DOI: 10.14529/jsfi210103

Size & Shape Matters: The Need of HPC Benchmarks of High Resolution Image Training for Deep Learning

Ferran Parés Pont[1], *Pedro Megias*[1], *Dario Garcia-Gasulla*[1], *Marta Garcia-Gasulla*[1], *Eduard Ayguadé*[1,2], *Jesús Labarta*[1,2]

One of the purposes of HPC benchmarks is to identify limitations and bottlenecks in hardware. This functionality is particularly influential when assessing performance on emerging tasks, the nature and requirements of which may not yet be fully understood. In this setting, a proper benchmark can steer the design of next generation hardware by properly identifying said requirements, and quicken the deployment of novel solutions. With the increasing popularity of deep learning workloads, benchmarks for this family of tasks have been gaining popularity. Particularly for image based tasks, which rely on the most well established family of deep learning models: Convolutional Neural Networks. Significantly, most benchmarks for CNN use low-resolution and fixed-shape (LR&FS) images. While this sort of inputs have been very successful for certain purposes, they are insufficient for some domains of special interest (*e.g.*, medical image diagnosis or autonomous driving) where one requires higher resolutions and variable-shape (HR&VS) images to avoid loss of information and deformation. As of today, it is still unclear how does image resolution and shape variability affect the nature of the problem from a computational perspective. In this paper we assess the differences between training with LR&FS and HR&VS, as means to justify the importance of building benchmarks specific for the latter. Our results on three different HPC clusters show significant variations in time, resources and memory management, highlighting the differences between LR&FS and HR&VS image deep learning.

Keywords: deep learning, convolutional neural networks, high-resolution images, variable-shape images, HPC benchmarks.

Introduction

Nowadays, on the race for exascale, the leading architectures panorama is as heterogeneous as ever. Looking at the Top500 list (November 2020), we find five different architectures in the first six positions of the list. At the same time, the software is more heterogeneous than ever as almost all scientific fields use high-performance environments. This heterogeneity entails significant differences in terms of computing needs and patterns, software's maturity, and data demands. And motivates an individualized analysis for all applications of interest.

In this scenario, HPC benchmarks are of paramount importance. On the one hand, to set a common ground to evaluate different architectures and systems. On the other hand, to provide tools for users or developers to understand the most suited architecture for their specific needs.

With the recent rise of Artificial Intelligence (AI) applications, particularly through Deep Neural Network models (DNNs), benchmarks for this family of tasks have gained relevance. Meanwhile, the establishment of Convolutional Neural Networks (CNNs) as the de facto solution for most image related tasks has turned them into the spearhead of DNNs models for benchmarking purposes.

A common (but not always good) practice when training DNN models for images is to down-sample their size before computation. AI practitioners typically do this to reduce the memory requirements and training time of the model. Unfortunately, such down-sampling process entails deformation of visual patterns and generalizes information loss. While many applications are

[1]Barcelona Supercomputing Center (BSC), Barcelona, Spain
[2]Universitat Politecnica de Catalunya (UPC), Barcelona, Spain

not particularly affected by these drawbacks because they do not rely on details of the input (*e.g.*, telling cats from dogs), on other key domains where images are naturally of high-resolution and variable shape (HR&VS) down-sampling can entail a significant performance reductions. This includes strategic and critical applications like autonomous driving [8, 30], satellite data analysis [28], and medical diagnosis [12, 21].

The popularity of image-related tasks in AI, that can be solved successfully by down-sampling the data, has motivated the definition of several HPC benchmarks for low-resolution (LR) images. As the AI community looks for new challenges, the applications which benefit from HR&VS properties are gaining attention. This is highlighting the limitations of current systems and the need for HR&VS specific benchmarks, influencing the design of the next generation of hardware.

High-resolution (HR) data entails large memory requirements, which limits the amount of images that can be processed together (*i.e.*, in the same batch). Batch size limitations have a significant effect on the efficiency of the computation, while its impact on model performance remains under study by the AI community. At the same time, current accelerators (*e.g.*, GPUs, TPUs) are rather limited in terms of memory capacity, although workarounds to load larger memories than the one offered by the device have already been proposed (as discussed in Section 1). These workarounds include model parallelism [3, 10], activations re-computation [7] and offloading [6], enabling greater memory loads at the cost of computation efficiency. In this high-memory load context, avoiding accelerators and using CPU computation must be considered as a feasible alternative. The two main arguments in favour of CPUs being the lack of added components and functionalities for their execution, and the access to large memory devices, enabling larger, and therefore more efficient, batch sizes.

Variable shape is challenging to deal with, particularly in the context of batch training. All images in a batch need to have the exact same shape to be computed, and when your data is of VS, the easiest way to achieve a uniform shape without deformation or loss of information is through padding. Padding is a technique used to extend the size of an image by adding fixed-valued pixels (*e.g.*, zeros), and which can be used to fill the gaps between images of different shape found in the same batch. However, padding is something to be minimized for several reasons [29]. First, it introduces noise (*i.e.*, non-informative pixels) which can affect the learning process. Second, it increases the computational cost of the task, as padding pixels are also computed by the CNN. And third, padding increases the memory requirement of the task, as these values still need to be kept in memory. To complicate things even further, random batching in a VS problem can result in landscape and portrait images batched together. This entails huge amounts of padding, to the point where more padding than informative pixels may remain in the batch.

To motivate research towards hardware that can deal with HR&VS data, in this paper we evaluate the computational differences between a LR problem and a HR&VS problem. We use a public dataset composed by HR&VS images (MAMe, with an average of 6.6 megapixels per images), and explore the behavior of HPC clusters on different versions of it (low, mid and high resolution). Our results illustrate very distinct behaviors among clusters and problems, motivating the idea that better hardware for LR may not be better hardware for HR, and the other way around.

The remaining of the document is organized as follows. In Section 1 we review the related work, in Section 2 the details of the proposed Deep Learning Benchmark based on High Resolution images are explained. Section 3 describes the environment employed for the evaluation and the

complete performance comparison of the benchmark. Finally, we summarize the findings of the paper in the conclusions Section and the future work.

1. Related Work

After Linpack, the most popular benchmarks in the HPC community target specific architecture features: HPCG [11], NAS parallel benchmarks [4] or IO500 [24]. For the purpose of evaluating new architectures, these benchmarks are too specific [22], which calls for a bottom up approach. This means developing benchmarks, kernels and micro-applications that mimic some of the features or phases of scientific workloads: CORAL [2], Graph500 [25]. Even using real workloads to evaluate emerging systems [5, 9, 27].

In this context and with the growing popularity of AI applicaions, several HPC benchmarks for DNNs have been released. This includes AI500, [15], where high resolution images are used on weather predictions; HPL-AI, where mixed-precision operations are tested through solving a system of linear equations, [17, 18] or MLBench, on which different datasets from diverse fields can be processed, [20].

