Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Methodology for Selecting and Ranking High-Performance GPU Configurations in Deep Learning Training

David Cortes

d.cortes@uib.es

University of the Balearic Islands

Carlos Juiz

University of the Balearic Islands

Belen Bermejo

University of the Balearic Islands

Research Article

Keywords:

Posted Date: November 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8784434/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Methodology for Selecting and Ranking High-Performance GPU Configurations in Deep Learning Training

David Cortes[1*†], Carlos Juiz[1†] and Belen Bermejo[1†]

[1*]Computer Science Department, University of the Balearic Islands, Spain.

*Corresponding author(s). E-mail(s): d.cortes@uib.es;
Contributing authors: cjuiz@uib.es; belen.bermejo@uib.es;
†These authors contributed equally to this work.

Abstract

This study introduces a methodology for the systematic evaluation and classification of GPU configurations used in Deep Learning training tasks. The proposed approach follows a three-stage process that includes the identification of relevant performance variables, the grouping of systems through clustering techniques, and a comparative ranking of the resulting categories based on scalability and efficiency criteria. By structuring the analysis around metrics such as training time, speedup, throughput, and resource efficiency, the methodology enables the discovery of hardware configurations that achieve favourable trade-offs between computational performance and accelerator utilization. Although demonstrated on a specific training workload, the approach is general and can be applied to any dataset or benchmarking scenario that exposes equivalent performance indicators. The results offer a reproducible framework for selecting high-performance GPU setups in large-scale AI environments, contributing to more efficient and cost-effective training infrastructures.

Keywords: Benchmarking, Deep Learning, GPUs, MLPerf, Performance, Methodology

1 Introduction

Hardware performance is a key factor in optimizing compute-intensive tasks, such as training Deep Learning (DL) models. As these models, such as ResNet-50, become more complex and require large datasets for training, the demand for more powerful and efficient hardware accelerators has grown significantly, how GPUs. In this context, proper hardware selection has a direct impact on training latency, resource utilization, and operational costs [1, 2], highlighting the importance of conducting a comparative analysis of the performance of different GPUs in training such models.

The MLPerf Training benchmark [3] has become an industry standard for evaluating the performance of various hardware configurations in deep training tasks, including ResNet-50, a widely used model for image classification. This benchmark provides a reliable platform for assessing how different hardware architectures and technologies address the challenges of scalability and efficiency in Machine Learning (ML). The concepts of efficiency and ideal scalability are fundamental when evaluating hardware performance in DL training. It is important to note that efficiency refers to how effectively computational resources are utilized and is commonly defined as the ratio between achieved speedup and the number of accelerators used [4]. In an ideal scalability scenario, increasing the number of GPUs would result in a proportional reduction in training time, for example, doubling the number of accelerators (GPUs), the execution time will be a half. However, in real-world systems, factors such as interconnect bottlenecks, memory bandwidth limitations, and communication overhead often prevent perfect scalability. Productivity per GPU is another crucial metric, representing the number of images processed per second per GPU. Understanding these concepts is essential for assessing the trade-offs between performance gains and resource allocation in high-performance computing environments.

This study proposes a methodology for the systematic analysis and classification of GPU configurations used in Deep Learning training tasks. While the MLPerf Training benchmark for ResNet-50 serves as the reference dataset in this work, the proposed approach is generalizable and can be applied to other datasets with similar characteristics. The methodology focuses on identifying relevant performance variables, grouping systems with comparable behaviors, and ranking them according to their trade-offs between performance and scalability. Through this structured process, the study provides a reproducible framework that facilitates the evaluation and selection of high-performance GPU configurations in large-scale AI training environments.

The structure of this article is as follows: Section 2, provides the foundational knowledge used in this study. Section 3, reviews related works. Section 4, Problem Statement, describes the initial data from the MLPerf Training benchmark and the limitations observed in scalability, with additional calculations validating these observations. Section 5, Methodology, explains the steps undertaken, including variable analysis, clustering, and observation, leading to the comparative analysis presented in Section 6, Results. Section 7, Discussion, addresses heuristics, observed phenomena, and their implications, while Section 8, Conclusions, summarizes the main findings and recommendations.

2 Background

2.1 Deep Learning

Deep Learning is an advanced branch of Machine Learning that utilizes artificial neural networks with multiple layers to model and learn complex patterns in large datasets [5]. Artificial neural networks are computational models inspired by the functioning of the human brain, consisting of artificial neurons organized in layers. Each neuron receives a set of inputs, performs a mathematical operation, and transmits its output to other neurons [6]. These networks can learn complex relationships within data by adjusting the connection weights between neurons through training algorithms such as backpropagation [6].

Unlike traditional ML methods, where feature extraction requires manual intervention, DL enables automatic extraction of high-level features directly from input data [7]. This makes it particularly effective for handling unstructured data such as images, text, and audio. Its ability to identify subtle patterns has led to significant advancements in applications like image and voice recognition, natural language processing, and computer vision [7].

ResNet-50 is a deep convolutional neural network architecture developed by Microsoft in 2015 [8]. Convolutional Neural Networks (CNNs) are specifically designed to process data with grid-like structures, such as images [9]. These networks apply convolutional filters to the input data to automatically detect features like edges, textures, and more complex patterns at deeper layers of the network [10].

Moreover, ResNet-50 belongs to the family of Residual Networks (ResNets), which are characterized by introducing residual connections. These connections allow part of the input information to bypass intermediate layers and pass directly to later layers, helping to mitigate the vanishing gradient problem a common challenge when training very deep networks [8]. By addressing this issue, ResNet-50 enables the training of deeper and more efficient networks, capable of capturing more complex and abstract representations of the input data.

Regarding the ResNet-50 architecture, it consists of 50 layers with trainable weights, making it well-suited for tasks such as image classification and segmentation. For this study, the ResNet-50 network was trained using the ImageNet dataset, a benchmark dataset in computer vision [11, 12]. ImageNet contains approximately 1.2 million labelled images distributed across 1,000 distinct categories encompassing a wide range of objects, animals, and scenes. Each image is associated with a label corresponding to a specific class, making it an ideal resource for training and evaluating image classification models [11, 12]. The diversity and scale of ImageNet enable accurate measurement of the generalization capability and performance of complex architectures such as ResNet-50. By training on this dataset, ResNet-50 learns hierarchical representations, starting with features in the initial layers and progressing toward more abstract patterns in the deeper layers. The use of ImageNet in this study provides a robust reference framework for evaluating the performance and efficiency of hardware configurations in large-scale DL model training tasks.

