Journal of Parallel and Distributed Computing 165 (2022) 32–51

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Decoupling GPGPU voltage-frequency scaling for deep-learning applications

Francisco Mendes, Pedro Tomás, Nuno Roma [*]

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

A R T I C L E   I N F O

Article history:
Received 28 February 2021
Received in revised form 26 October 2021
Accepted 7 March 2022
Available online 21 March 2022

Keywords:
GPU
DVFS
Undervoltage
Energy efficiency
CNN

A B S T R A C T

The use of GPUs to accelerate DNN training and inference is already widely adopted, allowing for a significant performance increase. However, this performance is usually obtained at the cost of a consequent increase in energy consumption. While several solutions have been proposed to perform voltage-frequency scaling on GPUs, these are still one-dimensional, by simply adjusting the frequency while relying on default voltage settings. To overcome this limitation, this paper introduces a new methodology to fully characterize the impact of non-conventional DVFS on GPUs. The proposed approach was evaluated on two devices, an AMD Vega 10 Frontier Edition and an AMD Radeon 5700XT. When applying this non-conventional DVFS scheme to DNN training, 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 of the EDP (up to 41%) on the training procedure of CNN models, with no degradation of the networks accuracy.

1. Introduction

In the last few years, Deep Neural Networks (DNNs) have had a significant impact on industry and society, by allowing for important breakthroughs in many application domains, including computer vision, speech recognition, natural language processing, drug discovery, genomics, etc. [20]. However, although the end-user perception of DNNs is most often circumscribed to their inference phase, before DNN models can be deployed they need to pass by a costly training procedure, which usually requires the use of significant computational resources, 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, to cope with the associated computational effort and reduce the processing time. Nowadays, Graphics Processing Units (GPUs) have emerged as the *de-facto* computational accelerators for the execution of DNNs, both for the training and inference phases. They differ from conventional CPUs by including thousands of computing cores (Compute Units - CUs) and a large bandwidth memory module. As a result, GPUs are now commonly deployed on most supercomputers, data centers and other computational infrastructures related with the development and deployment of artificial intelligence algorithms.

In another perspective, researchers have also been investigating and developing different software infrastructures, algorithms and techniques to manage and optimize the execution of DNNs on GPUs [13]. However, most optimization techniques neglect the energy impact of the training phase of Deep Learning (DL) models, usually resulting in high electricity costs.

To overcome this problem, several research works 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 [14,17], eventually trading network accuracy with increased processing performance and lower energy consumption.

Researchers have also looked at other alternative approaches, such as the exploitation of Dynamic Voltage and Frequency Scaling (DVFS) on both the inference and training phases. In fact, by carefully selecting the used voltage-frequency levels, significant energy savings can be obtained, although depending on the considered DNN architecture and computing device [22]. This is achieved through a careful balance between the performance and power consumption of the different GPU components (particularly the core and global memory) in order 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 [22], by increasing the voltage and frequency levels at certain parts of the applications (as long as the GPU total power envelope and thermal limits are not surpassed).

* Corresponding author.
E-mail address: nuno.roma@inesc-id.pt (N. Roma).

https://doi.org/10.1016/j.jpdc.2022.03.004
0743-7315/© 2022 Elsevier Inc. All rights reserved.

Nevertheless, most state-of-the-art works only consider tightly coupled Frequency-Voltage (F-V) levels, often predefined by GPU manufacturers. However, these F-V levels usually neglect 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, F-V scaling techniques are applied for the training and inference of state-of-the-art networks, leading to the conclusion that significant energy savings and Energy Delay Product (EDP) gains can be attained by working outside of the conventional range of the F-V levels.

In accordance, the main contributions of this work are:

- Proposal of a new methodology, and a corresponding open-source synthetic benchmark suit,[1] to fully characterize the impact of F-V scaling on modern GPU architectures;
- Evaluation of the proposed methodology with two different GPUs (AMD Vega 10 Frontier Edition and AMD Radeon 5700 XT), leading to the observation that both devices can be safely undervolted (from the default voltage setup), with the DRAM-Cache controller and the Arithmetic and Logic Unit (ALU) being the most sensitive components to voltage drops.
- Demonstration that decoupled F-V scaling provide significant energy savings, but also that it can also contribute to some performance gains, as a side effect of the observed power savings.
- Evaluation of the introduced computational errors (due to undervoltage), showing that it can be safely applied to both the training and inference phases of DNNs without compromising the networks accuracy.

The rest of the paper is organized as follows. Section 2 introduces some related work on applying DVFS to DNNs and the background knowledge that sustains it. Section 3 describes the proposed methodology and demonstrates the implications of applying decoupled F-V scaling to the primary GPU architectural components. Section 4 outlines the application of this technique on different DNN layers and section 5 concludes this work.

2. 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 [4–6], Wang [25] and Fan [1,2]) to determine the most suitable F-V pair to the running application. 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., Guerreiro [7]), relying on performance counters gathered from previous executions.

Nevertheless, most of these solutions rely on tightly coupled F-V 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. In contrast, this paper focuses on decoupling the F-V levels by investigating and then reducing the per-kernel voltage margins to a minimum.

Nonetheless, other works have already investigated voltage guardband reduction on different processors to improve energy efficiency. Kalogirou et al. [8] explore such a concept to reduce the CPU consumption on cloud data centers. Papadimitriou et al. [16] comprehensively characterized the voltage guardbands in different ARM processors to predict the minimum working voltage using performance counters across different types of applications. Nakhaee et al. [15] 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. [10] studied the undervoltage effect in NVIDIA GPUs to conclude that there is a significant voltage guardband dependent on application kernels, which can result in up to 25% energy savings if reduced to a minimum. Similarly, Thomas et al. [23] 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. [21] 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 additional impact of frequency scaling, as it will be addressed in this paper, making their experimental studies at a fixed frequency.

This same problem was also investigated in Field Programmable Gate Array (FPGA) devices. As an example, [19] and [18] characterized the BRAMs behavior, although again, disregarding the frequency scaling influence.

Hence, 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 herein presented considers a complete detachment of the voltage and frequency setups 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 truly 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.

3. Architecture characterization with independent voltage and frequency scaling

To thoroughly characterize the GPU architecture sensitivity to decoupled F-V scaling, a set of kernels was devised, which were carefully crafted to stress different GPU components. By following the workflow depicted in Fig. 1, each kernel was executed under different F-V configurations, exploring the range of possible frequencies and voltages (coined as Usable Exploration Space - UES). This allows determining the frequency-dependent minimum operating voltage that (still) leads to a correct GPU operation (V_{min}) and to understand the impact of independent voltage scaling on performance and energy consumption. The kernels, presented in Table 1, are described in the following subsections and are available as open source at `https://github.com/hpc-ulisboa/nonconventional-dvfs`.

3.1. Characterization benchmarks

The devised benchmarks individually characterize the different components of the two main F-V domains (*global memory* and *core*), namely: DRAM, Shared Memory, L2 Cache, 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

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

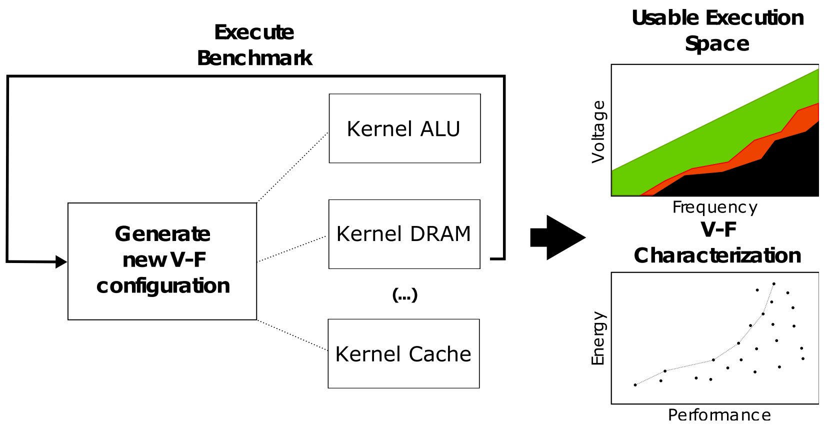


Fig. 1. Overview of the conducted methodology and its objective outputs.

Table 1
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 scheduling operation
Mix (reduction)	FP64/32/16	Evaluates the simultaneous impact of stressing multiple GPU components

`DATA_TYPE` placeholder with standard integer, single and double precision floating-point types. However, due to space limitations, only the experimental results that reveal a greater sensitivity to V_{min} will be herein reported, corresponding to DRAM, Cache, and MAC.

Finally, it will also be evaluated a coarser and more representative kernel in many GPGPU applications - the reduction - simultaneously stressing multiple architectural components.

3.1.1. DRAM

The DRAM benchmark (see Listing 1) was devised (and validated through GPU counters) to determine the memory sensitivity to decoupled F-V scaling. In this benchmark, including a single kernel, each thread is responsible for accessing the global memory and retrieving two values. The accesses are defined to guarantee coalescing between threads and ensure a high memory throughput. These two values are summed and placed on an output vector. A constant value (C) determines the distance between accesses, and its value is sufficiently large to guarantee that the new data to be fetched is not present on the local caches. For each data fetch, the `OPS` parameter 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.

Listing 2 renders a benchmark that was specifically designed to evaluate the occurrence of the *bit-flip* phenomenon and the preservation of the data in memory when exposed to undervoltage. A *bit-flip* is an unintentional state switch (from 0 to 1, or vice versa) of any individual bit stored on a DRAM or other kinds of volatile memories. Kim [9] exposed the existence of *bit-flipping* on CPU DRAM, induced by the continuous activation of a DRAM row, trying to corrupt the data in near-by rows. The benchmark presented on Listing 2 is a GPU implementation of the *rowhammer*[2] benchmark, which was specifically developed by Google to test this exact problem, and was used here to assess if undervolting the GPU increases the possibility of *bit-flipping*.

For the considered memory tests (DRAM, Cache and Shared Memory), only the integer data type was considered, since the effects on memory are the same for floating-point operations and it is easier to identify data corruption with integer data types.

3.1.2. Cache

Even though the caches are part of the memory subsystem, they are under the *core* F-V domain. The devised benchmark, presented in Listing 3, follows a similar stressing pattern to Listing 1. However, it includes an addition external loop on variable `k`, which ensures that after the first execution the data is available on one of the two levels of cache. Hence, this kernel is able to test both the state machines responsible for cache management and communication with the DRAM (as verified by relying on GPU counters).

For this benchmark, the number of issued requests to the cache and to the DRAM-cache controller stays the same independently of the `OPS` value. However, the frequency of those requests is inversely proportional to `OPS`.

3.1.3. Shared memory

The benchmark devised to characterize the Shared Memory is presented in Listing 4. This component is shared between threads in the same compute unit (CU) and it is used to ensure the communication between the different executing threads. Hence, the developed benchmark uses this component to move data around. Similarly to the DRAM and cache kernels, the `OPS` parameter controls the distance between memory requests, allowing to control the level of stress over this component. To guarantee a correct and repeatable execution, the synchronization directive `__sync-`

[2] github.com/google/rowhammer-test.