The most relevant for our work (due to the dimensionality of inputs) is MLPerf. In November 2020, this organization released an HPC benchmark for DNNs trainings based on two different tasks [1]. First, a 3D CNN (CosmoFlow [23]) trained with N-body cosmological simulation data to predict cosmological parameters. In this benchmark, data is composed by 128^3 voxels, which in standard 3-channel RGB form would correspond to images of 836x836x3, that is, 2.1 MP. Although the size of these data samples is larger than popular datasets, it still falls short of MAMe [26], with an average size of 6.6 MP. Furthermore, in CosmoFlow all samples have exactly the same size. That is, the data is of fixed size (FS).

The second task proposed in the MLPerf HPC benchmark is a 2D CNN model (DeepCam [19]). That is a convolutional encoder-decoder architecture based on ResNet-50, and trained to process climate related data and identify extreme weather phenomena. This weather data consists of a set of images all of fixed shape 768x1152x16. In standard 3-channel RGB form this would correspond to 2172x2172x3 images, that is, 4.7 MP. This is the same resolution used by another HR benchmark, AI500.

The aforementioned benchmarks are considered of high-resolution, and they are righfully so when compared to alternative tasks [26]. However, it is relatively easy to find data sources of relevance which already exceed these sizes (*e.g.*, mammographies [12]). Significantly, most benchmarking tasks (including both of MLPerf) have fixed size inputs, which simplifies memory management. Again, datasets of variable shape are easy to find in fields of relevance, including the majority of medical imaging sources [31].

The main difference between the MLPerf benchmarks and our work is both the purpose and scope. Regarding the former, MLPerf works with standard low-resolution and fixed-shape images (LR&FS) and its purpose is to overcome benchmarking variability inherent to DL systems due to hyperparameters, stochasticity, software and hardware differences. Instead, the main focus of our work is the use of novel HR&VS images that produce a training execution with different characteristics and, on the other side, we avoid focusing on benchmarking variablity by using partial deterministic training processes. HR&VS data produces a different training execution because this particular type of data requires to modify the training setting, hence producing an execution with different characteristics in comparison to regular LR&FS trainings. Regarding the scope, MLPerf benchmarks are executed on typical accelerators (GPUs & TPUs) to check the

efficiency on current state-of-the-art DL processes, while our work introduces a novel training execution that focus on benchmark hardware based on a different set of hardware characteristics, characteristics required to efficiently execute novel HR&VS trainings.

1.1. Memory Workarounds

In this work we consider a dataset of higher resolution based on MAMe. We use a ResNet18 architecture, which is smaller than the one used by DeepCam, and which requires less memory space for activations and gradients. Yet, in our HR&VS setting, roughly 3% of the training samples already require more than 16 GB of memory. This means the HR&VS task we evaluate in this paper does not fit in memory for accelerators with standard 16 GB memories, even when using batch size one. In these cases, some workarounds enable the use of larger memory loads at the cost of reducing the computation efficiency. Some of these alternatives being:

- **Model parallelism** [3, 10]: Split the model and assign each part to a set of accelerator devices. This technique splits the memory load between devices, effectively increasing the amount of memory available proportional to the number of accelerators. However, this comes at the cost of computation efficiency due to the need of multiple synchronization points and dependencies between such devices.
- **Re-computation** [7]: Re-computing the network activations every time the back-propagation process requires them, instead of keeping them in memory. This considerably reduces the memory from network activations at the cost of some repeated computation. For each step, instead of computing a single inference process, re-computation will compute up to n inference processes, where n is the number of layers in the network.
- **Offloading** [6]: Offloading network activations from accelerator to system memory. Whenever the back-propagation process requires a set of activations, they are transferred back from system to accelerator memory. This technique allows to alleviate memory requirements on the accelerator but at the cost of higher memory access times due to data movement between accelerator and system memory.

All these accelerator workarounds entail a reduction in computational efficiency, the scale of which has not been properly assessed so far. In this context, it is impossible to discard CPU computation as a competitive alternative, until proven otherwise. Particularly, since CPU computation has access to larger memory capacities (and thus, larger batch sizes), while avoiding the overhead of additional components and operations. If anything, CPU computation should be considered as the baseline for all workarounds based on accelerators.

2. High Resolution and Variable Shape (HR&VS) Benchmark

Training DL models with images of high-resolution and variable shape (HR&VS) is an active area of research within the AI field. Using HR&VS has desirable properties, such as keeping all the original information without any loss or deformation. Additionally, there is a set of problems that can only be computed under a HR regime because they require both attention to detail and understanding the overall structure.

In this work we analyze the computational differences between CNN training processes using low-resolution and fixed-shape images (LR&FS trainings) and HR&VS trainings. The goal is to assess the suitability of current hardware for both LR&FS trainings and HR&VS trainings, while highlighting the disparities between both. For such purpose, we use the MAMe dataset

(Museum Art Medium dataset [26]) because of the extreme HR&VS properties of its samples. MAMe contains 37,407 HR photographies of artworks hosted in three different museums. From this dataset, we produce three different input sets: low resolution (LR&FS), mid-resolution (MR&VS) and high resolution (HR&VS), as illustrated in Fig. 1. Notice the LR also implements a fixed shape for all data (FS), while the MR and HR enable variable shape inputs (VS) by adding padding when batching.

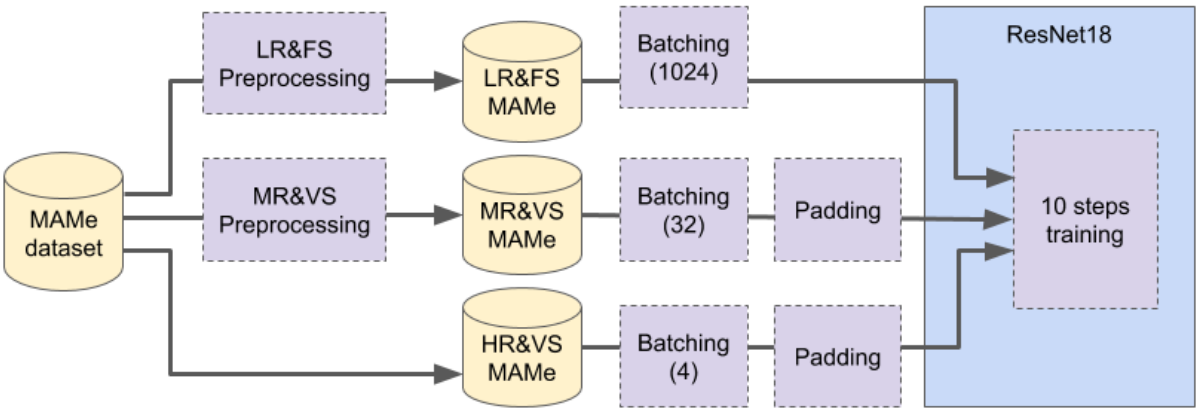


Figure 1. Diagram of the three tasks evaluated

The LR&FS pre-processing consists of down-sampling of all images to a fixed resolution of 256x256 pixels. For the MR&VS, the pre-processing consists of detecting which dimension (*i.e.*, width or height) is smaller, forcing it to 500 pixels long, while keeping the other dimension proportional *w.r.t.* the original aspect ratio of the image (*e.g.*, a 1000x2000 image becomes 500x1000). Finally, HR&VS requires no pre-processing, maintaining the original shape of images as in the MAMe dataset. Table 1 shows the megapixels of a batch for all three settings. Notice the LR has a bigger megapixel size than MR because of the bigger batch size. Also notice the significant variations in megapixels for the experiments that include VS (MR & HR), and how the corresponding padding is close to the amount of informative pixels.