2.2 Clustering

Clustering is a fundamental technique in unsupervised learning, whose primary goal is to organize and classify objects' data points or clusters based on their intrinsic similarities, without the need for predefined labels [13]. This technique is widely applied in fields such as market segmentation, genomic data analysis, and image processing due to its ability to uncover hidden patterns and underlying structures in data [13].

There are various clustering methods, each with distinct characteristics and applications [13]. In this study, we utilize four methods: K-means [14], Agglomerative Clustering [15], Gaussian Mixture Models (GMM) [16], and Spectral Clustering [17].

K-means is one of the most popular and simplest clustering algorithms. It divides the data into, k, predefined groups, assigning each point to the cluster with the nearest centroid and iteratively minimizing the distance between points and their assigned centroid. While it is efficient and easy to implement, K-means assumes that clusters have a spherical shape and are sensitive to centroid initialization, which can impact the quality of the results [14].

Agglomerative Clustering is a hierarchical method that adopts a bottom-up approach. Initially, it considers each data point as an individual cluster and progressively merges the closest cluster until a complete hierarchical structure is formed [15]. This method is particularly useful for exploring the hierarchical structure of data [13].

GMM, employs a probabilistic approach to clustering, assuming that the data originates from a mixture of Gaussian distributions. Unlike K-means, GMM is more flexible and can capture clusters of varying shapes and sizes, making it suitable for data with more complex structures [16]. Finally, Spectral Clustering uses eigenvectors, which describe the principal directions in which the data varies. These eigenvectors are derived from the spectral decomposition of a similarity matrix, allowing complex data to be transformed into a simpler and more manageable representation [17]. This technique is particularly effective for identifying clusters with non-convex shapes and can handle nonlinear relationships between data points. It is especially useful when traditional methods, such as K-means, fail to correctly capture the underlying structures [13]. In this study, to measure the quality of the obtained clusters, various validation indices are used to evaluate and describe different characteristics [13]. The indices included in this article are the Silhouette Score [18], the Davies-Bouldin Index [19], and the Calinski-Harabasz Index [20]. Silhouette Score evaluates how similar an object is to its cluster compared to other clusters, providing a measure of internal cohesion and separation between clusters. Higher values indicate better clustering assignments [18]. Davies-Bouldin Index assesses the separation between clusters and the compactness within them. Lower values indicate better clustering [19]. Calinski-Harabasz Index, also known as the variance ratio criterion, compares the dispersion between and within clusters [21]. Higher values suggest better-defined clusters [20].

Although Random Forest is primarily used for classification and regression tasks, it can also play an important role in the context of clustering [22], particularly in the selection of relevant features [23]. Random Forest builds multiple decision trees and combines their results to achieve more accurate and robust predictions. A key advantage is its ability to measure the importance of variables in prediction. This capability allows the identification of key features within a dataset, which is essential

for reducing dimensionality and improving the performance of clustering algorithms applied subsequently [23].

2.3 Performance and Scalability

In ML and Artificial Intelligence, performance and scalability are critical aspects that determine the viability and success of solutions in real-world environments. In this context, the focus of this work is on training time, that is, the time required to adjust a model's parameters using a given dataset. Within this framework use the following terms: the performance of a computing system can be defined as the inverse of the response time of a workload. The faster the execution is completed, the higher the performance of the system [4]. In terms of model training, performance can be defined as the inverse of the time it takes for the model to complete the training process. Performance, when applied to the comparison of systems, includes speedup, which measures how much faster one system is compared to another. It can be defined as the ratio of execution times, and in our training context, it is calculated as the ratio of the old time to the new time [4], as shown in **Equation 1**.

$$\text{Speedup} = \frac{\text{Reference Time}}{\text{New Time}} \tag{1}$$

Equation 1. Speedup formula.

Efficiency is defined as the ratio between the achieved speedup and the number of accelerators or processors used [4]. Mathematically, it is expressed as shown in **Equation 2**. This metric indicates how effectively the available resources are being utilized. Efficiency reaches a maximum value of 1 in ideal cases, where the performance improvement is proportional to the number of accelerators employed.

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of accelerators}} \tag{2}$$

Equation 2. Efficiency Formula.

Scalability is the ability of a system to adapt to an increase in workload while maintaining or improving its performance. This concept includes capacity planning, which enables the flexible allocation of resources according to variable demands, as seen in distributed applications or on-demand systems [4]. Throughput represents the amount of work or tasks completed per unit of time. In the context of ML model training, throughput is defined as the number of images processed per unit of time during training.

$$\text{Throughput} = \frac{\text{Number of images processed}}{\text{Total training time}} \tag{3}$$

Equation 3. Throughput Formula.

Throughput is a key metric for evaluating the productivity of a system [4]. In this article, we use the size of the dataset, ImageNet, as the number of images processed divided by the time taken to train the model, as shown in **Equation 3**. Additionally, using this resulting value, we can calculate the productivity associated with the

number of accelerators, defined as the ratio between the total throughput and the number of accelerators (GPUs) used during training. Mathematically, this is shown in **Equation 4**. This metric allows for the evaluation of the individual efficiency of each GPU in contributing to the overall system throughput.

$$\text{Productivity per GPU} = \frac{\text{Throughput}}{\text{Number of accelerators}} \tag{4}$$

Equation 4. GPU Productivity Formula.

3 Related Work

Research on benchmarking and evaluating hardware for Deep Learning training has expanded considerably in recent years, addressing aspects such as accelerator performance, scalability, and software efficiency. These studies have contributed important empirical evidence on how different architectures behave under diverse workloads. However, most existing works focus on direct performance comparisons and do not provide a structured methodology for analyzing hardware configurations beyond raw benchmark results. In this context, our study positions its contribution by introducing a systematic approach that integrates variable selection, clustering, and performance-based ranking, offering a framework that complements and extends the insights provided by prior research. These works can be categorized into several groups based on their primary focus.

3.1 Training and Performance Benchmarks

Our work focuses on the training phase, specifically on evaluating GPU performance through a structured methodology rather than through isolated performance comparisons. Similarly, [24] focus on the scalability of training ResNet-50 on Intel Xeon CPU architectures. This study uses the MLPerf Training benchmark specifically for ResNet-50, providing performance data exclusively for CPUs in DL tasks. [25], broadens the scope of comparison by including specific TPU architectures alongside GPUs and CPUs. This work offers a broader perspective on the performance of different hardware platforms across various deep-learning tasks. However, it lacks diversity in GPU types and tests from different manufacturers. Several studies have focused on comparing the performance of CPUs and GPUs in specific deep-learning tasks. [26] Specifically evaluates performance in image classification tasks. [27] Extends this comparison to the context of big data, providing conclusions about how these architectures perform with large data volumes. [28] takes a different approach, focusing on performance profiling analysis for DL models on both platforms.