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

    #pragma unroll
    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++) asm volatile ("");
        OUT[threadId] = r0;
    }
}
```

Listing 1: DRAM Benchmark Code.

```
void  DRAMstresser(DATA_TYPE *IN, DATA_TYPE *OUT) {
    const int ite = threadIdx.x;
    volatile register DATA_TYPE r0;

    // Initiate output memory
    OUT[ite] = IN[ite];
    OUT[ite + THREADS * BLOCKS] = IN[ite + THREADS * BLOCKS];

    for (int i = 0; i < N; i++) {
        r0 = IN[ite];
        #pragma unroll
        for(int j = 0; j < OPS; j++) asm volatile ("");
        OUT[ite] = r0;
    }
}
```

Listing 2: DRAM Bit-Flip Stress Test Code - *rowhammer* inspired benchmark.

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

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

Listing 3: CacheL2 Benchmark Code.

```
void  SharedMemorycode(DATA_TYPE *IN, DATA_TYPE *OUT) {
    __shared__ DATA_TYPE shared[THREADS];
    const int ite = blockIdx * THREADS + threadIdx;
    const int t = threadIdx.x;
    const int tr = THREADS - t - 1;
    volatile register DATA_TYPE r0 = IN[ite];

    for (int i = 0; i < N; i += UNROLL_ITE) {
        #pragma unroll
        for(int j = 0; j < UNROLL_ITE; j++)
            shared[t] = r0;
            __syncthreads();
            for(int k = 0; k < OPS; k++)
                asm volatile ("");
            r0 = shared[tr];
            __syncthreads();
    }
    OUT[ite] = r0;
}
```

Listing 4: Shared Memory Benchmark code.

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

Listing 5: ALU Benchmark Code.

threads() is used to synchronize all the threads that use the same shared memory.

3.1.4. MAC

Listing 5 presents the devised benchmark to stress the ALU. A greater emphasis was devoted to the MAC operation, due to its prevalence in the 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 warp 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 also be 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 it 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, with a medium level of dependencies still existing in the code.

3.1.5. *Non-linear operations*

Besides the MAC operation, the ALU also computes a set of non-linear functions, including exponential, logarithmic and trigonometric operations. For such purpose, it uses the Special Function Unit (SFU). The devised benchmark, presented in Listing 6, tests those operations to find if the undervoltage mechanism, when in

```
void  NonLinearcode(DATA_TYPE *IN, DATA_TYPE *OUT) {
    const int ite = (blockIdx * THREADS + threadIdx) * 4;
    volatile DATA_TYPE r0, r1, r2, r3;

    r0=IN[ite]; r1=IN[ite+1]; r2=IN[ite+2]; r3=IN[ite+3];

    for(j=0; j < N; j+= UNROLL_ITE) {
        #pragma_unroll
        for(j=0; j < UNROLL_ITE; j++) {
            r0 = exp(r2);
            r1 = cos(r3);
            r2 = log(r0);
            r3 = sin(r1);
        }
    }
    OUT[ite/4] = r0;
}
```

Listing 6: Non-linear Operations Benchmark Code.

use, introduces some modifications in the critical path and so, negatively influences the guardband size.

3.1.6. Branches

Listing 7 presents the benchmark devised to test the DVFS influence on the execution of branches on a kernel, as these may cause divergence between threads and/or break the execution flow. In this listing, the `#define BRANCHES` directive sets the desired number of branches to test, with the corresponding value being defined from the set {1,2,4,8}.

3.1.7. Reduction

The `reduction` benchmark, presented in Listing 8, performs the reduction of a N-sized vector to $N/blockDim$ elements, by performing an element-wise sum. The tested implementation of this operation is considered the one that achieves the highest performance, and so, it is the most widely used. 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.

3.2. Non-conventional DVFS experimental setup

The devised benchmarks were applied to characterize two AMD GPUs from different architectural generations: the AMD Vega 10 Frontier Edition (GNC5) and the AMD Radeon 5700 XT (RDNA), whose specifications are presented in Table 2. The sole inclusion of AMD devices comes from the reduced availability of drivers and convenient software APIs from other manufacturers (e.g., NVIDIA) for an independent and decoupled control over the GPU frequency and voltage. In accordance, the GPU vendor rocm-smi[3] tool was used to set an independent and decoupled control over the frequency and voltage levels applied to these devices. Moreover, to ensure that the GPU power cap does not limit any of the intended configurations, its value was changed to match each GPU thermal design cap (220W to 300W for the Vega 10 and 190W to 285W on the Radeon 5700 XT). The GPUs under test were 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 2, were selected as the starting points for the non-conventional DVFS. For each considered DVFS configuration, the benchmarks were executed ten times to obtain the median value of the execution time and energy consumption.

It is worth noting that the Radeon 5700 XT GPU does not have a DVFS DRAM domain, applying always the same frequency and voltage level on that component. For each frequency, the devised experiments on this particular GPU device starts at the maximum voltage ($1200mV$) and a gradual undervoltage of the F-V domain under test is applied with $50mV$ steps.

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 significantly 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 level, some *computational errors* are observed 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 GPU crash point, the undervoltage step was reduced to $10mV$ when computation errors start occurring. Furthermore, when dealing with the *DRAM* F-V domain, the *Core* F-V domain was set to its default values; during the characterization of the *Core* F-V domain, the highest frequency and default voltage of the *DRAM* was selected.

The GPU power consumption was measured using gpowerSAMPLER[4] [4], at every millisecond. At the end of the execution, the energy is computed as the integral of all the measurements taken.

3.3. Characterization results

For an easier understanding of the obtained results, the following charts represent only the data-points that correspond to voltage levels equal-to and lower-than the default voltage of each frequency level (no interesting data is found at higher voltage levels). For the Core domain, only the frequencies above and equal to 1440 MHz (for the Vega 10) and 1600 MHz (for the Radeon 5700 XT) are shown in order to reduce the charts size. For all frequencies below, it was found that the GPUs could be run at any voltage from the default to the minimum (900 mV for the Vega 10 and 750 mV for the Radeon 5700 XT). Furthermore, the execution time, energy consumption, and energy-delay product charts were normalized to the results achieved at the highest core frequency and default voltage level – Vega 10: (1600 MHz; 1200 mV) and Radeon 5700 XT: (2000 MHz; 1200 mV)) –, so that a smaller number indicates an improvement regarding the base configuration.

3.3.1. DRAM

Fig. 2 illustrates the usable voltage range of the DRAM domain in the VEGA 10 GPU, together with its normalized performance and energy consumption when varying the `OPS` parameter between 0 and 50 operations (see Listing 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 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 to the conclusion that

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

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

```
#define BRANCHES VALUE
void  Branchescode(DATA_TYPE *IN, *OUT) {
    const int ite = (blockIdx * THREADS + threadIdx;= % MEM_BLOCK;
    const int branch =  ite % BRANCHES;

    volatile register DATA_TYPE r0, r1, r2, r3;

    for (int i = 0; i < N; i++) {
        if(branch == 0)   r0 = IN[ite];
        #if BRANCHES >= 4
            else if(branch == 1)   r1 = IN[ite];
            else if(branch == 2)   r2 = IN[ite];
        #elif BRANCHES == 8
            else if(branch == 3)   r3 = IN[ite];
            else if(branch == 4)   r0 = IN[ite];
            else if(branch == 5)   r1 = IN[ite];
            else if(branch == 6)   r2 = IN[ite];
        #elif BRANCHES >= 2
            else {r3 = IN[ite];}
        #endif
        OUT[ite] = r0;
    }
}
```

Listing 7: Branches Benchmark Code.

```
void Reduction(DATA_TYPE * idata, DATA_TYPE * 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 8: Reduction Kernel Code.

Table 2
Considered GPUs in the conducted experimental characterization.

Model	Unit	Vega 10	Radeon 5700 XT
Architecture		GNC5	RDNA
CUs		64	40
DRAM size	[GB]	16	8
Default Power Cap	[W]	220	190
Core frequency range	[MHz]	[852 - 1980]	[800 - 2050]
Core voltage range	[mV]	[900 - 1200]	[750 - 1200]
DRAM frequency range	[MHz]	[500 - 1200]	Fixed at 1000
DRAM voltage range	[mV]	[800 - 1200]	Fixed at 1000
Default Frequency-Voltage (F-V) setups			
Core	[(MHz ; mV)]	{(995 ; 900), (1140 ; 950), (1350 ; 1050), (1440 ; 1100), (1530 ; 1150), (1600 ; 1200)}	{(1200 ; 950), (1400 ; 1000), (1600 ; 1050), (1800 ; 1100), (2000 ; 1200)}
DRAM	[(MHz ; mV)]	{(500 ; 900), (800 ; 950), (950 ; 1000)}	(1000 ; 1000)

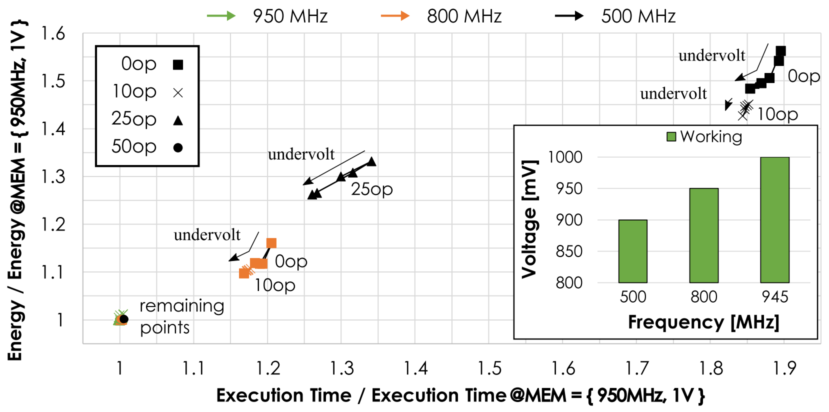


Fig. 2. DRAM DVFS in the VEGA 10 GPU: Normalized energy consumption, execution time and usable DRAM voltage for each frequency configuration.

