Journal of Systems Architecture*, vol. 99, p. 101635, Oct. 2019, original Publisher: Elsevier.

[3] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated DVFS and Precision Control for Deep Neural Networks," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 136–140, Jul. 2019.

[4] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The Impact of GPU DVFS on the Energy and Performance of Deep Learning: an Empirical Study," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ser. e-Energy '19. Phoenix, AZ, USA: Association for Computing Machinery, Jun. 2019, pp. 315–325.

[5] J. Guerreiro, A. Ilic, N. Roma, and P. Tomas, "GPGPU Power Modeling for Multi-domain Voltage-Frequency Scaling," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 789–800.

[6] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, "GPU Static Modeling Using PTX and Deep Structured Learning," *IEEE Access*, vol. 7, pp. 159 150–159 161, 2019.

[7] ——, "Modeling and Decoupling the GPU Power Consumption for Cross-Domain DVFS," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2494–2506, Nov. 2019.

[8] Q. Wang and X. Chu, "GPGPU Performance Estimation with Core and Memory Frequency Scaling," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018, pp. 417–424.

[9] K. Fan, B. Cosenza, and B. Juurlink, "Predictable GPUs Frequency Scaling for Energy and Performance," in *Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP 2019. Kyoto, Japan: Association for Computing Machinery, 2019, pp. 1–10.

[10] ——, "Accurate Energy and Performance Prediction for Frequency-Scaled GPU Kernels," *Computation*, vol. 8, no. 2, p. 37, 2020.

[11] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, "DVFS-aware application classification to improve GPGPUs energy efficiency," *Parallel Computing*, vol. 83, pp. 93–117, 2019.

[12] C. Kalogirou, P. Koutsovasilis, C. D. Antonopoulos, N. Bellas, S. Lalis, S. Venugopal, and C. Pinto, "Exploiting CPU Voltage Margins to Increase the Profit of Cloud Infrastructure Providers," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2019, pp. 302–311.

[13] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, and S. Das, "Harnessing voltage margins for energy efficiency in multicore CPUs," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. Cambridge, Massachusetts: Association for Computing Machinery, Oct. 2017, pp. 503–516.

[14] F. Nakhaee, M. Kamal, A. Afzali-Kusha, M. Pedram, S. M. Fakhraie, and H. Dorosti, "Lifetime improvement by exploiting aggressive voltage scaling during runtime of error-resilient applications," *Integration*, vol. 61, pp. 29–38, Mar. 2018.

[15] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, and V. J. Reddi, "Safe limits on voltage reduction efficiency in GPUs: A direct measurement approach," in *MICRO*, Dec. 2015, pp. 294–307, iSSN: 2379-3155.

[16] R. Thomas, K. Barber, N. Sedaghati, L. Zhou, and R. Teodorescu, "Core tunneling: Variation-aware voltage noise mitigation in GPUs," in *HPCA*, Mar. 2016, pp. 151–162, iSSN: 2378-203X.

[17] J. Tan, S. L. Song, K. Yan, X. Fu, A. Marquez, and D. Kerbyson, "Combating the Reliability Challenge of GPU Register File at Low Supply Voltage," in *PACT*. Haifa, Israel: ACM Press, 2016, pp. 3–15.

[18] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *DATE*, Mar. 2015, pp. 701–706, iSSN: 1558-1101.

[19] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs," in *BDCloud-SocialCom-SustainCom*, Oct. 2016, pp. 477–484.

[20] I. Masliah, A. Abdelfattah, A. Haidar, S. Tomov, M. Baboulin, J. Falcou, and J. Dongarra, "High-Performance Matrix-Matrix Multiplications of Very Small Matrices," in *Euro-Par 2016: Parallel Processing*, P.-F. Dutot and D. Trystram, Eds. Cham: Springer International Publishing, 2016, vol. 9833, pp. 659–671, series Title: Lecture Notes in Computer Science.