3.2 Optimized Architectures and Frameworks

Some works have focused on optimizing architectures for specific hardware. [29] Proposes a CNN architecture specifically optimized for GPUs, while [25] concentrate on energy efficiency for real-time object detection applications.

Regarding frameworks, [30] and [31] offer comparisons between different DL frameworks, with the former specifically focusing on GPU performance under software variations while training AlexNet.

In [32] authors combine the analysis of frameworks with performance comparisons on CPUs and GPUs, providing a comprehensive view of the DL tools ecosystem.

Table 1 Comparison of Related Works

Title	Hardware	Benchmark	Methodology	Time	Performance	Scalability
This work	GPUs	MLPerf Training (ResNet50)	✓	✓	✓	✓
[24]	Intel Xeon	MLPerf Training (ResNet50)	✗	✓	✓	✗
[29]	GPUs	Various	✗	✗	✗	✗
[25]	GPUs	Object detection	✗	✗	✗	✗
[26]	CPUs, GPUs	CPUs, GPUs Image classification	✗	✓	✓	✗
[31]	Not specified	Various	✗	✗	✓	✗
[30]	GPUs	AlexNet	✗	✓	✓	✗
[32]	CPUs, GPUs	Various	✗	✓	✓	✗
[33]	TPUs, GPUs, CPUs	Various	✗	✗	✓	✓
[27]	CPUs, GPUs	Various	✗	✗	✗	✗
[28]	CPUs, GPUs	Various	✗	✗	✗	✗

Note: ✓ indicates the feature is present, ✗ means it is not. DL, refers to whether DL techniques were used. Bold formatting can be added to highlight best-performing or most relevant results.

Table 1 demonstrates that our study stands out for its specific focus on using MLPerf Training as a tool to evaluate and compare different GPU configurations. Unlike previous works, our research proposes a systematic and reproducible method that integrates key metrics, such as training time, throughput, and GPU efficiency, enabling a standardized comparison of hardware performance. This study is, therefore, the first to establish a structured method for analysing and comparing accelerator configurations, highlighting not only the fastest configurations but also those that achieve an optimal balance between performance and computational efficiency. This provides a significant and practical contribution to the field of ML model training, facilitating decision-making in hardware selection for high-performance environments.

Although previous studies provide valuable analyses of hardware performance and scalability, no prior work has been identified that addresses this problem from a methodological standpoint. Existing research does not offer a structured framework for selecting relevant variables, grouping heterogeneous configurations, or ranking systems

based on efficiency and scalability. This gap underscores the need for the methodology proposed in this study.

4 State of the Problem

The MLPerf Training benchmark, widely recognized as a standard for evaluating DL performance, provides detailed public data [34] on the performance of various hardware configurations in training models such as ResNet-50. This model, used for image classification tasks, allows for an analysis of how performance varies as the number of GPUs scales in distributed systems.

The MLPerf Training Benchmark V3.1 dataset offers a wide range of features describing both the hardware components used and their performance in specific DL tasks. These features, as shown in **Table 2**, are essential for analysing the behavior of the different hardware configurations evaluated in the benchmark. In **Table 2**, each feature and its description can be observed individually.

Table 2 MLPerf Training Data Headers and Descriptions, based on [34]

Feature 1	Description
Public ID	Unique identifier for each system assessed.
Organization	Institution or company responsible for reporting system performance.
System Name	Specific name of the hardware system used in the evaluation.
Host Processor	Number of processors (CPUs) present in the system.
Host Processor Model Name	Model name of the CPU used, described verbatim.
Accelerators Per Node	Number of graphics accelerators (GPUs) available per node.
Accelerator Model Name	Specific GPU model used, described verbatim.
Latency (in Minutes)	Time in minutes required to fully train the ResNet-50 model using the ImageNet dataset.

Note: This table summarizes key features reported in the MLPerf Training results, as described in [34]. Descriptions follow the structure from the official dataset schema.

Although observing the natural behavior of the variables in [34], where training times are highlighted as key for detailed performance analysis, we calculated classical metrics to study in detail how the quantity and type of hardware influence scalability, identifying relevant patterns for optimizing distributed systems. This information is enriched with the calculation of throughput, as defined in **Equation 3**, which in this document represents the estimated number of images processed per second during model training. This value was calculated using the size of the ImageNet dataset and the total training time reported by each system. The calculation of productivity per GPU, as defined in **Equation 4**, represents the ratio between total productivity and

the number of GPUs used in the system. Speedup, defined in **Equation 1**, measures the acceleration factor achieved when using GPUs compared to a reference system without GPUs. Finally, efficiency, defined in **Equation 2**, indicates the proportion of the achieved speedup relative to the number of GPUs, showing how well the system scaled compared to an ideal model. **Figure 1**, shows the relationship between the number of GPUs and speedup on a logarithmic scale to highlight system behavior. The X-axis represents the number of GPUs, with the last point corresponding to a configuration with more than 3,000 GPUs. The Y-axis indicates speedup, where higher values reflect greater acceleration. The blue line represents the theoretical linear speedup, while the plotted points demonstrate a loss of efficiency as the number of GPUs increases, diverging from the ideal line. Configurations with 8 GPUs exhibit behavior closer to maximum productivity per GPU, whereas systems with a higher number of GPUs show lower-than-expected speedup. This demonstrates that if smaller configurations maintained their throughput as they scaled, the speedup achieved in larger systems would be significantly higher, underscoring the need to identify optimal configurations that maximize the efficiency and scalability of the hardware used.

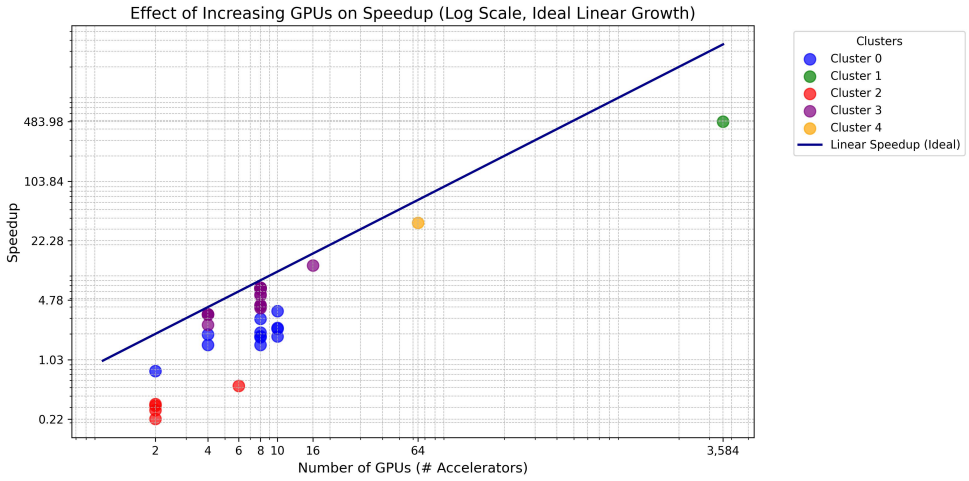


Fig. 1 Relationship between the number of GPUs and Speedup. Own elaboration

A preliminary analysis of the public MLPerf Training V3.1 data for the ResNet set reveals that in many configurations, doubling the number of GPUs does not proportionally reduce training time, as would be expected under an ideal model of linear scalability. For instance, while configurations with 8 GPUs can achieve training times close to 13 minutes, increasing to 16 GPUs does not necessarily result in a significant