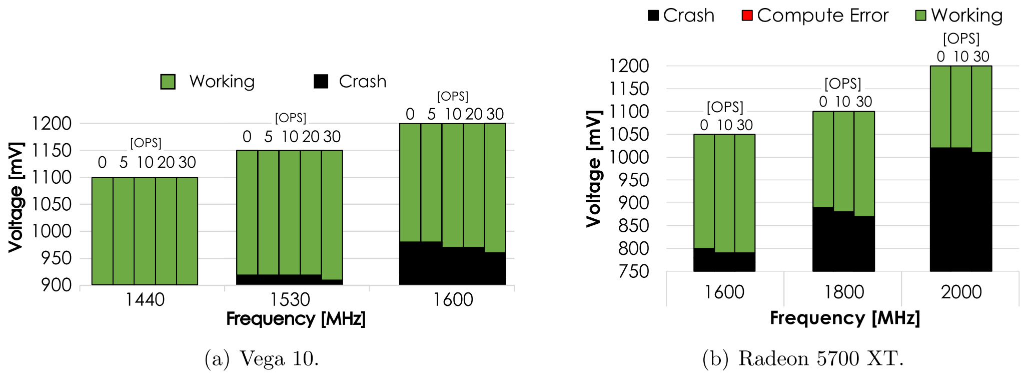


Fig. 3. Core domain - Cache L2 - Usable voltage levels for each frequency configuration when varying the cache stress (OPS value).

the relative weight of the *DRAM* power consumption is not significant when compared with the *Core*. In accordance, the highest *DRAM* frequency will be hence-forwardly considered, guaranteeing the maximum performance, and leaving voltage at the default levels.

3.3.2. Cache

Figs. 3(a) and 3(b) present the usable voltage intervals for different F-V setups for the two tested GPUs when executing the proposed Cache L2 benchmark code (see Listing 3). Only for Core-domain frequencies higher than 1530 MHz the applied undervoltage resulted in the program crash. A critical observation is that no computation errors occur, meaning that the Cache L2 either works normally or makes the GPU crash. This phenomenon is of significant importance to identify the root cause of failures: whenever it is observed a GPU crash without prior computation errors, the Cache L2 might be the preeminent component causing it. Furthermore, it is observed that an increase of the `OPS` parameter 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 remains the same), it can be concluded that it is the Cache-DRAM controller that limits the undervoltage range.

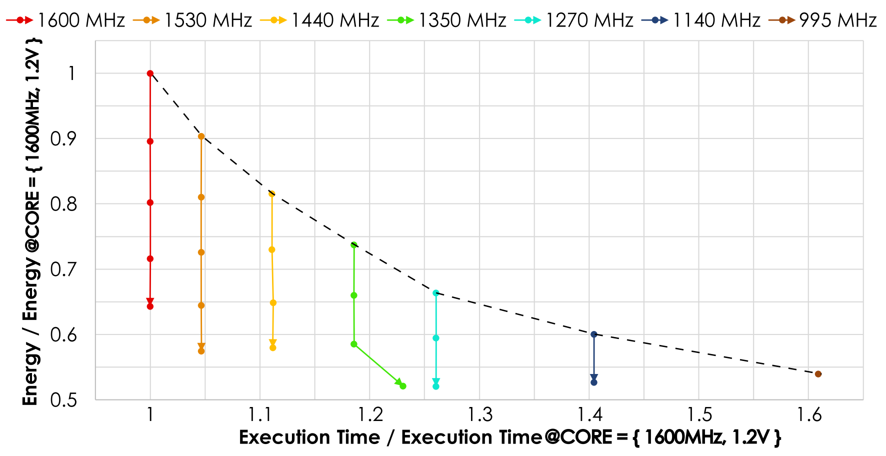
A similar behavior is observed for the Radeon 5700 XT GPU. For frequencies below 1600 MHz, it is possible to use the GPU at the lowest voltage level without any computation errors or crashes being observed. For higher frequencies, the GPU may crash when the voltage is reduced. Increasing the value of the `OPS` parameter (decreasing the Cache stress) increases the undervoltage capabilities.

In what concerns the energy and performance variations (see Figs. 4(a) and 4(b)), applying voltage scaling at the default frequency (dashed line) allows a reduction of energy consumption as high as 46.1% for the Vega 10 GPU and 30% for the Radeon 5700 XT GPU. However, this also results in a performance degradation of 61% on both GPUs. Nevertheless, the advantage of performing non-conventional DVFS becomes clear by setting the operating point to one of the observed configurations that provide some energy reduction without any performance degradation. As an example, it is possible to run the GPU Core at 1600 MHz and 1000 mV (the last operating point of the red line in Fig. 4(a)), thus achieving an energy reduction of 35.7% with no performance degradation. In the Radeon 5700 XT case, the use of non-conventional F-V pairs yields a minimum energy consumption that is even lower than the most energy-saving default F-V configuration (rightmost point of the black dashed line in Fig. 4(b)), with the configuration of {1800 MHz; 0.8V} achieving an energy consumption reduction of 41% with only a 10% performance downgrade (last operating point of the orange line in Fig. 4(b)). It can also be observed that the greatest reduction of the energy consumption is twice the one that is achieved at the most energy-saving default F-V configuration.

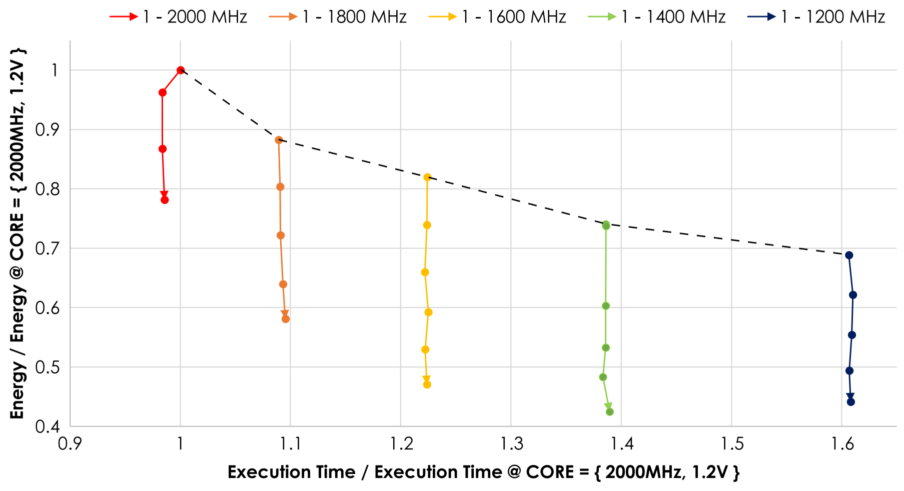
Figs. 5(a) and 5(b) present the obtained Energy-Delay Product (EDP) for the L2 Cache benchmark. As it can be observed, this component favors the utilization of the lowest voltage levels to achieve an EDP improvement of over 40%. Hence, when comparing the non-conventional F-V results to the default ones, it can be concluded that using the proposed configurations significantly improves performance and energy-consumption, and so the resulting energy efficiency.

3.3.3. ALU

Figs. 6(a) and 6(b) represent the usable undervoltage range for the ALU MAC benchmark. Since most DL frameworks adopt single-precision floating-point numbers by default, the presented results of the benchmark refer to this specific 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 and

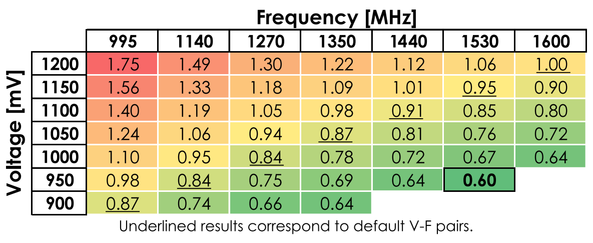


(a) Vega 10.



(b) Radeon 5700 XT.

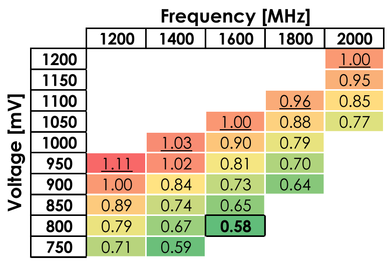
Fig. 4. Core domain - Cache L2 - Normalized energy and performance variations with `OPS`=0. The black dashed line connects the default F-V configurations. The colored arrow lines represent successive F-V operating-points that were evaluated during the undervoltage process.



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

Underlined results correspond to default V-F pairs.

(a) Vega 10.



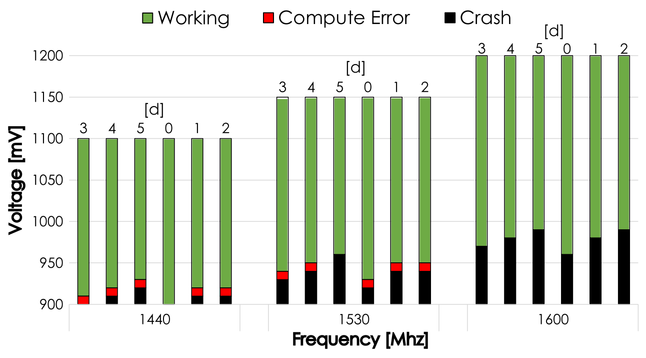
Voltage [mV] \ Frequency [MHz]	1200	1400	1600	1800	2000
1200					<u>1.00</u>
1150					0.95
1100				<u>0.96</u>	0.85
1050			<u>1.00</u>	0.88	0.77
1000		<u>1.03</u>	0.90	0.79	
950	<u>1.11</u>	1.02	0.81	0.70	
900	1.00	0.84	0.73	0.64	
850	0.89	0.74	0.65		
800	0.79	0.67	**0.58**		
750	0.71	0.59			

(b) Radeon 5700 XT.

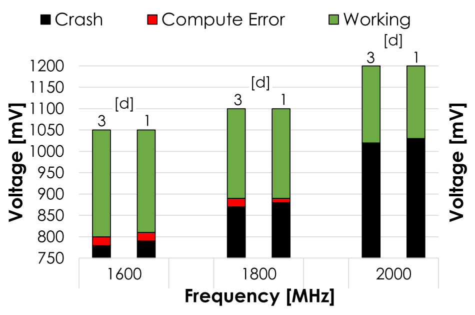
Fig. 5. Core domain - Cache L2 - Obtained normalized Energy-Delay Product (EDP) with `OPS`=0.

the GPU crashes if further undervoltage is applied. It is also observed that the voltage margin increases with the operating frequency, from around 170 mV for 1440 MHz to around 210 mV for 1600 MHz. On the Radeon 5700 XT GPU, the ALU benchmark works correctly independently of the applied voltage for frequencies below 1600 MHz. At higher frequencies, and similarly to what happens with the Vega 10 GPU, a computation error margin exists when undervolting, before the occurrence of crashes. Overall, the Radeon 5700 XT GPU allows more than 200 mV of safe undervoltage across the complete frequency spectrum. It is also observed a small reduction of the undervoltage capabilities for the `d` parameter values that incur in the existence of code dependencies (see Listing 5). However, when compared with the setup with no dependencies, the undervoltage range of the benchmark configuration that represent the general case (`d=3`) is only reduced by 10 mV.