Table 1. Megapixels of each experiment, per batch size for 10 randomly seeded batches. Informative corresponds to megapixels of original data, while padding corresponds to megapixels added after batching to obtain a homogeneous shape among batched samples

		Informative			Padding			Total		
Input	**Batch Size**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**
LR&FS	1024	67.1	67.1	67.1	0	0	0	67.1	67.1	67.1
MR&VS	32	10.9	11.5	12.1	14.5	35.3	87.3	25.4	46.8	99.4
HR&VS	4	16.8	34.6	73.8	10.4	26.8	46.8	27.2	61.5	120.6

To obtain maximum comparability among all training processes, it is important that all of them use the same CNN architecture for the training process. The one which can process both *fixed shape* and *variable shape* data. In other words, the architecture has to be input-agnostic, capable of dealing with different input shapes (although channels dimension will always have a size of 3, Red-Green-Blue channels). We use a Resnet18 architecture [14] because of its popularity among AI practitioners, adding an extra adaptive pooling layer [13] right before the fully-connected layer, as this enables this particular architecture to be input-agnostic.

The batching policy is also shared among input sets, producing batches of samples randomly. While this produces homogeneous tensors for LR&VS, the variable shape nature of MR&VS and HR&VS requires the addition of padding. In these two scenarios, input processing pads the images in the same batch to fill the gaps between images, ensuring the same shape. Batch sizes also differ when training on each input set, to process a similar amount of pixels. By doing so, each executionl allocates nearly the same memory on each batch, ensuring that workloads are not heavily distant among cases. In detail, we use a batch size of 1024 for LR&FS, 32 for MR&VS and 4 for HR&VS corresponding to approximate average memory loads of 28 GB, 20 GB and 26 GB, respectively (see Tab. 2 for further details).

Table 2. Main memory allocation for each task in GB, per batch size for 10 randomly seeded batches

Input sets	Batch Size	Minimum	Average	Maximum
LR&FS	1024	27.9	28.3	28.655
MR&VS	32	11.0	19.8	41.518
HR&VS	4	12.2	25.7	38.003

The training process uses an Adam optimizer [16] with a learning rate of 0.0001, β_1 of 0.9, β_2 of 0.999 and no weight decay. For reproducibility purposes, random seeds are fixed to ensure that the process always join the same images in every batch. All the code necessary for reproducing our experiments is publicly available at our GitHub repository[3].

3. Experiments

In this section, we present the performance results when training the CNN ResNet18 with our proposed setup using a HR&VS input, and a comparison to training the same model using LR&FS and MR&VS. Our goal is to illustrate the computational particularities of a HR&VS setting, the practical relevance of which will continue to grow in the near future.

To that end we follow a top-down approach, analyzing first the elapsed time per megapixel for the training, and entering into detail in the later sections by looking at low level metrics such as: IPC, number of instructions executed per Megapixel, differentiating between counting all the pixels or just the informative ones (*i.e.*, not including the padding pixels), or last level cache misses per 1000 instructions.

3.1. Environment

We use three clusters for the performance evaluation: MareNostrum4, Minotauro and CTE-AMD. In Tab. 3 we can see the different characteristics of each cluster.

One of the main differences between these architectures and that is not highlighted in the previous table is the memory hierarchy. The Zen2 architecture, on which the CTE-AMD processor is based, forms groups of 4 cores, called CCX, and each one of those can directly access its own slice of 16 MB L3 cache. These CCX are paired inside CCDs, and our processor contains a total of 8 CCDs. This means that L3 cache in CTE-AMD is a 256 MB shared resource, but not all this cache is accessible by all cores.

[3]https://github.com/HPAI-BSC/SizeMatters

Table 3. Hardware configuration of the nodes used in the experiments

	MareNostrum4	Minotauro	CTE-AMD
CPU name	Intel Xeon Platinum 8160	Intel Xeon E5-2630 v3	AMD EPYC ROME 7742
Sockets	2	2	1
Cores/socket	24	8	64
Threads/core	1	1	2
Frequency	2.1 GHz	2.4 GHz	2.25 GHz
L1 Cache	private 32 KiB	private 32 KiB	private 32 KiB
L2 Cache	private 1024 KiB	private 256 KiB	private 512 KiB
L3 Cache	shared 33 MiB	shared 20 MiB	shared 256 MiB
Memory/node	96 GiB	128 GiB	1024 GiB
Memory tech	DDR4-2666	DDR4-2133	DDR4-3200

To obtain further insights, we compare the experiments at three levels: using four cores, using one socket and using a full node. Only one node of each cluster is used to avoid the effect of network in the results and focus on architectural differences. We also collect hardware counters using a linux embedded tool, Perf, which allows us to record the events listed in the following table.

Table 4. Hardware counters obtained per cluster

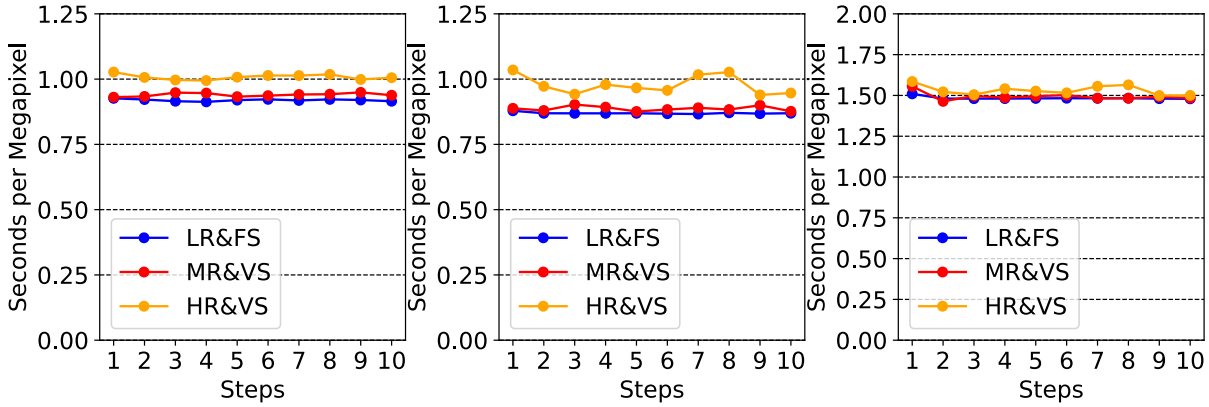
Marenostrum4 and Minotauro	CTE AMD
CPU-CYCLES	CPU-CYCLES
INSTRUCTIONS	INSTRUCTIONS
MEM_LOAD_RETIRED.L3_MISS	l3_misses

We use Singularity (version 3.6.4 on CTE-AMD and 3.5.2 on Marenostrum4 and Minotauro) to containerize the experiments and ensure reproducibility. The same container is used in the three clusters, it includes pytorch 1.6.0, torchvision 0.7.0, numpy 1.17.4, pandas 0.25.3, pillow 6.2.1, python-dateutil 2.8.1, pytz 2019.3 and six 1.13.0.