reduction in time, highlighting a scalability issue. This issue is likely due to factors such as communication overhead, hardware saturation, and limitations in software architecture, which begin to have a significant impact as the system scales.

5 Proposed Methodology

As we mentioned previously, this work proposes a methodology aiming to provide guidance, as observed in Section 4, where the increase in GPUs and the lack of a proportional reduction in training time highlight the complexity of factors influencing scalability, including communication overhead, hardware saturation, and software architectural limitations. In response to this situation, a methodology has been designed to compare various hardware configurations and analyse their impact on the performance and efficiency of training the ResNet-50 DL model.

The proposed methodological strategy seeks not only to identify patterns and key variables but also to classify configurations based on their behavior, with the goal of guiding the selection of the most suitable environments for scalable training of networks like ResNet in high-performance computational scenarios. This study employs a proposed methodology structured into three phases (see **Figure 2**) to evaluate and compare the performance of different hardware configurations in DL model training. Using data from the MLPerf Training Benchmark V3.1, the objective is to understand how hardware variables influence the efficiency and scalability of training. This approach is divided into three main phases, each designed to extract key insights and construct a comprehensive and reproducible analysis. Part A of **Figure 2** shows a quick overview of the process. Part B of **Figure 2** goes into detail about the actions performed in each phase, which are described below. In the first phase, an exhaustive analysis of the variables described in the Problem Statement section was conducted to identify those with a significant impact on system performance and efficiency. An iterative process was employed to evaluate the importance of each variable. A random forest model was used to calculate the relative importance of each feature. In each iteration, one variable was selected as the prediction target while the others acted as predictors. Importance was determined based on its impact on reducing the model's mean squared error (MSE). The importance values obtained for each variable were accumulated across iterations, allowing the identification of those with a significant impact in multiple contexts. Variables whose accumulated importance exceeded a threshold of 0.5 were selected as the most relevant for further analysis. In the second phase, hardware configurations were grouped based on similarities in their characteristics and performance, facilitating the identification of common patterns. The selected variables were normalized using the RobustScaler method, which is resistant to outliers. The k-means clustering algorithm was applied after evaluating several clustering methods, including agglomerative clustering, Gaussian mixture models, and spectral clustering. Each result was validated using internal validation metrics such as the Silhouette Coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index. The optimal number of clusters was determined to be five, optimizing the representation of relationships and patterns in the data.

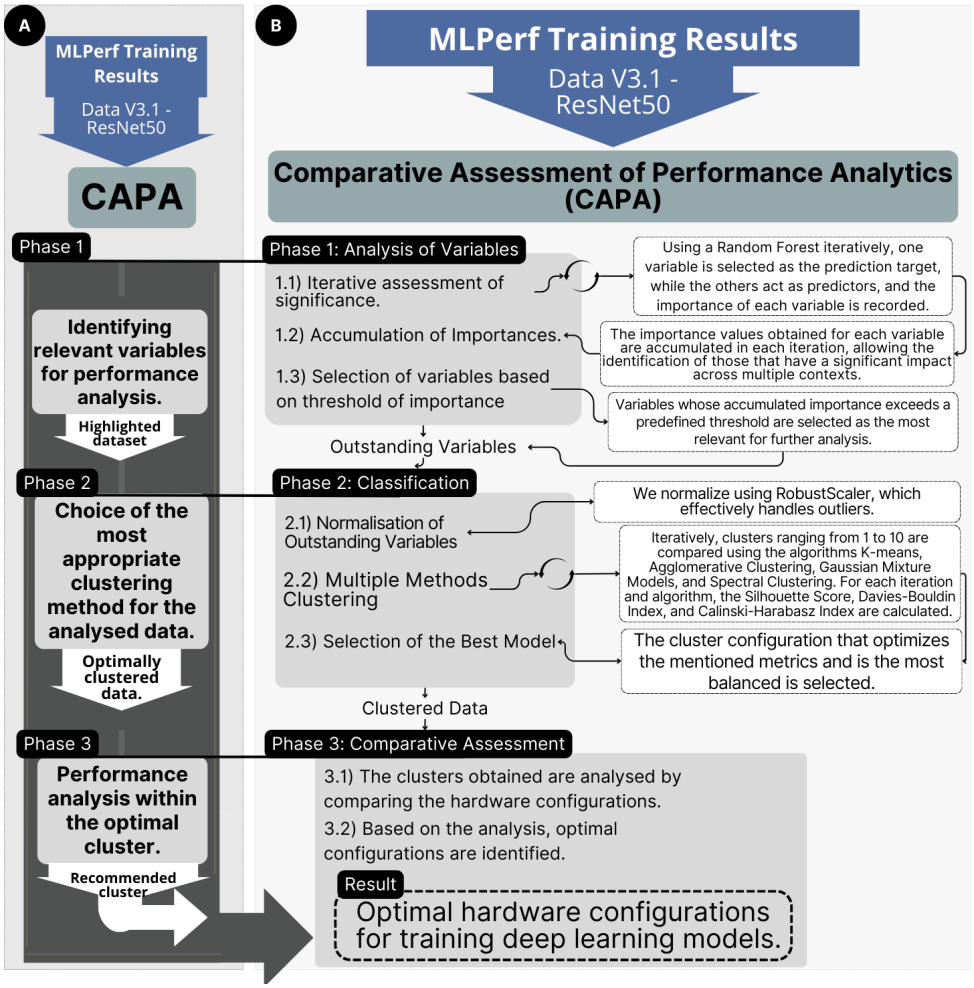


Fig. 2 Overview of the CAPA methodology. The process is divided into three main phases. 1) Analysis of variables using iterative Random Forest-based importance assessment to select the most relevant features; 2) Classification through normalization and application of multiple clustering algorithms, with model selection based on internal validation metrics; and 3) Comparative assessment of clusters to identify optimal hardware configurations for training DL models. Own elaboration

In the third phase, the results were integrated to perform a detailed analysis of the impact of hardware configurations on training performance. Key metrics such as Productivity, Speedup, and Efficiency were evaluated. These metrics were analysed both individually and across clusters, identifying similarities and differences in the behavior of the configurations.