Figs. 7(a) and 7(b) represent the normalized (to the highest core domain F-V configuration) energy consumption and execution time of the MAC benchmark for different F-V pairs. An interesting phenomenon is observed in the energy-execution time plot for the highest frequencies of both GPUs. Performing undervoltage not only reduces energy consumption (as expected), but it

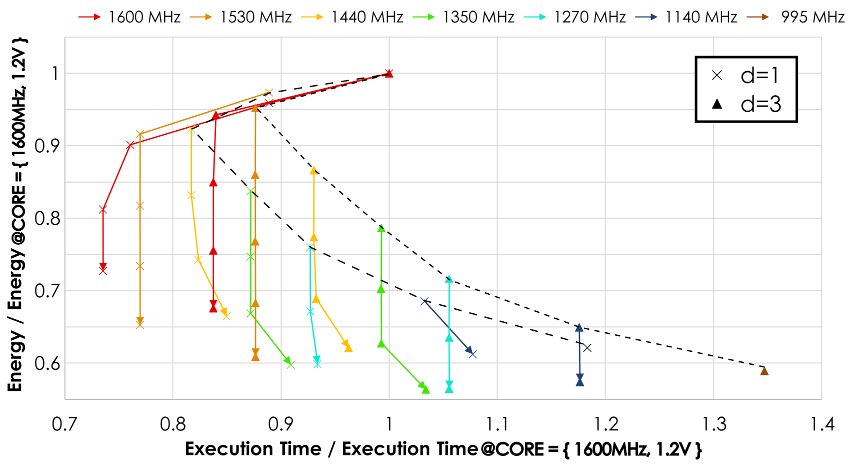


(a) Vega 10.

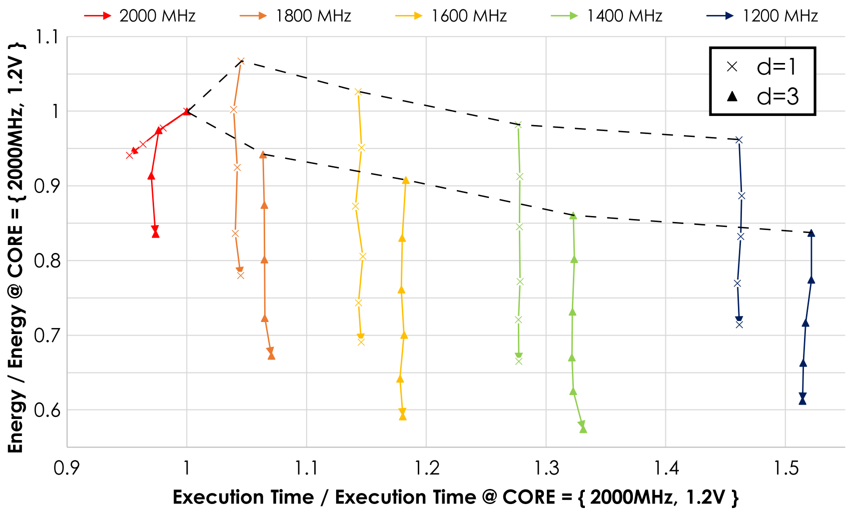


(b) Radeon 5700 XT.

Fig. 6. Core domain - ALU-MAC - Usable voltage margins for the different data types and operands precision.



(a) Vega 10.



(b) Radeon 5700 XT.

Fig. 7. Core domain - ALU-MAC - Normalized energy and performance chart for the benchmark setup with different values of d for single precision floating-point (see Listing 5). The dashed lines connect the results for default F-V configurations.

also allows for a faster overall 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, pacing the execution (i.e., scaling F-V down) until the power is reduced. By undervolting, the power significantly decreases (as $P_{Static} \propto V$ and $P_{Dynamic} \propto V^2$, see [4]), which allows a sustained maintenance of the desired F-V configuration without activating the protection mechanisms.

Finally, Figs. 8(a) and 8(b) present the obtained EDP chart when using the single-precision floating-point data-type. It can be seen that while the Vega 10 GPU favors higher frequencies to achieve the best energy efficiency, the Radeon 5700 XT achieves better results at the frequencies where it is possible to undervolt the most.

d=1

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

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.

(a) Vega 10.

d=1

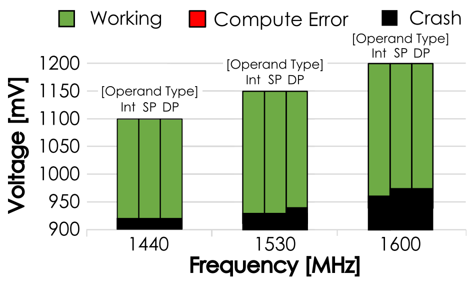
Voltage [mV]	1200	1400	1600	1800	2000
1200					1.00
1150					0.93
1100				0.96	0.85
1050			1.07	0.91	0.82
1000		0.96	0.97	0.87	
950	0.90	0.88	0.92	0.78	
900	0.84	0.83	0.85	0.72	
850	0.80	0.79	0.80		
800	0.75	0.73	0.75		
750	0.71	**0.69**	0.70		

d=3

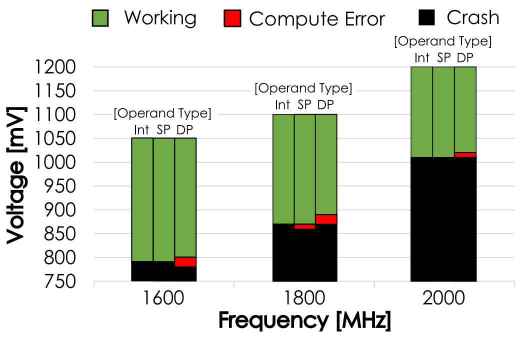
Voltage [mV]	1200	1400	1600	1800	2000
1200					1.00
1150					0.96
1100				1.16	0.93
1050			1.14	1.08	0.88
1000		1.19	1.06	1.00	
950	1.30	1.12	0.98	0.92	
900	1.23	1.04	0.91	0.85	
850	1.14	0.97	0.85	0.80	
800	1.07	0.91	**0.79**		
750	1.01	0.85			

(b) Radeon 5700 XT.

Fig. 8. Core domain - ALU-MAC - Obtained normalized Energy-Delay Product (EDP) for d=1, 3.



(a) Vega 10.



(b) Radeon 5700 XT.

Fig. 9. Core domain - Reduction benchmark - Usable voltage level for each frequency configuration with varying operand type: Int - 32-bit Integer, SP - Single-Precision Floating-Point, DP - Double-Precision Floating-Point.

3.3.4. Reduction

Figs. 9(a) and 9(b) present the usable voltage range for the *reduction* benchmark. As it can be seen, due to the high pressure exercised on the L2 cache by this benchmark, the minimum usable voltage coincides with the one already measured for the Cache benchmark. The ALU benchmark can explain the compute errors range for the several data types, as well as the V_{min} value differences, with double-precision floating-point having a greater margin of computational errors (on the Radeon 5700 XT) and an overall reduced voltage guardband at the higher frequencies on both GPUs.

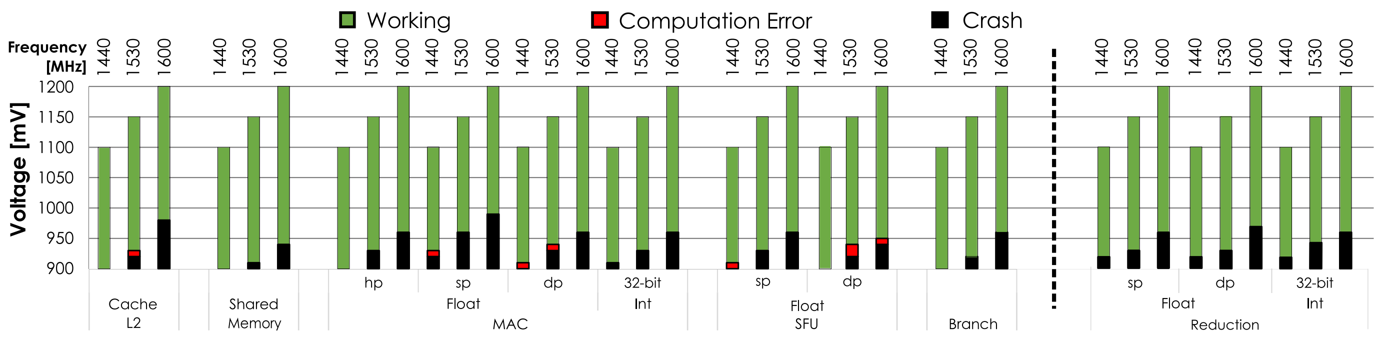
3.3.5. General comments and remarks

Fig. 10 presents a comprehensive comparison of the valid voltage ranges for all the considered architectural components of both GPUs. The general observation is that the Cache L2 and the ALU are the two components that tend to compromise the undervoltage capabilities. Cache L2 affects the kernels that are more memory intensive, while the ALU limits those that are more compute-intensive, either with linear or special/non-linear operations.

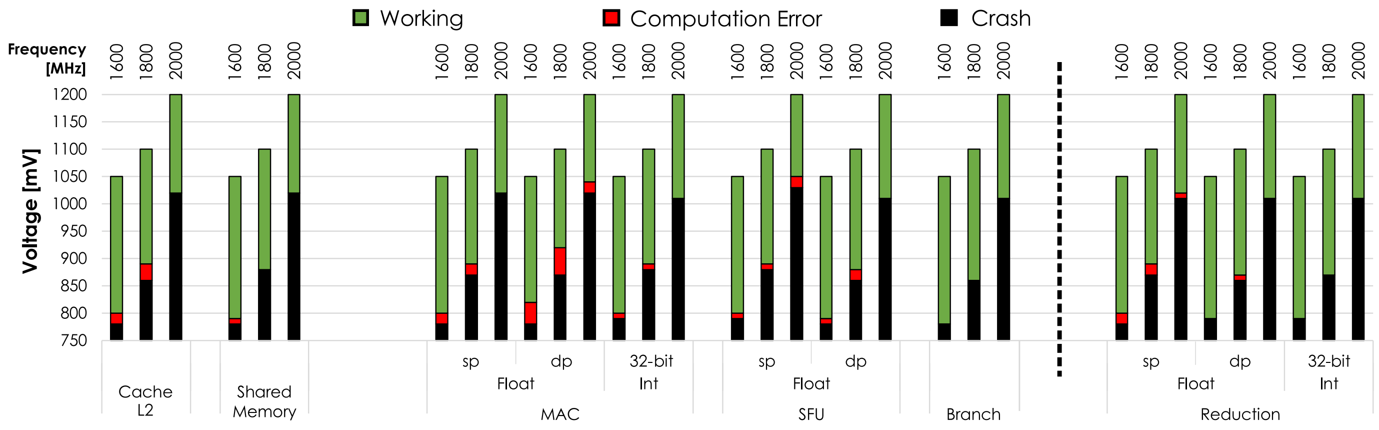
In more detail, the results of both benchmarks that test the ALU are similar. This allows concluding one of two things: either there are two similar critical paths (in terms of timing constraints) on both the linear and non-linear data paths; or the critical path is at the beginning of the ALU, where the operands are forwarded to the appropriate computational unit. Although it is not possible to assess which preposition is the correct one, the second explanation tends to be more credible. These conclusions will allow to understand and predict the behavior of more elaborate DL applications (see section 4).