As said before, the three data-sets explained in Section 2 are used to train the ResNet18 model. In order to homogenize the memory consumption of the different input sets we use a different batch size for each data-set. For LR&FS batch size is 1024, for MR&VS batch size is 32 and for HR&VS the batch size is 4. This will allocate roughly the same amount of memory for all the data-sets. In our results we show the average of three independent runs performing 10 training steps. We have verified that the variability between different runs is below 6%.

3.2. Execution Time

In this subsection we show the differences in timing when processing the different input sets in each cluster. In the context of random batching, LR&FS task produces batches with a constant shape because all images have same dimensions. However, in those cases where images have variable shape (MR&VS and HR&VS), the amount of pixels to compute in a batch may

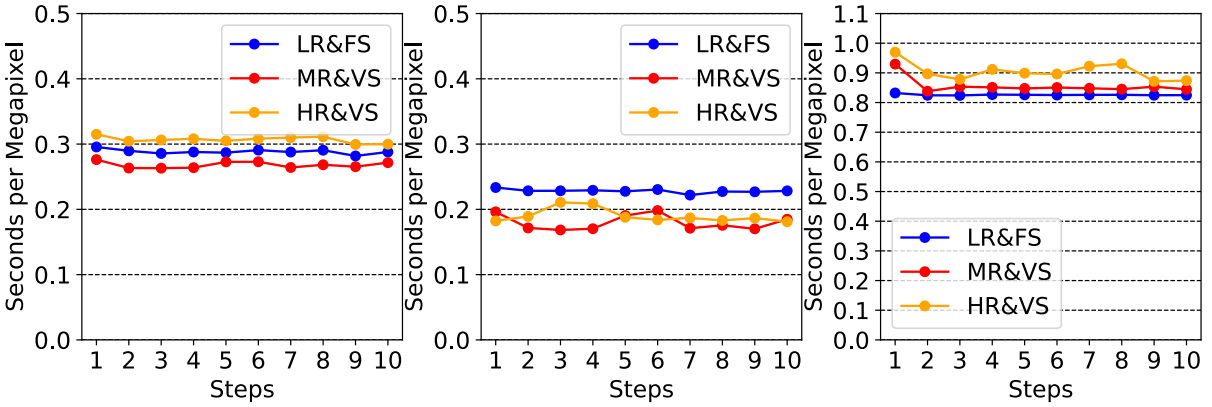


(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

Figure 2. Execution time per megapixel using 4 cores

vary significantly from step to step hence. For this reason we use the "Seconds per Megapixel" metric as a common ground.

In Fig. 2 we can see performance obtained on the different clusters when using 4 cores to process the different data sets. We observe not only that there is a difference of performance achieved by the different clusters, but also that the performance between the different data sets is different. In Minotauro the performance variation between the different data sets is minimum, while in CTE-AMD, the HR&VS data set performs worse than the others, presenting also a high variability between the different steps. In Marenostrum the HR&VS also shows a worse performance but without the variability seen in CTE-AMD.

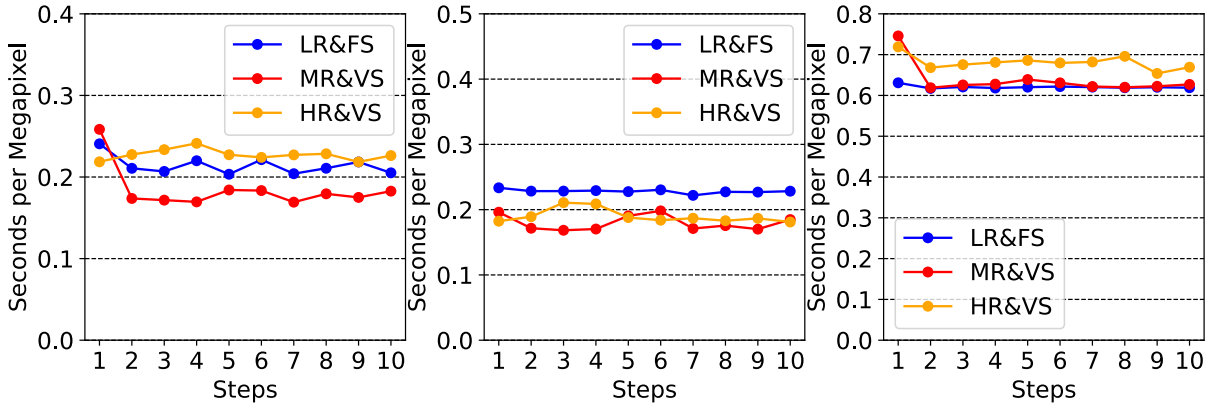


(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

Figure 3. Execution time per megapixel using 1 socket

In Fig. 3 we show the seconds per megapixel achieved when using one full socket in each cluster. In MareNostrum4 and Minotauro the HR&VS is the data set with worse performance but not in CTE-AMD. The LR&FS is the best performing one in Minotauro and the worse one in CTE-AMD. In CTE-AMD both HR&VS and MR&VS present a high variability and it is not clear which one performs better. This illustrates the differences in computational behaviour caused by a change in the input resolution.

The performance obtained when using a full node of each cluster can be seen in Fig. 4. We observe that, in MareNostrum4 the MR&VS performs clearly worse than the other data sets,



(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

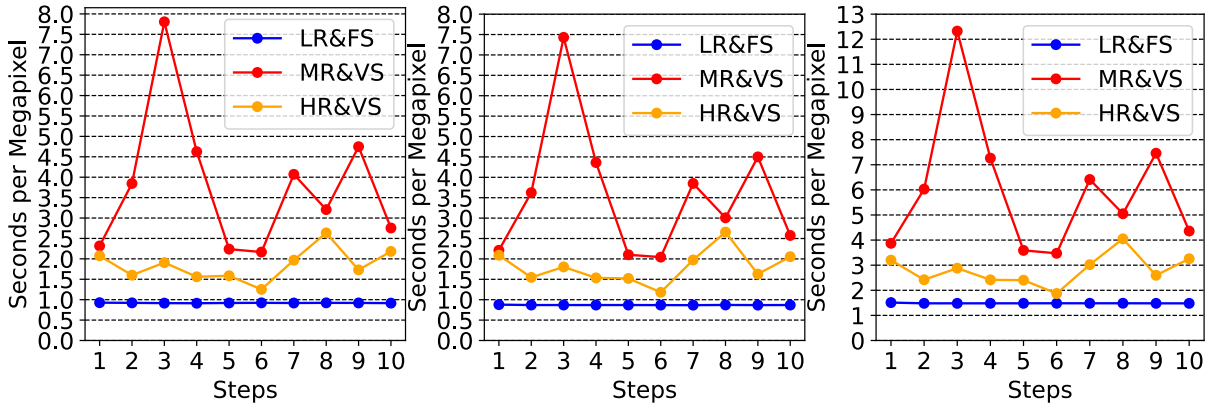
Figure 4. Execution time per mega pixel using 1 node

and also that the HR&VS is slightly better than the LR&FS data set. In Minotauro, on the other hand, the worse performing data set is the one using HR&VS images, without relevant difference between using LR&FS or MR&VS.