6 Results

AApplying the proposed methodology, which included the identification of relevant variables in Phase 1 and subsequent clustering with k-means in Phase 2, five clusters (0 to 4) were obtained. These clusters group the hardware configurations evaluated in the MLPerf Training Benchmark V3.1 for the ResNet dataset. These clusters summarize patterns of performance, scalability, and productivity per GPU, highlighting how different hardware configurations affect the relationship between the number of accelerators and training time.

Table 3 Example of Featured Configurations per Cluster

Cluster	System	GPUs	Time (m)	Equation 4	Speedup	Efficiency
0	Fujitsu PRIMERGY-CDI-A100x10-mxnet (A100-PCIe)	10	∼24.57	86.90	3.60	0.36
1	NVIDIA coreweave (H100-SXM5)	3584	0.183	32.56	483.98	0.14
2	xFusion FusionServer 2288H V7 6x NVIDIA L4	6	170.38	20.89	0.52	0.0866
3	Supermicro SYS-821GE-TNHR (8x H100-SXM5)	8	13.55	196.91	6.53	0.82
4	NVIDIA Eos_n8 (64x H100-SXM5)	64	2.50	133.20	35.36	0.55

Note: Times are expressed in minutes. Efficiency and speedup are computed as per Equation 4. Decimal commas have been replaced with periods for LaTeX formatting consistency. Highlighting (e.g., bold) can be added for best performance.

Table 3 presents specific examples to illustrate the typical behavior of each cluster. Cluster 3, for instance, demonstrates high efficiency (0.82), while Cluster 1, despite achieving a massive speedup, exhibits much lower efficiency (0.14) due to the enormous number of GPUs used. This reinforces the idea that scaling up the number of GPUs is not always necessary to improve performance; optimizing hardware characteristics and reducing sources of inefficiency can lead to more significant improvements.

6.1 Cluster Analysis

Figure 3, illustrates the relationship between efficiency, training time, and speedup, grouping systems into clusters based on their performance. Divided into three graphs, the left graph shows training time on the X-axis and efficiency on the Y-axis. It can be observed that configurations in Cluster 3 and Cluster 4 achieve higher levels of efficiency with shorter training times, positioned in the upper-left corner. Conversely, Cluster 2 and Cluster 1 exhibit longer training times and considerably lower efficiency levels, suggesting inferior performance compared to other configurations.

In this first graph, the clusters appear somewhat mixed. However, in the central and right graphs, the introduction of speedup as a variable on the Z-axis allows for

better visualization of the relationship between efficiency, training time, and speedup. Configurations in Cluster 3 show high efficiency and moderate speedup but with lower training times. In contrast, Cluster 2 is characterized by low levels of efficiency and speedup, coupled with higher training times, indicating less optimal configurations. These graphs highlight the behavior and distinguish the different groupings.

Cluster 0: This cluster includes configurations with accelerators such as NVIDIA A100-PCIe (in pairs, quadruples, and octuplets), A30, L40, and L40S. The number of GPUs ranges from 2 to 10. Training times decrease as the number of accelerators increases, from approximately 115 seconds with 2 A100-PCIe GPUs to about 24 seconds with 10 A100-PCIe GPUs. However, although speedup improves with the increase in GPUs (reaching around 3.6 with 10 GPUs), efficiency (Speedup/#GPUs) remains moderate, ranging from 0.2 to 0.49 at best. Productivity per GPU also increases but does not reach exceptional values. This suggests significant gains compared to a non-GPU environment but falls short of achieving perfect linear scalability: while greater speed is achieved with more GPUs, each additional GPU yields diminishing returns. Cluster 1: This cluster contains a single extreme configuration: 3584 H100-SXM5-80GB GPUs. Despite achieving an extremely high speedup (around 484), efficiency is low (0.135). This indicates that while the absolute minimum training times are achieved, the marginal contribution of each additional GPU is relatively small. This demonstrates that over-scaling can reduce overall time but with very limited per-GPU efficiency. Cluster 2: This cluster includes systems with less powerful accelerators or less optimized configurations (e.g., A10G, RTX A5000, L4) using between 2 and 6 GPUs. Training times range from 170 to 400 seconds, and speedup rarely exceeds 0.5 to 0.6. Efficiency is also quite low (below 0.2 in most cases). This indicates clear limitations in GPU utilization, whether due to communication bottlenecks, relatively low computational power, or other environmental constraints. This cluster shows the lowest relative performance and scalability. Cluster 3: This group comprises systems with next-generation GPUs (NVIDIA H100-SXM5, H100-PCIe, and Intel Gaudi2) using a moderate number of GPUs (4, 8, or 16). Training times range from 35 seconds to as low as 7.5 seconds. Productivity per GPU is very high (in the range of 117-200 ops/sec/GPU), speedup varies from 2.5 to 11.7, and efficiency approaches or even

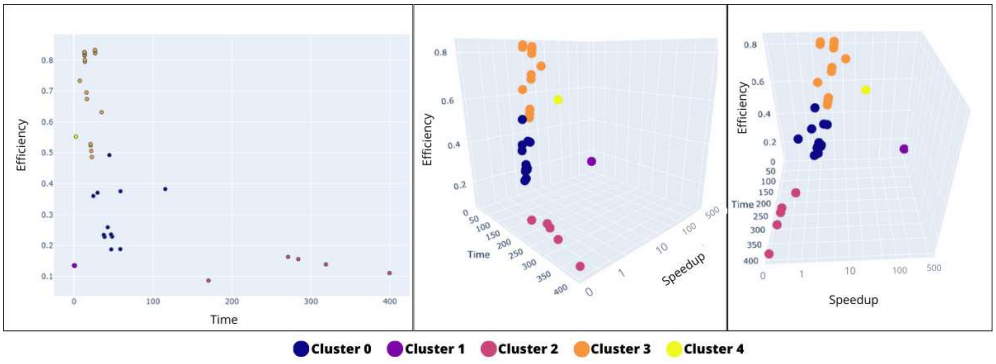


Fig. 3 Clustering and behavior of Calculated Metrics

exceeds 0.8 in configurations with 8 GPUs. These data confirm that Cluster 3 achieves an excellent balance between speed and efficient use of each accelerator. Although it does not always achieve the absolute lowest time (as seen in Cluster 4 with more GPUs), its high efficiency indicates that if this ratio were maintained when scaling up more GPUs, it would require fewer accelerators than larger clusters to achieve comparable times. This cluster is the one that "does more with less." Cluster 4: With 64 H100-SXM5 GPUs, Cluster 4 achieves one of the shortest training times (2.5 seconds) and a high speedup (35.36). However, its efficiency (0.55) is not as high as the best cases in Cluster 3. This indicates that while having more GPUs drastically reduces time, each additional GPU contributes less to the overall improvement compared to the optimized configurations of Cluster 3. Cluster 4 demonstrates the capability to reach absolute minimum training times but at the cost of slightly lower per-GPU efficiency.