Fig. 10 also presents the corresponding results for branch instructions, which allow to conclude that branch miss-prediction does not negatively impact the minimum voltage.

In general, even though two completely different GPU architectures are under evaluation, the general behavior of both is similar. At the lower frequencies, the minimum voltage can be safely applied without the introduction of computation errors or crashes. The utilization of undervoltage for higher frequencies will depend on which components the application being executed stresses the most. Across all frequencies, both GPUs allow for more than 20% of safe undervoltage, being a considerable amount to be explored in the following section.



(a) Vega 10.



(b) Radeon 5700 XT.

Fig. 10. Comparison of usable GPU core voltage ranges for all the considered architectural components of the GPU.

3.4. Temperature model

The ALU and Cache L2 benchmarks were tested at a set temperature of 45 °C, by fixing the GPU fan to 100% and waiting for the temperature to stabilize between runs. This specific temperature was chosen by observing that, while executing short benchmarks, the GPU's temperature stabilized around this value. However, during the execution of longer applications (or when changing the environmental conditions), the device can become hotter.

The work of Leng et al. [10] pointed out that only a small variation on V_{min} is observed due to temperature variations. However, the performed experiments only covered temperature levels up to 70 °C, easily surpassed by the GPU under use.

To access the undervoltage capabilities in broader temperature ranges, the benchmarks were continuously executed on both GPUs by varying the GPU fan speed, while analyzing the execution output. Varying the frequency did not change the temperature behavior, so the results of all executions were combined in Fig. 11, where only the results corresponding to voltage variations were represented.

Hence, changing the amount of undervolt or fan speed resulted in the same three different output scenarios that were previously observed:

- *Working* - the benchmark's output was correct and was the same as when running with conventional F-V pairs;
- *Computation Errors* - the benchmark's output was not correct; however, the GPU was still working and responding to the kernel commands;
- *Crash* - the GPU stopped working and responding to the commands of the application.

Overall, the undervoltage capabilities stay relatively the same until the 70 °C to 75 °C temperature, with the highest undervoltage capabilities being achieved at the 55 °C mark. After the 75 °C mark, and by following Freijado's work [3], the carriers' mobility decreases and starts limiting the undervolting capabilities of the CMOS circuit. This result led to the definition of a simple temperature model that limits the undervoltage potential for temperatures above 70 °C, as indicated in Fig. 11. This model acts as a fail-safe condition that guarantees that the non-conventional F-V exploration performed at 45 °C (as described until now) can be safely used across the complete temperature spectrum. Hence, the end-user will have to limit the applied percentage of undervoltage according to this model, depending on the current GPU temperature. By doing so, it guarantees that setting a safe non-conventional F-V pair will not cause a GPU crash as a consequence of the temperature rise.

4. CNN layer characterization with independent voltage and frequency scaling

As it was referred in Section 2, Tang [22] 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 decoupled undervoltage techniques, a broader range of F-V configurations is herein envisaged to provide even greater benefits.

In fact, an important characteristic of DNNs is their tolerance to a certain degree of computation errors [26], without any significant

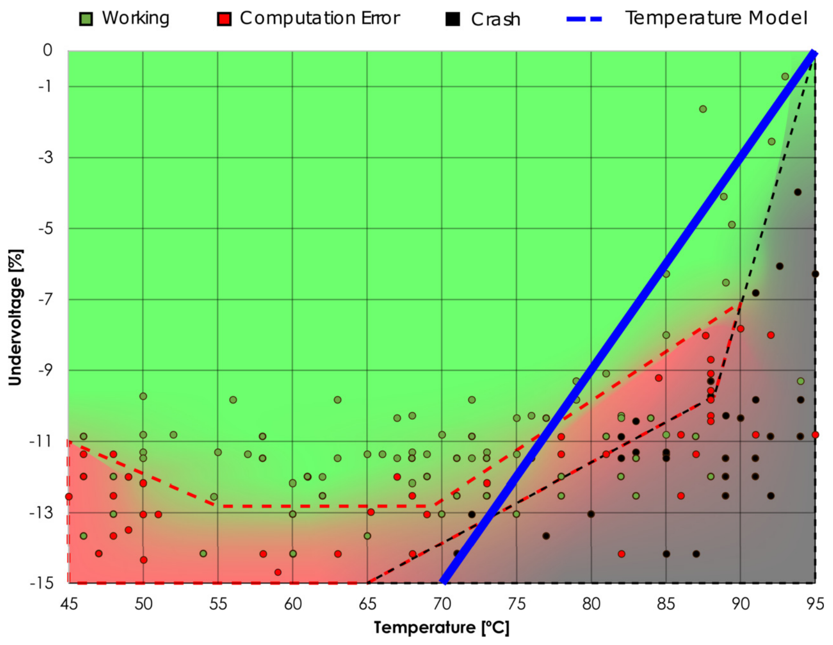


Fig. 11. Undervoltage capabilities with changing temperature conditions and the defined temperature model.

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.

On the other hand, depending on the field of application and the type of data being analyzed, different DNN architectures are commonly used to better tailor the learning capabilities of the network to the execution scenario. Two of the most common architectures are convolutional neural networks and transformers neural networks. Accordingly, these architectures are often used to tackle problems in computer vision and natural language processing (NLP).

With this observation in mind, the presented study was focused on convolutional and fully connected layers. These layers represent the two fundamental layers used by convolutional neural networks to detect patterns in images. The same layers are also present as the building blocks of the state-of-the-art natural language models using transformer architectures [24]. In this case, the layers are used to perform the analysis of sequential inputs of data. Convolutional and fully connected layers are also the only neural network layers supported and optimized by the GPU manufacturer (AMD) in their GPGPU mathematical libraries.

Looking at the particular case of the Convolutional Neural Network (CNN), its feature extraction ability is mainly 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 [11], which makes them particularly suited to exploit energy saving mechanisms.

Hence, to extend the presented benchmarking results to 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 on the Vega 10 GPU.

4.1. Convolution layer

The MIOpen library[5] 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 it is 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,[6] in order to understand how each algorithm is affected by DVFS, both in its inference and training phases.

Fig. 12 presents the set of valid voltage ranges that were obtained for the convolution layer in the inference and training phases. 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) were detected at lower voltage levels. 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 [23]. 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.

Moreover, 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 simplest of the three, requiring no data transformation and movement before its execution. In contrast, *GEMM* and *Winograd* require a pre-processing step, resulting in the activation of more GPU components, and making these algorithms more prone to GPU voltage drops.

When comparing the training and inference phases, it is observed that they present similar undervoltage capabilities (for all

[5] github.com/ROCmSoftwarePlatform/MIOpen.
[6] github.com/baidu-research/DeepBench.

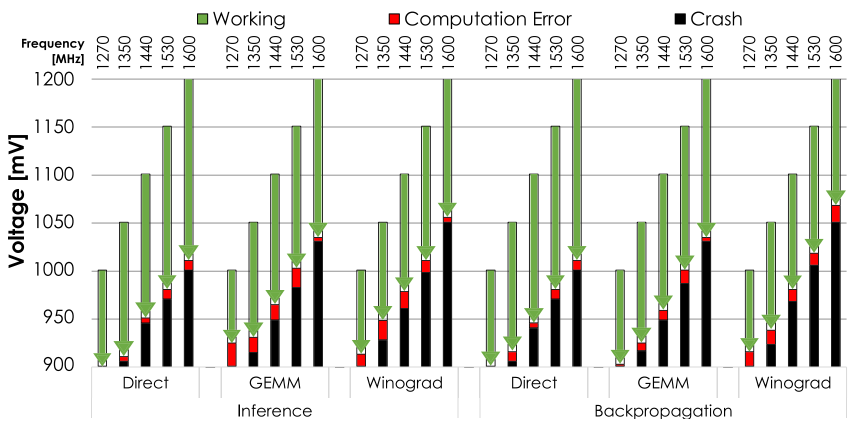


Fig. 12. Convolution Layer - Usable voltage range when applying F-V scaling to the GPU core.

algorithms), with the crash point diverging only around 10 mV. However, the inference algorithm is more prone to the introduction of computation errors, as can be seen by the bigger size of the red bars in Fig. 12.

Figs. 13(a) and 13(b) illustrate the impact of decoupled F-V scaling on the energy consumption, execution time and EDP. When working at each frequency default voltage level (black 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 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. 14 indicate that the most energy efficient configuration for the three algorithms is observed 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.

4.2. Fully connected layer

The RocBlas library[7] provides a single API for matrix multiplication - the underlying mathematical operation of the fully connected layer. By analyzing the performance counters and the kernels called by this library, it is possible to understand that multiplication is performed in one of two ways, depending on the size of the matrices. According to [12], 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 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 3.3.3). 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 3.3.2.

Fig. 15 illustrates the results of the conducted experiments and confirm the prediction: for small matrix sizes it is possible to perform a higher degree of undervoltage. Furthermore, the results depicted in the normalized energy-performance chart (see Fig. 16) and EDP heatmap (see Fig. 17) show that 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% in execution time.