With this first analysis, we have demonstrated that the HR&VS data set presents a different performance from the LR&FS when running on different architectures. This alone illustrates the particular nature of HR&VS when compared to other DNN tasks, and justifies specific benchmarks for it. To understand where the differences come from we expand the analysis in the following sections with additional metrics.

3.3. Padding Effect

One of the main differences between the different data sets is the padding, added to the HR&VS and MR&VS data sets, in order to keep a homogeneous shape within each batch. We refer to these pixels as non-informative, as they do not carry any information but they are computed by the network nevertheless. In this section, we study the performance of the different experiments in each cluster taking into account only the informative pixels.

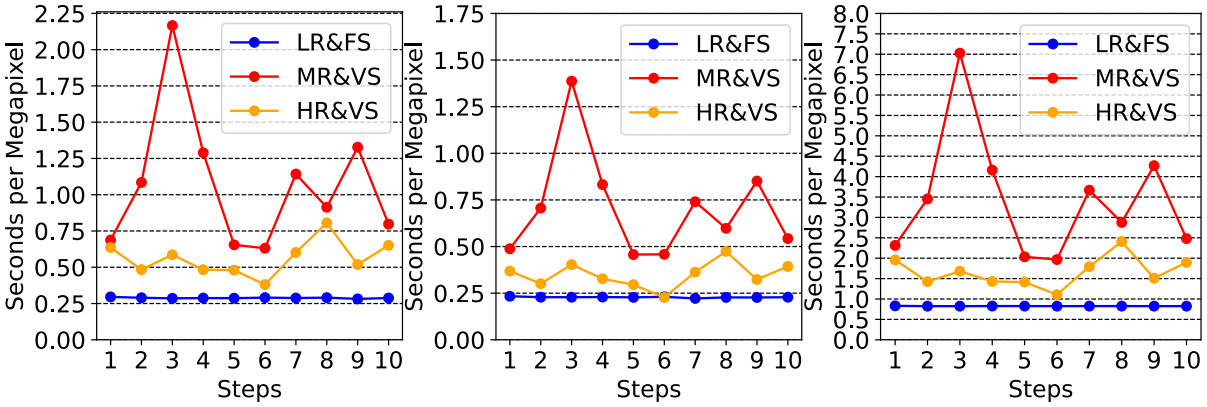


(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

Figure 5. Execution time per informative megapixel using 4 cores

In Fig. 5 we can see the seconds per informative Megapixel achieved by the different data sets when running in 4 cores of each cluster. We observe that, although, the performance obtained

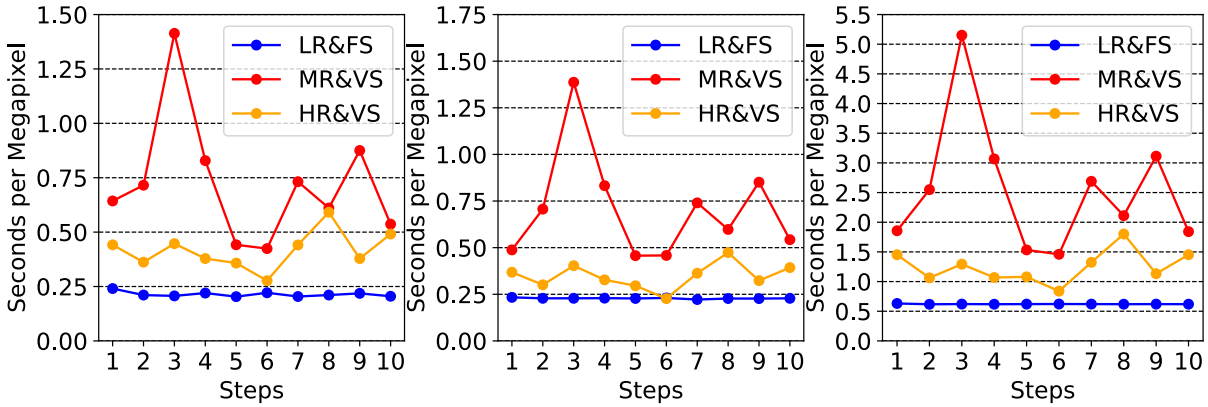
in each cluster is different, in all the clusters the best performing input set is the the LR&FS. Significantly, the LR&FS data set does not have padding pixels therefore, all the computation is done on informative pixels. This indicates that padding is adding a significant overhead to the computation. The worse performing data set is the MR&VS with a high variability between different time steps, this is explained because the MR&VS data set has a higher batch size than the HR&VS which means a higher variability in image shapes within the same batch and therefore, more padding pixels.



(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

Figure 6. Execution time per informative megapixel using 1 socket

The performance results when using one socket of each cluster can be seen in Fig. 6. In this case the results look similar to the ones obtained when using 4 cores, the order and shape of the lines are the same. The most relevant difference between the clusters is the difference in performance obtained between MareNostrum4 and CTE-AMD, compared when using 4 cores. While using 4 cores the performance of both clusters is very similar, when using a full socket CTE-AMD clearly outperforms MareNostrum4. This can be explained by the difference in the number of cores between the two clusters, 24 for MareNostrum4 and 64 for CTE-AMD.