Table 4 Approximate Average Metrics per Cluster.

Cluster	Predominant GPU Type	Range of GPU Quantity	Avg. Time (min)	Harmonic Mean of ProduGPU	Speedup Harmonic Mean	Efficiency Harmonic Mean
0	A100-PCIe, A30, L40/L40S	2 to 10	44.3	∼77	∼1.99	∼0.32
1	H100-SXM5	3584	<1	∼32	∼484	∼0.135
2	A10G, RTX A5000, L4	2 to 6	288	∼30	∼0.3	∼0.1
3	H100-SXM5, H100-PCIe, Gaudi2	4 to 16	19	∼157	∼4.6	∼0.6
4	H100-SXM5 (large scale)	64	2.5	∼133	∼35	0.55

Note: Values are averaged per cluster grouping. Time in minutes, ProduGPU represents productivity per GPU. Harmonic means used to better reflect relative ratios across clusters.

6.2 Optimal Cluster

Finally, once the significant performance patterns of the various hardware configurations evaluated were identified, as shown in **Table 4**, one of the clusters emerged as particularly noteworthy. This cluster, Cluster 3, groups systems that, despite not having the highest number of GPUs or the fastest in terms of theoretical computing capacity, achieve remarkably high productivity. Cluster 3 is characterized by configurations with a moderate number of GPUs, generally between 4 and 16, capable of processing a high number of images per second during the training of the ResNet-50 model. This moderate scale avoids the adverse effects typically associated with massive increases in accelerators (as seen in Cluster 4) while outperforming the efficiency and performance of configurations with fewer GPUs or previous generations (Clusters 0, 1, and 2). The GPUs in this cluster, H100-SXM5 (in both SXM5 and

PCIe variants) and Intel Gaudi2 represent the latest generation of accelerators, offering enhanced computing power, memory bandwidth, and improved integration with the rest of the system. Additionally, Cluster 3 systems incorporate high-end CPUs (Intel Xeon Platinum, AMD EPYC), which enhance the capacity to feed and coordinate workloads directed to the GPUs. The analysis focused on Cluster 3 reveals a highly favourable scenario regarding the balance between invested hardware resources and achieved performance. Systems in this cluster, featuring state-of-the-art accelerators such as NVIDIA H100-SXM5-80GB GPUs and Intel Gaudi2, exhibit substantially shorter training times compared to configurations with less advanced hardware. Moreover, these systems demonstrate high productivity per GPU, significant speedup, and outstanding efficiency. These results suggest that the computational capacity of each accelerator is utilized almost optimally, indicating a remarkable proximity to the ideal of linear scalability. Cluster 3 comprises systems with a moderate number of GPUs, typically between 4 and 16 units, paired with high-performance processors (CPUs). This highlights a carefully achieved hardware-software balance. Unlike scenarios where increasing the number of GPUs massively fails to yield proportional improvements in training time, the configurations in this cluster manage to "do more with less." This means that by maintaining the current efficiency while scaling to more accelerators, fewer GPUs would be required compared to larger configurations (such as those in Cluster 4) to achieve similar training times. Consequently, this approach reduces not only equipment costs but also infrastructure complexity and overall energy consumption. The efficiency achieved in Cluster 3 supports the hypothesis that performance does not depend solely on the quantity of resources but on the quality, architecture, and optimizations adopted in the computing environment. This cluster, as a whole, becomes a benchmark: it represents a balance between hardware investment, productivity, training speed, and GPU efficiency. The results suggest that with an optimal combination of next-generation accelerators, powerful processors, appropriate network topologies, and highly optimized software, it is possible to approach near-ideal scalability, offering a clear direction for future improvements and training system designs. This balance appears to be key to maximizing training performance without incurring penalties associated with communication and synchronization in large-scale systems.

7 Discussion

It is important to note that the proposed methodology, despite providing a structured process for selecting, grouping, and ranking hardware configurations, also faces certain limitations derived from the characteristics of the input data and the assumptions made during analysis. The effectiveness of the clustering and ranking stages depends directly on the quality, consistency, and representativeness of the performance metrics available. Differences in system configuration, software environments, and reporting practices can introduce variability that the methodology cannot fully control. Additionally, the feature selection process relies on the measurable variables exposed in each dataset; therefore, aspects such as interconnect topology, memory bandwidth behavior, or software level optimizations although highly relevant-remain outside the scope

of the current approach. These factors suggest that while the methodology is generalizable, its results must be interpreted in light of the data sources and contextual conditions under which the performance metrics were collected.

The results obtained in this study reveal significant phenomena associated not only with the number and type of GPUs used but also with the quantity and capacity of CPUs in conjunction with the system architecture. While GPUs are essential for accelerating the intensive mathematical operations required for training computer vision models such as ResNet-50, the role of CPUs and the overall hardware configuration should not be underestimated.

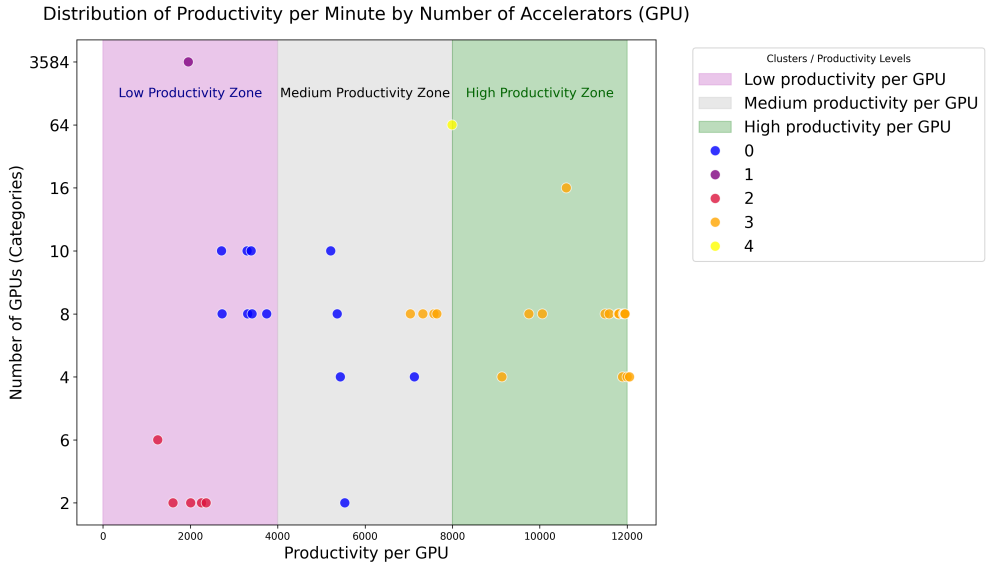


Fig. 4 Comparison between GPU Quantity and Productivity Per GPU, for each System. Own elaboration