4.3. Error analysis

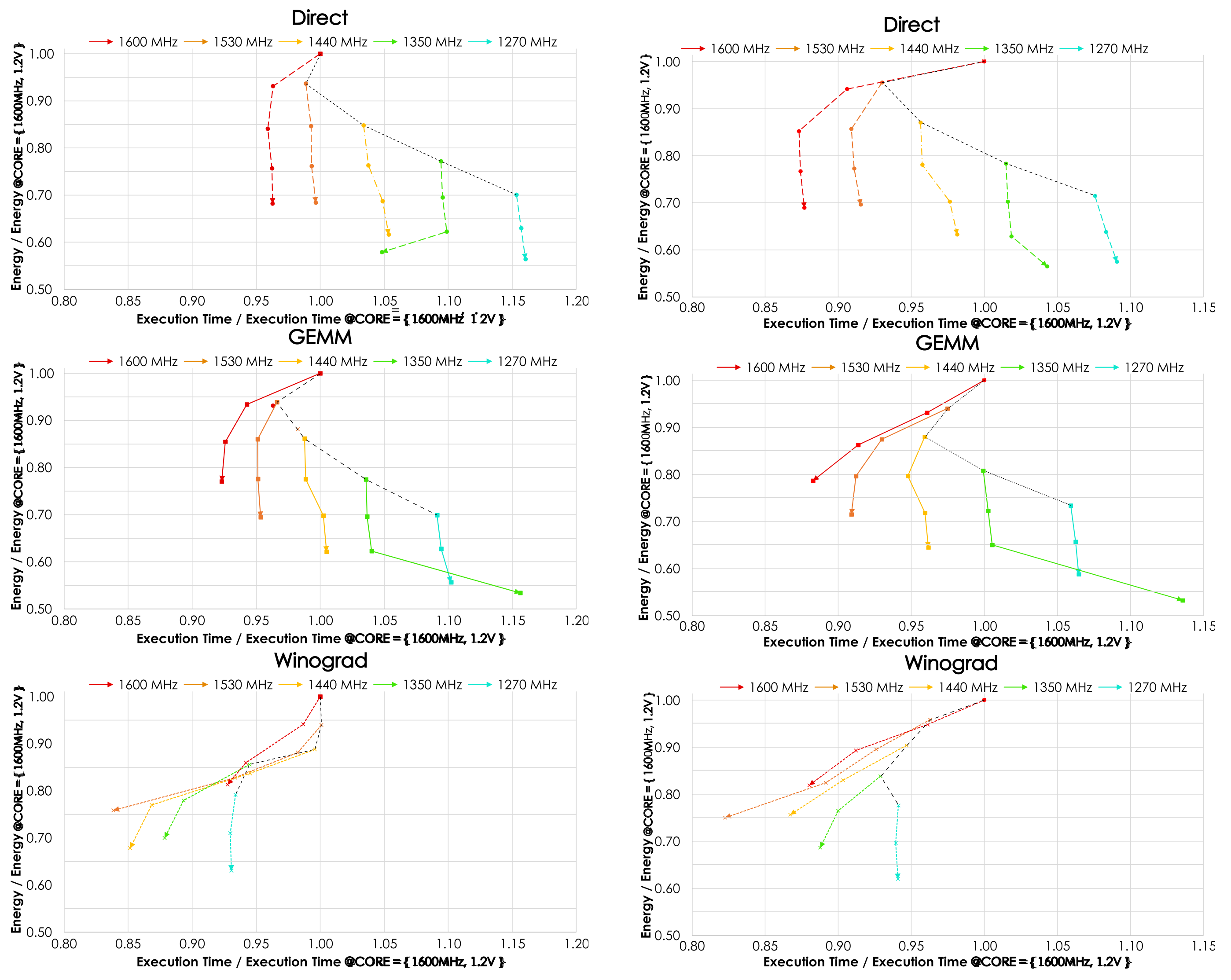
To evaluate the eventual occurrence of computation errors due to the utilization of decoupled F-V scaling, each benchmark was executed both with the default "automatic" parameterization and with the F-V pair under testing, with the same input data. A *warmup_kernel* was also executed before each of these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

For the architecture characterization benchmarks (see Section 3), a computation error was asserted whenever any of the output vectors differs between the executions. For the CNN layers characterization (see Section 4), a different error metric was adopted due to the utilization of software libraries (versus custom kernels) operating over floating-point 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 the 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}.

4.3.1. Convolution layer

Fig. 18 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 magnitude.

[7] github.com/ROCmSoftwarePlatform/rocBLAS.



(a) Inference (b) Training

Fig. 13. Convolution Layer - Normalized energy and performance chart when applying F-V scaling to the GPU core during inference and training phases.

Inference	Direct					GEMM					Winograd				
	Frequency [MHz]					Frequency [MHz]					Frequency [MHz]				
Voltage [mV]	1270	1350	1440	1530	1600	1270	1350	1440	1530	1600	1270	1350	1440	1530	1600
1200	1.21	1.14	1.09	1.02	1.00	1.15	1.13	1.04	1.02	1.00	1.14	1.10	1.05	1.02	1.00
1150	1.09	1.03	0.97	0.93	0.90	1.05	1.02	0.94	0.91	0.88	1.05	1.02	0.97	0.94	0.93
1100	0.99	0.93	0.88	0.84	0.81	0.94	0.90	0.85	0.82	0.79	0.95	0.92	0.88	0.87	0.81
1050	0.90	0.84	0.79	0.76	0.73	0.85	0.80	0.77	0.74	0.71	0.84	0.81	0.79	0.77	0.75
1000	0.81	0.76	0.72	0.68	0.66	0.76	0.72	0.70	0.66		0.74	0.70	0.67	0.64	
950	0.73	0.68	0.65			0.69	0.65	0.62			0.66	0.61	**0.58**		
900	0.65	**0.61**				**0.61**	0.62				0.59				

Training	Frequency [MHz]					Frequency [MHz]					Frequency [MHz]				
Voltage [mV]	1270	1350	1440	1530	1600	1270	1350	1440	1530	1600	1270	1350	1440	1530	1600
1200	1.15	1.09	1.03	1.02	1.00	1.16	1.11	1.06	1.02	1.00	1.13	1.09	1.05	1.01	1.00
1150	1.05	0.98	0.92	0.89	0.85	1.05	0.99	0.95	0.92	0.89	1.03	0.99	0.95	0.92	0.91
1100	0.94	0.88	0.83	0.78	0.74	0.95	0.90	0.84	0.81	0.79	0.92	0.89	0.86	0.83	0.81
1050	0.85	0.79	0.75	0.70	0.67	0.86	0.81	0.75	0.73	0.69	0.82	0.78	0.75	0.74	0.72
1000	0.77	0.71	0.69	0.64	0.60	0.78	0.72	0.69	0.65		0.73	0.69	0.66	0.62	
950	0.69	0.64	0.62			0.70	0.65	0.62			0.65	0.61			
900	0.63	**0.59**				0.63	**0.61**				**0.58**				

Underlined results correspond to default V-F pairs.

Fig. 14. Convolution Layer - Energy Delay Product (EDP) when applying F-V scaling to the GPU core during inference (top) and training (bottom) phases.

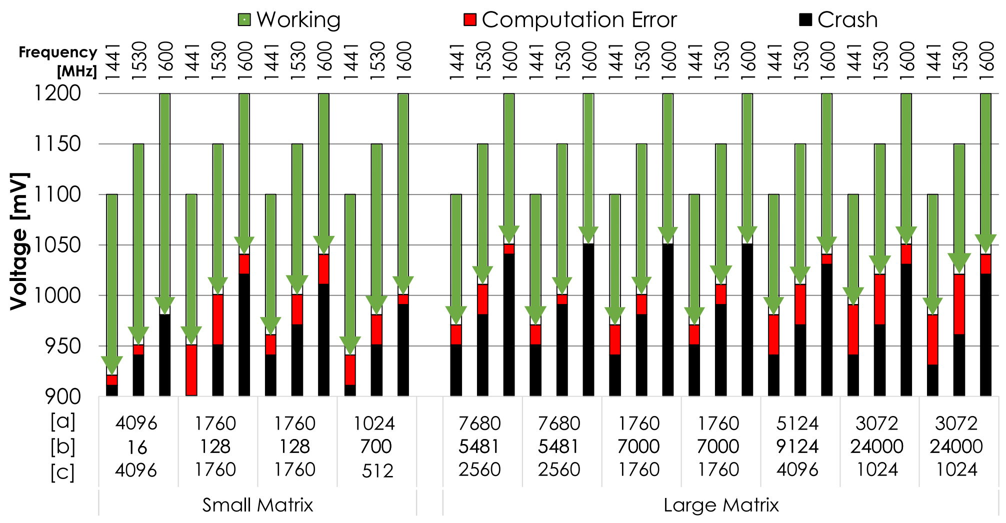


Fig. 15. Fully Connected Layer - Usable GPU core voltage range. Values represent matrix sizes (example $A_{a\times b} \cdot B_{b\times c}$).

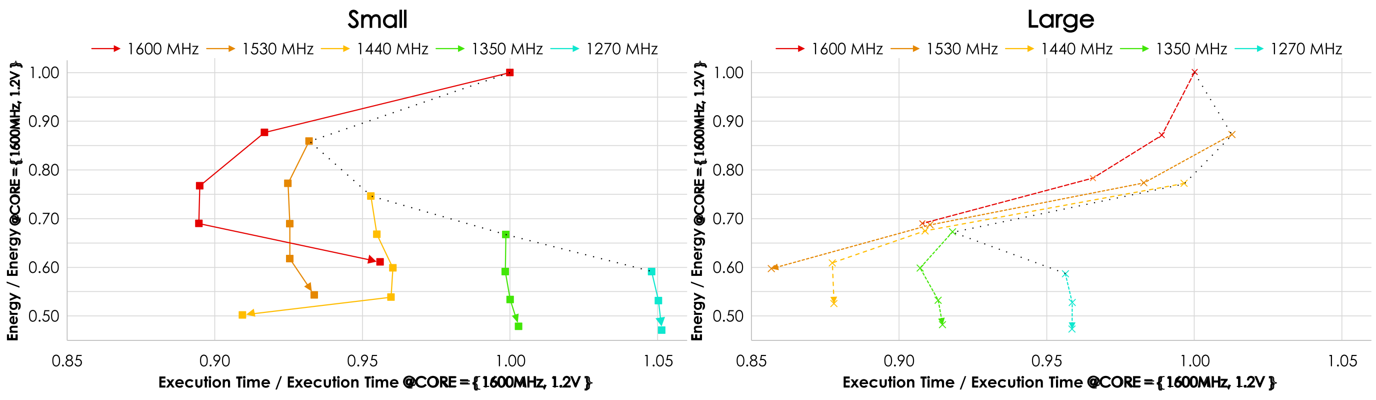


Fig. 16. Fully Connected Layer - Normalized energy and performance chart for small and large matrices, when applying F-V scaling to the GPU core (black dashed lines connect the results for default F-V configurations).

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**			

Underlined results correspond to default V-F pairs.

Fig. 17. Fully Connected Layer - Energy-Delay Product heat-map for small (left) and large (right) matrices when applying F-V scaling to the GPU core.

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.

4.3.2. *Fully connected*

Fig. 19 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).

4.4. *Complete CNN training with non-conventional F-V*

The previously obtained results evidence that the small number and the insignificant amplitude of the computational errors introduced by 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. For this evaluation, four well-known CNN models (AlexNet, LeNet, VGG11 and WideResNet) were trained with decoupled F-V configurations. For each case, the previously performed error analyses were applied to identify the best F-V configurations and to assess the induced errors. In more detail, Fig. 20 presents the behavior of the loss and model accuracy of each considered network, measured with the test dataset over the training session on all the tested F-V pairs. It is possible to observe that the two metrics' progress is, within small variations, the same for all tested F-V configurations. Such observation prompted the conclusion about the validation of the results obtained in the previous subsection.

Table 3 presents the obtained median classification accuracy over ten runs. From these results, it can be observed 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. Table 4 presents the identified best configurations considering the maximum undervoltage at the highest frequency, and the F-V configurations that minimize the EDP.

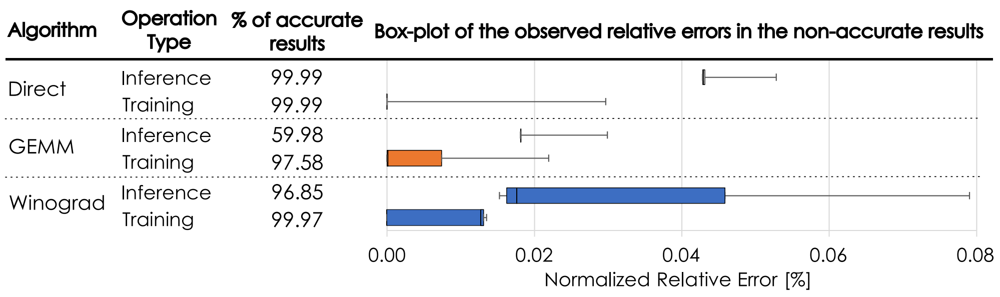


Fig. 18. Convolution Layer - Average percentage of accurate results and relative error distribution of non-accurate outputs for the minimum usable core voltage across all considered core frequency values.