(a) MareNostrum4 (b) CTE-AMD (c) Minotauro

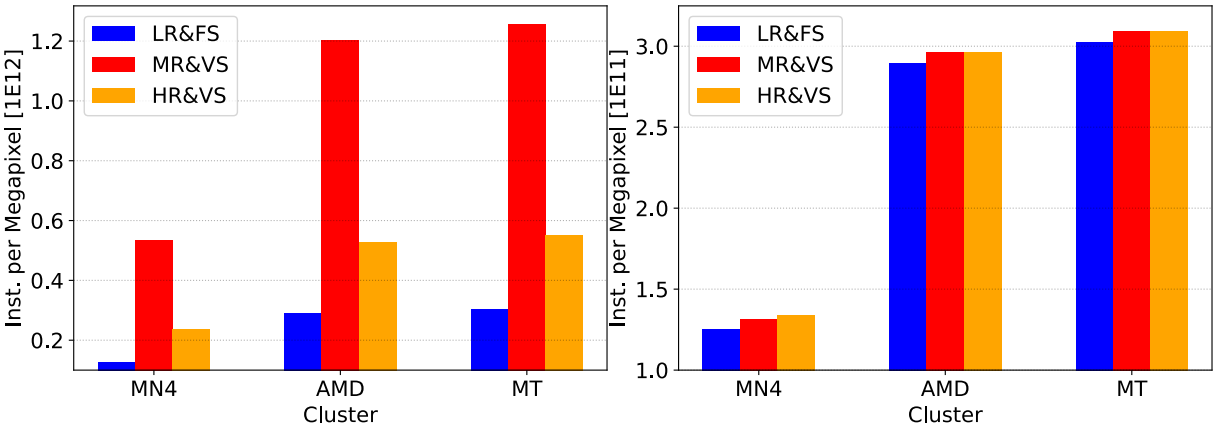
Figure 7. Execution time per informative megapixel using 1 node

In Fig. 7 we show the performance obtained taking into account the informative pixels when using the full node. While the performance of the different data sets remains the same for the different clusters, we can observe that one node of Marenostrum4 achieves the same performance

as the CTE-AMD with less cores. With these results we can conclude that the padding pixels (which have always zero value) add overhead to the training, but less than informative pixels.

3.4. Executed Instructions

In this section, we study the amount of instructions required to train the model with the different inputs using the two metrics: with and without non-informative pixels. Note that in this case the number of instructions executed are obtained from hardware counters at the end of the execution, therefore, metrics are not detailed per time step.



(a) Informative pixels

(b) All pixels

Figure 8. Number of instructions per MP in 4 cores

In Fig. 8 we can see the Instructions per Megapixel executed in each experiment and cluster. On the right hand the plot depicts the Instructions per Megapixel taking into account all the pixels processed (including padding), on the left hand side we can see the same metric but taking into account only the informative pixels.

The most straightforward conclusion from this experiment is that the number of instructions necessary to process a pixel is almost the same if the pixel is informative or not. This can be seen in Fig. 8b, where LR&FS (which holds no padding) executes almost the same number of instructions as the MR&VS or HR&VS per pixel. From Fig. 8a we can also see that there are important differences in the number of instructions executed per informative Megapixels in the different input sets. The pattern of this difference (higher for MR&VS, lower for LR&FS) confirms that the difference comes from the padding pixels that are being executed but not accounted.

We can observe the same pattern across the different architectures: MR&VS always presents a higher number of instructions per MP than the other inputs and LR&FS the lowest one. This means that the kind and number of instructions executed do not depend on the resolution of the image or the batch size used.

Finally the last observation is the difference in the number of instructions executed per Megapixel between the different clusters. While Minotauro and CTE-AMD show a similar number of instructions executed, Marenostrum4 roughly uses half the number of instructions that the other two clusters. In order to explain the lesser number of instructions executed by MareNostrum4 cluster, we have to point at its CPU capabilities. In particular, we can see that the Intel Xeon Platinum 8160 is capable of executing AVX512 instructions, while CTE-AMD and Minotauro can only execute AVX2 and the vector length of AVX512 is twice the size of AVX2 vector.

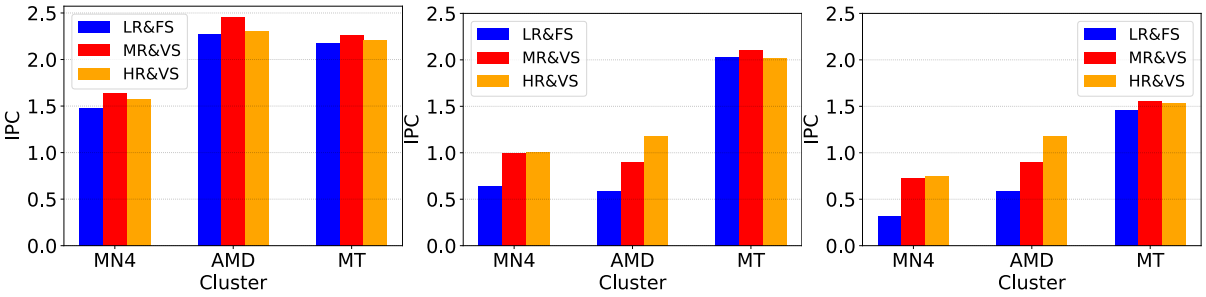
This observation also indicates that all input sets are making an intensive use of the vector units of the different processors.

We do not show the corresponding plots for the execution on one socket and one node because they show the same pattern and there is no difference in the number of instructions executed.

After these results, we can conclude that the number of instructions depends on the total amount of pixels, and non-informative pixels require the same number of instructions as the informative ones. Also, there is no difference in the cost in terms of instructions when processing LR, MR or HR pixels.

3.5. Instructions per Cycle (IPC)

In the previous sections we have seen that there is a difference in the execution time between the different input sets and that this difference does not come from the total number of instructions executed. Here we analyze the IPC of each cluster when facing each of the input sets for the different configurations and clusters. In Fig. 9 we can see the average IPC obtained by each input set in each cluster when using four cores, one socket or one node.



(a) 4 cores (b) 1 socket (c) 1 node

Figure 9. Instructions per cycle (IPC)

Looking at Fig. 9a we observe that all the clusters share the same pattern when training the network with the different image input sets. The MR&VS input set achieves the highest IPC in all the cases and LR&FS is the lowest one. Knowing that the MR&VS input set is the one that includes more padding pixels and that the LR&FS does not include padding pixels, we can assume that, although the informative and non-informative pixels need the same number of instructions, the instructions used for non-informative pixels are faster to execute (*i.e.*, use less cycles to complete).

We also observe a notable difference between the IPC achieved by the different clusters where Marenostrum4 shows a lower IPC than the other clusters. In the previous section we have seen that Marenostrum4 executes less instructions for the same input than the other clusters but these instructions take more cycles than the ones executed by CTE-AMD and Minotauro.

In Section 3.2 Fig. 2a Marenostrum4 and CTE-AMD has showed a similar performance in terms of execution time per Megapixel. It is interesting to see that this is achieved by both architectures using different vector units but delivering the same performance. In Marenostrum4 executing less instructions at a lower IPC and in CTE-AMD executing more instructions at a higher IPC.

Looking at Fig. 9b we find the IPC when using a whole socket of each cluster. We can notice important variations on shapes. In this case Minotauro gets the best IPC having almost

no difference with the one obtained when using 4 cores, this is due to the fact that Minotauro is the one with the fewer cores per socket (8, versus 24 in Marenostrum4 and 64 in CTE-AMD) with 8 cores the shared resources of the socket are not being saturated.

On the other hand, CTE-AMD shows an important drop in the IPC when using one socket (64 cores) with respect to using 4 cores. This could indicate a saturation on the memory bandwidth but we will verify this assumption in the following section looking at memory access hardware counters. Marenostrum4 also shows a drop in the IPC when using a full socket (24 cores) but not as drastic as the one on CTE-AMD.