Figure 4 illustrates the relationship between the number of GPUs and productivity per GPU across the different clusters. The highlighted cluster maintains higher productivity per GPU compared to the others, especially when the number of GPUs is low or moderate. This suggests that investing in a balanced configuration with fewer but better-optimized GPUs can be more beneficial than merely increasing the number of GPUs without considering the implications for scalability and efficiency.

In addition, **Figure 4**, shows that the system configuration with the highest number of GPUs achieves a per-GPU productivity comparable to that of slower configurations. When classifying the results into three equidistant ranges (low, medium, and high throughput per GPU), it becomes evident that the initially fastest system fails to maintain a significant advantage when considering throughput relative to the number of GPUs. In other words, when evaluating throughput per graphical acceleration unit, configurations with more GPUs do not necessarily outperform the slower

systems. This finding highlights the importance of considering not only the absolute speed of the system but also the efficiency with which the available hardware resources are utilized. One of the key aspects identified is the interaction between CPUs and GPUs during training. CPUs are responsible for crucial tasks such as data preparation and preprocessing, memory management, and coordinating the training processes. In systems with a high number of GPUs but limited CPU performance, bottlenecks arise in supplying data to the GPUs. This results in underutilization of the GPUs, which remain idle while waiting for the CPUs to provide the necessary data. Therefore, a proper balance between the computational capabilities of CPUs and GPUs is essential to optimize overall system performance.

Moreover, system architecture plays a critical role in training efficiency. The speed and bandwidth of interconnections between CPUs and GPUs, such as PCIe and NV Link, directly affect the speed at which data can be transferred and processed. Systems with advanced architectures that provide high-speed communication between components can mitigate data transfer delays, improving GPU utilization and, consequently, overall productivity.

Specifically, in training vision models with images, large volumes of data require efficient handling. The size of image datasets, such as ImageNet, implies that the storage system and input/output (I/O) subsystem must support high read and write rates. If storage performance is inadequate, bottlenecks arise that limit the flow of data to the CPUs and, subsequently, to the GPUs.

To address these challenges, it is crucial to implement strategies to optimize the data pipeline. Techniques such as parallel preprocessing, multithreading for loading and decoding images, and the use of optimized data formats can significantly reduce data preparation time. Additionally, employing high-speed storage and distributed file systems can improve I/O performance, ensuring a steady flow of data to the GPUs.

Another relevant factor is the choice of GPU architecture. GPUs specifically designed for DL applications, which include tensor cores and support for mixed precision, offer substantial performance improvements. These features allow matrix operations and convolutions to be executed more efficiently, accelerating the training of computer vision models.

The findings also highlight that simply increasing the number of GPUs is not always advantageous. As more GPUs are added, performance gains may diminish due to communication and synchronization overhead between devices. This is particularly true for computer vision models, where operations require high inter-device data communication. Therefore, optimizing existing configurations and improving communication efficiency between GPUs can be more beneficial than scaling up the number of GPUs.

As part of the proposed methodology, thresholds were defined heuristically to determine variable importance. This value can be adjusted and could have been related in another way, such as using the mean of the resulting importance.

One of the main limitations of this study is the inability to make direct modifications to the evaluated hardware, as the data used is provided by each organization. Without physical access to the machines a researcher, it is not possible to analyse how small variations in hardware configurations, such as reducing memory, adjusting the

CPU-to-GPU ratio, or optimizing other system components, affect training performance and efficiency. This restriction limits the study to the reported configurations, preventing the exploration of fine-tuning adjustments that could improve the balance between performance and resource usage. Consequently, the results reflect an analysis based on available data, without the possibility of conducting controlled experiments to evaluate specific configurations.

We acknowledge the importance of including comparisons with AMD GPUs, among others, in future research and plan to address this issue in subsequent studies.

An important factor in the scalability and efficiency of GPU-based systems is the role of interconnections, such as NVLink, PCIe, and Infinity Fabric, which influence data transfer speed and overall system performance. However, this study focuses on hardware performance under standardized conditions, and a direct evaluation of these interconnects was not included in the analysis. We acknowledge this as a limitation and consider it beyond the scope of this work, but we recognize its relevance for future research.

8 Conclusions

In this work, we proposed a methodology leveraging clustering techniques to group configurations according to their underlying characteristics and enables a comprehensive analysis of performance and efficiency metrics. To illustrate the proposed methodology, we conducted a comparison of multiple hardware configurations employed for training the ResNet-50 model, emphasizing on the scalability constraints encountered in systems comprising a large number of GPUs. As a result, patterns have been identified that provide a better understanding of the relationships between hardware and training time, offering concrete guidelines for the selection and optimization of computational environments designed to accelerate large-scale DL.

From the obtained results, we can conclude that, optimal performance in training computer vision models with images does not solely depend on GPU power but rather on a balanced synergy between CPUs, GPUs, and system architecture. Considering factors such as resource balance, data transfer efficiency, and the specific capabilities of GPUs is essential to overcome the scalability limitations observed. These insights are crucial for designing high-performance computing systems that maximize productivity and efficiency in artificial intelligence applications.

As a future work, the proposed methodology can be applied to other datasets and workloads published by the MLPerf Training benchmark. The application of the proposed methodology will allow testing any type of scenario, improving the methodology by each iteration. Moreover, this will contribute to a better understanding of scalability and efficiency in distributed DL training. Additionally, comparing the behavior of systems under different workloads may help identify optimal configurations that perform well across a variety of tasks.

Declarations

- Funding: No funding was received to assist with the preparation of this manuscript.

- Conflict of interest/Competing interests: The authors declare that they have no competing interests.
- Ethics approval and consent to participate: Not applicable.
- Consent for publication: Not applicable.
- Data availability: The datasets analyzed during the current study are available in the MLPerf Training repository
- Materials availability: Not applicable.
- Code availability: Not applicable.
- Author contribution: All authors contributed equally to the conception, analysis, and writing of this manuscript. All authors read and approved the final version.