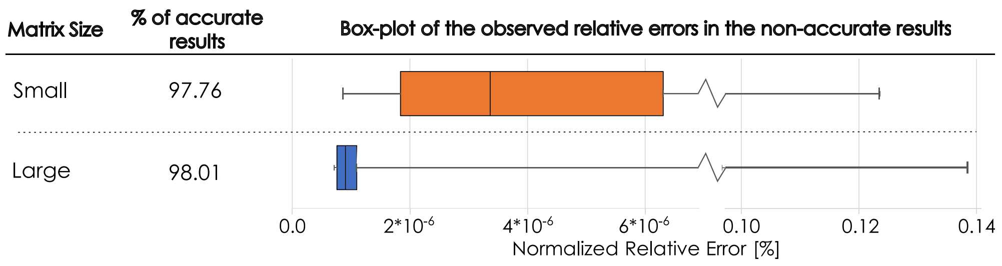


Fig. 19. Fully Connected - Average percentage of accurate results and relative error distribution of non-accurate outputs for the minimum usable core voltage across all considered core frequency values.

Table 3
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

Table 4
Non-conventional F-V settings per CNN model. Percentage [%] column represents energy improvement vs default F-V setup.

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	24	1600 - 1.05	20	1600 - 1.1	22	1600 - 1.05	23
Proposed @ best EDP	1530 - 1.0	38	1270 - 1.0	38	1440 - 1.0	33	1530 - 1.0	38

[*] DVFS setup that optimizes EDP using manufacturer voltage values.

Figs. 21 and 22 depict the energy performance charts and EDP results for the conducted F-V exploration. It is particularly worth noting the comparison of the results corresponding to the automatic DVFS system (represented as a black dot in Fig. 21) versus the non-conventional F-V configurations. In this experiment, the training procedure was executed while allowing the DVFS system to automatically vary the current F-V pair and adjust all the corresponding parameters. In neither of the four tested models did the automatic system achieve the best performance or energy consumption, demonstrating the potential for decoupled F-V configurations.

In particular, the default F-V pairs are able to produce either the lowest energy consumption or the highest performance. Hence, the main benefit of exploring the proposed non-conventional F-V is the possibility to attain GPU operating-points corresponding to higher or even the highest frequency level (maximizing performance), while having an energy consumption similar to a default low-frequency configuration.

Finally, Table 5 emphasizes and summarizes the main achievements of this research, by comparing the CNN execution at the default F-V setup (with frequency scaling using default voltage values) with the proposed approach. It considers two operating conditions: (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 all three metrics (energy, training time, and EDP) without compromising the resulting accuracy of the whole CNN. Hence, the main benefit of this approach is to allow the utilization of higher or even the highest frequency (maximizing performance) while having a similar energy consumption to that of lower frequency setups.

4.5. Decoupled voltage-frequency scaling in other domains

The present study was only focused on neural networks as the target application since this domain respects a set of necessary heuristics that could guarantee more considerable success before

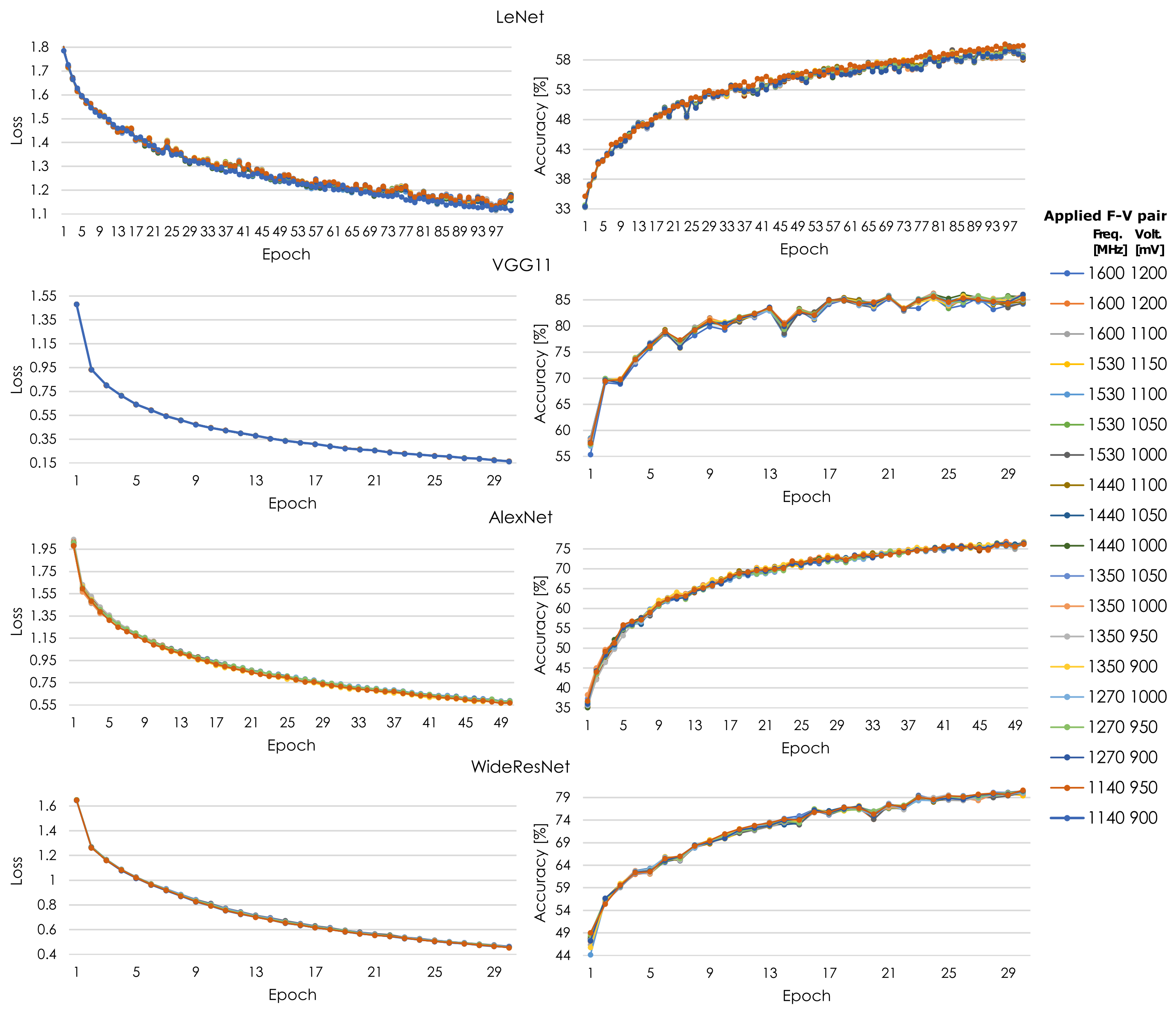


Fig. 20. Core domain - CNN models - superposition of the obtained loss and model accuracy for all considered F-V configurations.

Table 5
Evaluation of performance, energy, and EDP when applying non-conventional DVFS in the training of neural networks.

Metric	Selected configuration	Improvement *vs* default F-V			
		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 F-V 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 F-V configuration of the GPU.

the study. The application uses an iterative process to solve the problem that converges to the final solution. Moreover, the algorithm can tolerate small percentages of errors without significantly affecting the final result of the computation.

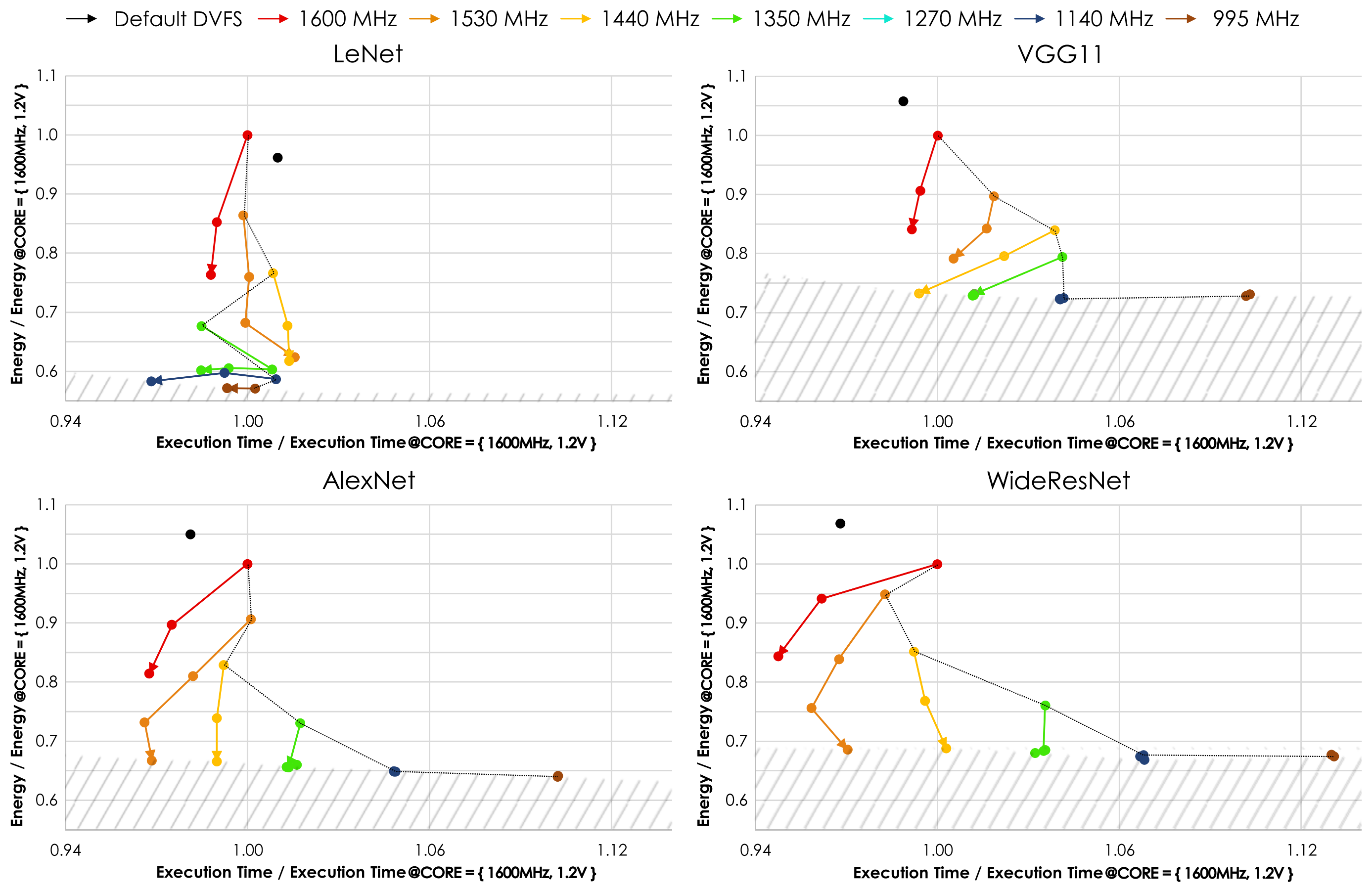


Fig. 21. Core domain - CNN models - Normalized energy consumption and execution time for training + inference. The dashed line connects the default F-V pairs, and the diagonal striped pattern indicates the plateau of minimum energy consumption.