Looking at the different input sets we also see relevant differences. The LR&FS input set has a lower IPC when running in one socket than when running in 4 cores. We know that it is not related to the padding pixels as MR&VS and HR&VS do not show the same trend (MR&VS having more padding pixels than HR&VS). We could relate it to the batch size, this means that although the amount of memory accessed is roughly the same as shown in Section 2 Tab. 2 there is a difference depending if it is organized in more images or less.

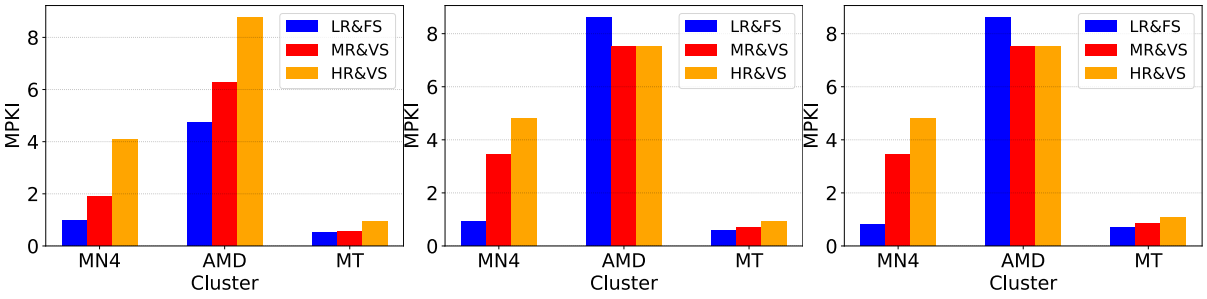
In Fig. 9c we can see the IPC achieved by each cluster when using a full node. Both Minotauro and Marenostrum 4 show a lower IPC than when using one socket, meaning that using more cores some of the shared resources of the node are being saturated. It is also interesting to notice the different behavior of the different input sets. In Minotauro there seems to be not a very important difference in the IPC obtained by the different inputs. Therefore, there is a difference in the execution time shown in Fig. 4c that is not explained by the IPC nor by the number of instructions executed by the different input sizes. At this point this difference can only come from a difference in the frequency (which is fixed for the HPC systems being evaluated but could come from low cycles per microsecond due to I/O operations or OS preemption) or the useful execution of instructions, meaning that with the current approach we account for all the instructions executed, but there could be phases with busy waiting processes or threads that are not performing useful work. A further detailed analysis is needed to unveil this difference.

Between Marenostrum4 and CTE-AMD there is also an important difference, in Marenostrum4 MR&VS achieves the same IPC as HR&VS while in CTE-AMD the IPC obtained when training with HR&VS is higher than when using MR&VS. We will try to understand this differences in the following section looking at memory HWC in detail.

3.6. Memory Access

In this section, we analyze the data access when using the different input sets for the training of the network. In Fig. 10 we can see the misses of the last level cache (LLC), that in all the clusters correspond to L3, per 1000 instructions (MPKI). We assume every miss on L3 implies a data transfer from memory, therefore, we use this metric as a measure of the pressure on the memory system.

Looking at Fig. 10a, that corresponds to the execution using 4 cores, we see a common behaviour in all the clusters: the HR&VS input set has a higher MPKI than the other input sets, and LR&FS has the lower MPKI. The CTE-AMD cluster shows a much higher MPKI for all the input sets than the other clusters, this can be explained by its architecture, as the 4 cores that are being used belong to the same CCX and share a 16 MB L3. This is quite small compared to the 33 MB available in the L3 of Marenostrum4.



(a) 4 cores (b) 1 socket (c) 1 node

Figure 10. LLC Misses per 1000 instructions (MPKI)

We have to take into account that in CTE-AMD when a core misses an access to its L3 cache slice, the data can come from another CCX cache or from main memory and with the given hardware counters we cannot differentiate this two cases.

We can see the MPKI for the execution using one socket in Fig. 10b. In this case we observe an important change in the behaviour of the CTE-AMD. The LR&VS set is the one showing a higher MPKI, almost twice the one observed when using 4 cores. MR&VS is also higher but HR&VS shows a lower MPKI than when using 4 cores. The increase in MPKI for the LR&FS and MR&VS can be explained because the L3 is shared and the different processes are removing each others data from L3. And the decrease of HR&VS is an effect of having more capacity of L3 available, as now using the whole socket it has 256 MB of L3 available.

It is clear that the different input sets present a different behaviour in terms of memory accesses. Probably the LR&FS can reuse less data because of the higher batch size, while HR&VS can reuse more data from the caches.

Conclusions

Motivated by the need of processing high resolution and variable shape images, we have uncovered relevant performance differences and needs. These are visible between the different input sets when running in clusters with different system configurations and CPU architectures.

The use of padding pixels, needed to create a batch of variable shape images, has a significant impact in the performance. As shown in Figs. 8 and 9, padding pixels require the same amount of instructions as informative pixels but they can be processed faster than informative ones.

We have not been able to explain the differences in performance between the different input sets when running in Minotauro using the aggregated hardware counters. A more detailed analysis based on tracing is necessary.

Clearly, the use of vector units such as AVX512 is beneficial for this kind of workloads, but it is interesting to notice also that smaller vector units can deliver the same performance if they can run at a higher IPC.

We have also demonstrated that the memory access pattern is different between regular LR&FS and novel HR&VS sets, even though all the input sets used roughly the same amount of memory the MPKI measured are different. This means that the configuration of the batch has an impact in the memory access pattern, by batch configuration we mean number of images, size of the images and amount of padding pixels.

Future Work

The work presented here analyzes a problem (training CNNs with HR&VS data) of relevance for the next generation of AI services (*e.g.*, medical diagnosis, autonomous driving), which is at the limit of what can be computed efficiently by current HPC infrastructure (due to memory requirements). For assessing the relevance of this work, the definition of future work is of paramount importance.

Following the problem introduction of this paper, we foresee two main milestones in the future. First, gaining further insights into the problem at hand, since the analysis here presented is of limited depth. And second, implementing and releasing a closed benchmark to facilitate adoption by the community.

The analysis of this work is of limited scope because we have only access to hardware counters and a limited list of them. The next step in the analysis will be to do a detailed performance analysis using tracing tools to understand the parallel and computational behaviour of the different executions.

Finally, although all codes and environments needed to reproduce our results are publicly available, further work is needed towards transforming the work of this paper into an HPC benchmark. The best way to do so is to integrate our work with a well established benchmarking organization, like MLPerf. At that point, we expect researchers all over the world to fully tackle the proposed problem, eventually solving it through the next generation of HPC hardware and software.