References

[1] Cottier, B., Rahman, R., Fattorini, L., Maslej, N., Besiroglu, T., Owen, D.: The rising costs of training frontier ai models (2024)

[2] Bermejo, B., Juiz, C.: Improving cloud/edge sustainability through artificial intelligence: A systematic review. Journal of Parallel and Distributed Computing **176**, 41–54 (2023)

[3] Mattson, P., Cheng, C., Diamos, G., et.al: Mlperf training benchmark. In: Dhillon, I., Papailiopoulos, D., Sze, V. (eds.) Proceedings of Machine Learning and Systems, vol. 2, pp. 336–349 (2020)

[4] Molero, X., Juiz, C., Rode no, M.: Evaluación Y Modelado del Rendimiento de Los Sistemas Informáticos, 1st edn. Pearson Educación London, ??? (2004)

[5] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**, 2278–2323 (1998) https://doi.org/10.1109/5.726791 . Deep Learning

[6] Wang, S.-C.: Artificial neural network. Interdisciplinary Computing in Java Programming, 81–100 (2003) https://doi.org/10.1007/978-1-4615-0377-4_5

[7] Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 2015 521:7553 **521**, 436–444 (2015) https://doi.org/10.1038/nature14539

[8] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2016-December**, 770–778 (2015) https://doi.org/10.1109/CVPR.2016.90 . Micro ResNet

[9] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., Ghayvat, H.: Cnn variants for computer vision: History, architecture, application, challenges and future scope. Electronics 2021, Vol. 10, Page 2470 **10**, 2470 (2021) https://doi.org/10.3390/ELECTRONICS10202470 . CNN, Vision importancia

[10] O'Shea, K., Nash, R.: An introduction to convolutional neural networks. International Journal for Research in Applied Science and Engineering Technology **10**, 943–947 (2015) https://doi.org/10.22214/ijraset.2022.47789

[11] Deng, J., Dong, W., al, R.S.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE Computer Society, ??? (2009). https://doi.org/10.1109/CVPR.2009.5206848 . https://doi.org/10.1109/CVPR.2009.5206848

[12] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet. https://www.image-net.org/

[13] Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals of Data Science 2015 2:2 **2**, 165–193 (2015) https://doi.org/10.1007/S40745-015-0040-1

[14] MacQueen, J.: Some methods for classification and analysis of multivariate observations. books.google.comJ MacQueenProceedings of the fifth Berkeley symposium on mathematical (1965)

[15] Ackermann, M.R., Blömer, J., Kuntze, D., Sohler, C.: Analysis of agglomerative clustering. Algorithmica **69**, 184–215 (2014) https://doi.org/10.1007/S00453-012-9717-4/FIGURES/15

[16] McNicholas, P.D., Murphy, T.B.: Model-based clustering of microarray expression data via latent gaussian mixture models. Bioinformatics **26**, 2705–2712 (2010) https://doi.org/10.1093/BIOINFORMATICS/BTQ498

[17] Ng, A., Jordan, M., information, Y.W.-A., 2001: On spectral clustering: Analysis and an algorithm. proceedings.neurips.ccA Ng, M Jordan, Y WeissAdvances in neural information processing systems (2001)

[18] Shahapure, K.R., Nicholas, C.: Cluster quality analysis using silhouette score. Proceedings - 2020 IEEE 7th International Conference on Data Science and Advanced Analytics, DSAA 2020, 747–748 (2020) https://doi.org/10.1109/DSAA49011.2020.00096

[19] Davies, D., Bouldin, D.: A cluster separation measure. ieeexplore.ieee.org (1979)

[20] Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Taylor and FrancisT, Statistics-theory and Methods (1974)

[21] Wang, X., Xu, Y.: An improved index for clustering validation based on silhouette index and calinski-harabasz index. IOP Conference Series: Materials Science and Engineering **569**, 052024 (2019) https://doi.org/10.1088/1757-899X/569/5/052024

[22] Rigatti, S.: Random forest. meridian (2017)

[23] Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T.: Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics **8** (2007) https://doi.org/10.1186/1471-2105-8-25

[24] Wang, W., Hasabnis, N.: Distributed mlperf resnet50 training on intel xeon architectures with tensorflow. ACM International Conference Proceeding Series, 29–35 (2021) https://doi.org/10.1145/3440722.3440880

[25] Lee, Y., Hwang, J.-w., Lee, S., Bae, Y., Park, J.: An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection (2019)

[26] Jayasimhan, A., Pabitha, P.: A comparison between cpu and gpu for image classification using convolutional neural networks. 2022 International Conference on Communication, Computing and Internet of Things, IC3IoT 2022 - Proceedings (2022) https://doi.org/10.1109/IC3IOT53935.2022.9767990

[27] Baykal, S.I., Bulut, D., Sahingoz, O.K.: Comparing deep learning performance on bigdata by using cpus and gpus. 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting, EBBT 2018, 1–6 (2018) https://doi.org/10.1109/EBBT.2018.8391429

[28] Gyawali, D.: Comparative Analysis of CPU and GPU Profiling for Deep Learning Models (2023). https://arxiv.org/abs/2309.02521v3

[29] Ridnik, T., Lawen, H., Noy, A., Baruch, E.B., Sharir, G., Friedman, I.: TResNet: High Performance GPU-Dedicated Architecture (2021)

[30] Kim, H., Nam, H., Jung, W., Lee, J.: Performance analysis of cnn frameworks for gpus. ISPASS 2017 - IEEE International Symposium on Performance Analysis of Systems and Software, 55–64 (2017) https://doi.org/10.1109/ISPASS.2017.7975270

[31] Shatnawi, A., Al-Bdour, G., Al-Qurran, R., Al-Ayyoub, M.: A comparative study of open source deep learning frameworks. 2018 9th International Conference on Information and Communication Systems, ICICS 2018 **2018-January**, 72–77 (2018) https://doi.org/10.1109/IACS.2018.8355444

[32] Lerat, J.S., Mahmoudi, S.A., Mahmoudi, S.: Single node deep learning frameworks: Comparative study and cpu/gpu performance analysis. Concurrency and Computation: Practice and Experience **35**, 6730 (2023) https://doi.org/10.1002/CPE.6730

[33] Wang, Y.E., Wei, G.-Y., Brooks, D.: Benchmarking TPU, GPU, and CPU Platforms for Deep Learning (2019). https://arxiv.org/abs/1907.10701v4

[34] Mattson, P., Cheng, C., et.al, C.C.: MLCommons Training Tableau Public (bit.ly/42qPl7O). https://public.tableau.com/app/profile/data.

visualization6666/viz/shared/C4G8MKRBR