LeNet

Voltage [mV] \ Frequency [MHz]	1140	1270	1350	1440	1530	1600
1200						1.00
1150					0.86	0.84
1100				0.77	0.76	0.75
1050			0.67	0.69	0.68	
1000		**0.56**	0.61	0.63	0.63	
950	0.57	0.59	0.60			
900	0.57	0.59	0.59			

VGG11

Voltage [mV] \ Frequency [MHz]	1140	1270	1350	1440	1530	1600
1200						1.00
1150					0.91	0.90
1100				0.87	0.86	0.83
1050			0.83	0.81	0.80	
1000		0.75	0.74	**0.73**		
950	0.81	0.75	0.74			
900	0.80	0.75	0.74			

AlexNet

Voltage [mV] \ Frequency [MHz]	1140	1270	1350	1440	1530	1600
1200						1.00
1150					0.91	0.87
1100				0.82	0.80	0.79
1050			0.74	0.73	0.71	
1000		0.68	0.66	0.66	**0.65**	
950	0.71	0.68	0.67			
900	0.71	0.68	0.66			

WideResNet

Voltage [mV] \ Frequency [MHz]	1140	1270	1350	1440	1530	1600
1200						1.00
1150					0.93	0.91
1100				0.84	0.81	0.80
1050			0.79	0.77	0.72	
1000		0.72	0.71	0.69	**0.67**	
950	0.77	0.72	0.71			
900	0.76	0.71	0.70			

Underlined results correspond to default V-F pairs.

Fig. 22. Core domain - CNN models - Obtained normalized Energy-Delay Product (EDP) for training + inference.

As such, any other GPGPU application domain that also respects those two characteristics might also be an excellent candidate to benefit from decoupled voltage and frequency scaling.

5. 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 on the tested GPUs. It was also observed that it is safe to undervolt the considered GPUs between 15% and 25% 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 network accuracy. The obtained results also indicate that it is possible to improve the GPU EDP (by an average of 36.7%) while training complete CNN models. Overall, this paper shows how to characterize the sensitivity to undervoltage of a given GPU architecture, in order to reduce the GPU energy consumption without degrading the attained results.

CRediT authorship contribution statement

Conception and design of study: F. Mendes, P. Tomás, N. Roma; acquisition of data: F. Mendes; analysis and/or interpretation of data: F. Mendes.

Drafting the manuscript: F. Mendes; revising the manuscript critically for important intellectual content: P. Tomás, N. Roma.

Approval of the version of the manuscript to be published: P. Tomás, N. Roma.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

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

References

[1] K. Fan, B. Cosenza, B. Juurlink, Predictable GPUs frequency scaling for energy and performance, in: Proceedings of the 48th International Conference on Parallel Processing, ICPP 2019, Association for Computing Machinery, Kyoto, Japan, 2019, pp. 1–10.

[2] K. Fan, B. Cosenza, B. Juurlink, Accurate energy and performance prediction for frequency-scaled GPU kernels, Computation 8 (2020) 37.

[3] J.F. Freijedo, J. Semião, J.J. Rodriguez-Andina, F. Vargas, I.C. Teixeira, J.P. Teixeira, Modeling the effect of process, power-supply voltage and temperature variations on the timing response of nanometer digital circuits, J. Electron. Test. 28 (2012) 421–434, https://doi.org/10.1007/s10836-012-5297-0.

[4] J. Guerreiro, A. Ilic, N. Roma, 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.

[5] J. Guerreiro, A. Ilic, N. Roma, P. Tomás, GPU static modeling using PTX and deep structured learning, IEEE Access 7 (2019) 159150–159161, https://doi.org/10.1109/ACCESS.2019.2951218.

[6] J. Guerreiro, A. Ilic, N. Roma, P. Tomás, Modeling and decoupling the GPU power consumption for cross-domain DVFS, IEEE Trans. Parallel Distrib. Syst. 30 (2019) 2494–2506, https://doi.org/10.1109/TPDS.2019.2917181.

[7] J. Guerreiro, A. Ilic, N. Roma, P. Tomás, DVFS-aware application classification to improve GPGPUs energy efficiency, Parallel Comput. 83 (2019) 93–117, https://doi.org/10.1016/j.parco.2018.02.001.

[8] C. Kalogirou, P. Koutsovasilis, C.D. Antonopoulos, N. Bellas, S. Lalis, S. Venugopal, 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), 2019, pp. 302–311.

[9] Y. Kim, R. Daly, J. Kim, C. Fallin, J.H. Lee, D. Lee, C. Wilkerson, K. Lai, O. Mutlu, Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors, in: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), IEEE, Minneapolis, MN, USA, 2014, pp. 361–372.

[10] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, V.J. Reddi, Safe limits on voltage reduction efficiency in GPUs: a direct measurement approach, in: MICRO, 2015, pp. 294–307, ISSN: 2379-3155.

[11] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs, in: BDCloud-SocialCom-SustainCom, 2016, pp. 477–484.

[12] I. Masliah, A. Abdelfattah, A. Haidar, S. Tomov, M. Baboulin, J. Falcou, J. Dongarra, High-performance matrix-matrix multiplications of very small matrices, in: P.-F. Dutot, D. Trystram (Eds.), Euro-Par 2016: Parallel Processing, in: Series Title: Lecture Notes in Computer Science., vol. 9833, Springer International Publishing, Cham, 2016, pp. 659–671.

[13] S. Mittal, S. Vaishay, A survey of techniques for optimizing deep learning on GPUs, J. Syst. Archit. 99 (2019) 101635, https://doi.org/10.1016/j.sysarc.2019.101635, original Publisher: Elsevier.

[14] S.M. Nabavinejad, H. Hafez-Kolahi, S. Reda, Coordinated DVFS and precision control for deep neural networks, IEEE Comput. Archit. Lett. 18 (2019) 136–140, https://doi.org/10.1109/LCA.2019.2942020.

[15] F. Nakhaee, M. Kamal, A. Afzali-Kusha, M. Pedram, S.M. Fakhraie, H. Dorosti, Lifetime improvement by exploiting aggressive voltage scaling during runtime of error-resilient applications, Integration 61 (2018) 29–38, https://doi.org/10.1016/j.vlsi.2017.10.013.

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

[17] G. Raposo, P. Tomás, N. Roma, PositNN: training deep neural networks with mixed low-precision posit, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2021), 2021, pp. 7908–7912.

[18] B. Salami, O. Unsal, A. Cristal, Fault characterization through FPGA undervolting, in: 2018 28th International Conference on Field Programmable Logic and Applications (FPL), 2018, pp. 85–853, ISSN: 1946-1488.

[19] B. Salami, O.S. Unsal, A. Cristal Kestelman, Comprehensive evaluation of supply voltage underscaling in FPGA on-chip memories, in: 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2018, pp. 724–736.

[20] A. Shrestha, A. Mahmood, Review of deep learning algorithms and architectures, IEEE Access 7 (2019) 53040–53065, https://doi.org/10.1109/ACCESS.2019.2912200.

[21] J. Tan, S.L. Song, K. Yan, X. Fu, A. Marquez, D. Kerbyson, Combating the reliability challenge of GPU register file at low supply voltage, in: PACT, ACM Press, Haifa, Israel, 2016, pp. 3–15.

[22] Z. Tang, Y. Wang, Q. Wang, X. Chu, The impact of GPU DVFS on the energy and performance of deep learning: an empirical study, arXiv:1905.11012 [cs], 2019, http://arxiv.org/abs/1905.11012, arXiv:1905.11012.

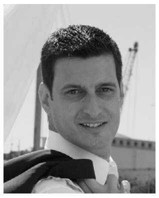
[23] R. Thomas, K. Barber, N. Sedaghati, L. Zhou, R. Teodorescu, Core tunneling: variation-aware voltage noise mitigation in GPUs, in: HPCA, 2016, pp. 151–162, ISSN: 2378-203X.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[25] Q. Wang, 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.

[26] Q. Zhang, T. Wang, Y. Tian, F. Yuan, Q. Xu, ApproxANN: an approximate computing framework for artificial neural network, in: DATE, 2015, pp. 701–706, ISSN: 1558-1101.

Francisco Mendes received his M.Sc. in Electrical and Computer Engineering (ECE) from Instituto Superior Técnico (IST), University of Lisbon (UL), Portugal, in 2020. He developed this work while a Researcher at Instituto de Engenharia de Sistemas e Computadores R&D (INESC-ID). His research interests include computer architectures and high-performance computing.



Pedro Tomás received the Ph.D. in Electrical and Computer Engineering (ECE) from Instituto Superior Técnico (IST), Technical University of Lisbon, Portugal, in 2009. He is an associate professor in the Dept. of ECE, IST, and a senior researcher of the High Performance Computing Architectures and Systems (HPCAS) group of Instituto de Engenharia de Sistemas e Computadores R&D (INESC-ID). His research activities include computer architectures, high-performance and energy-efficient computing and signal processing systems. He is a member of the IEEE Computer and Signal Processing Societies and has contributed to more than 70 papers to international peer-reviewed journals and conferences.

Nuno Roma received the Ph.D. degree in Electrical and Computer Engineering (ECE) from Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Portugal, in 2008. Currently, he is an Associate Professor with the Dept. of ECE of IST and he is a Senior Researcher of the High Performance Computing Architectures and Systems (HPCAS) group of Instituto de Engenharia de Sistemas e Computadores R&D (INESC-ID) - a not for profit R&D institute affiliated with IST. His research interests include computer architectures, specialized and dedicated structures for digital signal processing, energy-aware computing, parallel processing and high-performance computing systems. He contributed to more than 120 manuscripts to journals and international conferences and served as a Guest Editor of Springer Journal of Real-Time Image Processing (JRTIP) and of EURASIP Journal on Embedded Systems (JES). He has also acted as the organizing chair of several workshops and special sessions. He has a consolidated experience on funded research projects leadership and he is member of several research Networks of Excellence (NoE), including HiPEAC (European Network of Excellence on High Performance and Embedded Architecture and Compilation). Dr. Roma is a Senior Member of both IEEE and ACM.