References

1. Press Release 11/18/20: MLPerf Releases Inaugural Results for Leading High-Performance ML Training Systems (2020), `https://mlperf.org/press#mlperf-hpc-v0.7-results`

2. Advanced Simulation and Computing: CORAL Benchmarks. `https://asc.llnl.gov/coral-benchmarks`, accessed: 2021-04-06

3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR abs/1409.0473 (2014), `https://arxiv.org/abs/1409.0473`

4. Bailey, D., Harris, T., Saphir, W., Van Der Wijngaart, R., Woo, A., Yarrow, M.: The NAS parallel benchmarks 2.0. Tech. rep., Technical Report NAS-95-020, NASA Ames Research Center (1995)

5. Banchelli, F., Garcia-Gasulla, M., Houzeaux, G., Mantovani, F.: Benchmarking of state-of-the-art HPC Clusters with a Production CFD Code. In: Proceedings of the Platform for Advanced Scientific Computing Conference, 29 June-1 July 2020, Geneva, Switzerland. pp. 1–11 (2020), DOI: 10.1145/3394277.3401847

6. Beaumont, O., Eyraud-Dubois, L., Shilova, A.: Optimal GPU-CPU Offloading Strategies for Deep Neural Network Training, pp. 151–166 (2020), DOI: 10.1007/978-3-030-57675-2_10

7. Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training deep nets with sublinear memory cost. CoRR abs/1604.06174 (2016), https://arxiv.org/abs/1604.06174

8. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 21-26 July 2017, Honolulu, HI, USA. pp. 1907–1915. IEEE (2017), DOI: 10.1109/cvpr.2017.691

9. Criado, J., Garcia-Gasulla, M., Kumbhar, P., Awile, O., Magkanaris, I., Mantovani, F.: CoreNEURON: Performance and Energy Efficiency Evaluation on Intel and Arm CPUs. In: 2020 IEEE International Conference on Cluster Computing (CLUSTER), 14-17 Sept. 2020, Kobe, Japan. pp. 540–548. IEEE (2020), DOI: 10.1109/cluster49012.2020.00077

10. Dean, J., Corrado, G.S., Monga, R., et al.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)

11. Dongarra, J., Heroux, M.A., Luszczek, P.: HPCG benchmark: a new metric for ranking high performance computing systems. The International Journal of High Performance Computing Applications 30(1), 3–10 (2016), DOI: 10.1177/1094342015593158

12. Geras, K.J., Wolfson, S., Shen, Y., et al.: High-resolution breast cancer screening with multi-view deep convolutional neural networks. CoRR abs/1703.07047 (2017), https://arxiv.org/abs/1703.07047

13. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence 37(9), 1904–1916 (2015), DOI: 10.1007/978-3-319-10578-9_23

14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 27-30 June 2016, Las Vegas, NV, USA. pp. 770–778. IEEE (2016), DOI: 10.1109/cvpr.2016.90

15. Jiang, Z., Gao, W., Wang, L., et al.: HPC AI500: a benchmark suite for HPC AI systems. In: International Symposium on Benchmarking, Measuring and Optimization, 10-13 Dec. 2018, Seattle, WA, USA. pp. 10–22. Springer (2018), DOI: 10.1007/978-3-030-32813-9_2

16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014), https://arxiv.org/abs/1412.6980

17. Kudo, S., Nitadori, K., Ina, T., Imamura, T.: Implementation and Numerical Techniques for One EFlop/s HPL-AI Benchmark on Fugaku. In: Proceedings of the 11th IEEE/ACM Workshop on Latest Advances in Scalable Algorithms for Large-Scale, 13 Nov. 2020, GA, USA. pp. 69–76. IEEE (2020), DOI: 10.1109/ScalA51936.2020.00014

18. Kudo, S., Nitadori, K., Ina, T., Imamura, T.: Prompt report on Exa-scale HPL-AI benchmark. In: 2020 IEEE International Conference on Cluster Computing (CLUSTER), 14-17 Sept. 2020, Kobe, Japan. pp. 418–419. IEEE (2020), DOI: 10.1109/cluster49012.2020.00058

19. Kurth, T., Treichler, S., Romero, J., et al.: Exascale deep learning for climate analytics. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, 11-16 Nov. 2018, Dallas, TX, USA. pp. 649–660. IEEE (2018), DOI: 10.1109/sc.2018.00054

20. Liu, Y., Zhang, H., Zeng, L., Wu, W., Zhang, C.: MLbench: benchmarking machine learning services against human experts. Proceedings of the VLDB Endowment 11(10), 1220–1232 (2018), DOI: 10.14778/3231751.3231770

21. Lotter, W., Sorensen, G., Cox, D.: A multi-scale CNN and curriculum learning strategy for mammogram classification. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 17 Sept. 2017, Québec City, QC, Canada, pp. 169–177. Springer (2017), DOI: 10.1007/978-3-319-67558-9_20

22. Mantovani, F., Garcia-Gasulla, M., Gracia, J., et al.: Performance and energy consumption of HPC workloads on a cluster based on Arm ThunderX2 CPU. Future Generation Computer Systems 112, 800–818 (2020), DOI: 10.1016/j.future.2020.06.033

23. Mathuriya, A., Bard, D., Mendygral, P., et al.: CosmoFlow: Using deep learning to learn the universe at scale. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, 11-16 Nov. 2018, Dallas, TX, USA. pp. 819–829. IEEE (2018), DOI: 10.1109/sc.2018.00068

24. Monnier, N., Lofstead, J., Lawson, M., Curry, M.: Profiling platform storage using IO500 and mistral. In: 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW), 18 Nov. 2019, Denver, CO, USA. pp. 60–73. IEEE (2019), DOI: 10.1109/pdsw49588.2019.00011

25. Murphy, R.C., Wheeler, K.B., Barrett, B.W., Ang, J.A.: Introducing the Graph 500. Cray Users Group (CUG) 19, 45–74 (2010)

26. Parés, F., Arias-Duart, A., Garcia-Gasulla, D., et al.: A Closer Look at Art Mediums: The MAMe Image Classification Dataset. CoRR abs/2007.13693 (2020), `https://arxiv.org/abs/2007.13693`

27. Ramirez-Gargallo, G., Garcia-Gasulla, M., Mantovani, F.: TensorFlow on state-of-the-art HPC clusters: a machine learning use case. In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 14-17 May 2019, Larnaca, Cyprus. pp. 1–8. IEEE (2019), DOI: 10.1109/ccgrid.2019.00067

28. Reichstein, M., Camps-Valls, G., Stevens, B., et al.: Deep learning and process understanding for data-driven earth system science. Nature 566(7743), 195–204 (2019), DOI: 10.1038/s41586-019-0912-1

29. Sotiropoulos, I.N.: Handling variable shaped & high resolution images for multi-class classification problem. Master's thesis, Universitat Politècnica de Catalunya (2020)

30. Treml, M., Arjona-Medina, J., Unterthiner, T., et al.: Speeding up semantic segmentation for autonomous driving. In: MLITS, NIPS Workshop. vol. 2, p. 7 (2016)

31. Wu, N., Phang, J., Park, J., et al.: Deep neural networks improve radiologists' performance in breast cancer screening. IEEE transactions on medical imaging 39(4), 1184–1194 (2019), DOI: 10.1109/TMI.2019.2945514