Efficiency of Machine Learning Tasks on HPC Devices

G. Promyslov[1], A. Efremov[1], Y. Ilyasov[1],
V. Pisarev[1,2][0000−0002−9004−1839], and A. Timofeev[2,1][0000−0003−1156−893X]

[1] HSE University, Moscow, Russia
[2] Joint Institute for High Temperatures of RAS, Moscow, Russia

Abstract. Accurate benchmarking is critical for selecting computing architectures optimized for machine learning (ML) tasks. Conventional benchmarks such as High-Performance Linpack (HPL) and High Performance Conjugate Gradients (HPCG) often fail to capture the diversity and complexity of modern ML workloads. This study investigates the correlation between hardware parameters (e.g., processor architecture, cache size, frequency) and ML performance across various devices, including CPUs and accelerators. By studying a wide range of ML tasks, we identify key performance bottlenecks and explore whether a correlation-based approach can guide the selection of optimal hardware for an entire class of ML tasks. Our findings offer practical recommendations for future computing architectures and advance the efficient use of high-performance computing for diverse ML applications.

Keywords: HPC · Machine learning · Benchmarking · Performance Profiling · Hardware Efficiency

1 Introduction

Selecting an optimal computing architecture plays a pivotal role in attaining highest performance computationally-intensive tasks. Traditional benchmarks, such as High-Performance Linpack (HPL) and High Performance Conjugate Gradients (HPCG), have long been standards for evaluating supercomputer performance [7,6]. However, because they primarily focus on solving systems of linear equations, they do not fully representative for other types of workloads. In high-performance computing (HPC), this shortfall becomes particularly critical, as resource utilization in supercomputing environments must be maximized [8,22]. A particular domain that will be explored in this study is the ML workloads, whose unique characteristics include large-scale matrix operations, neural network optimization, and intensive data exchange.

Modern ML applications have become increasingly prevalent across diverse domains, ranging from big-data analytics to computer vision and natural language processing [11]. In the last decade, ML and deep learning methods are gaining popularity in scientific-oriented domains, such as computation chemistry [12,10], molecular simulations [3,16,23], materials design and discovery [18].

The growing popularity of ML has been accompanied by a proliferation of hardware platforms designed to accelerate computations, including graphics processing units (GPUs), tensor processing units (TPUs), and specialized accelerators [15]. This variety of hardware solutions heightens the need for more precise, ML-oriented evaluation criteria and benchmark suites that can determine which devices offer the most efficient execution of specific tasks.

Despite the abundance of benchmarks and testing frameworks, an unresolved question remains: can the correlations between hardware characteristics and ML performance be leveraged to select optimal equipment for an entire class of ML tasks? In this work, we hypothesize that such correlation analysis will not only provide a more objective assessment of performance but also inform the design of future computing systems optimized for ML workloads.

The main objective of this study is to develop and validate a correlation-based benchmarking method for analyzing hardware characteristics when running various ML tasks and obtain recommendations for computer elements and architecture which are optimal for ML class tasks.

Potential applications of these findings include data center procurement recommendations, the architectural planning of supercomputers for specific ML workloads, and the optimization of cloud-based platform configurations.

The remainder of this paper is organized as follows. The next section describes the methodology and the set of ML tasks employed in the study. We then present our performance analysis results and discuss the identified correlations. The final section summarizes our conclusions and provides recommendations for designing future computing systems aimed at ML workloads.

2 Related work

High-Performance Computing (HPC) has traditionally focused on large-scale simulations and data-intensive workloads using multi-core Central Processing Units (CPUs) and Graphics Processing Units (GPUs). In recent years, a substantial body of research has examined how the characteristics of these hardware components correlate with the performance of Machine Learning (ML) tasks, especially in the context of deep neural networks and gradient-boosting algorithms [6]. Early investigations in this area concentrated on establishing how memory bandwidth, core count, and heterogeneous computing architectures influence throughput and model convergence times. For example, studies have demonstrated that increasing core counts can reduce training time for convolutional neural networks (CNNs), but diminishing returns often arise beyond a certain threshold due to memory contention [24]. These findings underscore the importance of balancing compute and memory resourcesa concept formalized in Roofline models that capture performance limitations imposed by arithmetic intensity and memory bandwidth [24].

A variety of tools and benchmarks have been employed to assess the computational efficiency of ML tasks. Traditional HPC performance metrics such as floating-point operations per second (FLOPS) are now coupled with additional

ML-specific measures like accuracy, throughput, and timed epochs. Profilers (e.g., NVIDIA Nsight, Intel VTune) and platform-monitoring frameworks (e.g., PAPI, LIKWID) facilitate detailed performance tracing, allowing researchers to identify pipeline inefficiencies and bottlenecks [4]. Moreover, alternative evaluation methodologies such as Roofline analysis [24] and domain-specific benchmark suites (e.g., HPCG for high-performance conjugate gradient [6]) provide further insight into how architectural features and algorithmic structures intersect.

Benchmarking efforts specific to ML frameworks have grown substantially in recent years. TensorFlow [2], PyTorch [21], and XGBoost [5] are among the most widely used packages in both academic and industrial settings. Comparative studies have investigated scaling behaviors, GPU utilization, memory usage, and precision trade-offs (e.g., FP32 vs. FP16) to determine optimal configurations [2,21]. Much of this work relies on representative workloads such as image classification or language modeling for measuring throughput and latency. In parallel, community-driven benchmark suites like MLPerf [9,19] and AI Benchmark [14] emerged to provide standardized methodologies for evaluating end-to-end performance of ML pipelines. These suites incorporate tasks that stress different aspects of hardware and software stacks, such as dense and sparse computation, or recurrent and convolutional layers.

Despite these advances, existing studies often adopt narrow task definitions or focus on a limited subset of performance metrics. For instance, they may primarily target training throughput while overlooking other critical factors like performance variability across different hardware vendors or the interplay between HPC-style modeling tasks and intensive ML workloads. Furthermore, the emphasis on standard neural network architectures (ResNet, Transformer, etc.) in many benchmarking efforts can make it challenging to generalize findings to more specialized or emerging ML applications. Another frequently cited shortcoming is the lack of integrated analysis methods that combine hardware monitoring, energy consumption data, and application-level performance metricsa combination essential for holistic evaluations in HPC environments.

In this paper, we address these deficiencies by proposing an elaborated benchmarking approach that jointly considers ML tasks alongside a set of physics-based kernels within a single HPC context. Unlike prior work, our methodology tracks both application-level metrics (e.g., iteration time, solution accuracy) and low-level hardware characteristics (e.g., memory throughput, instruction mix) to capture a more comprehensive view of resource utilization. We also extend existing performance analysis techniques by incorporating real-time profiling and cross-hardware comparisons (CPU vs. GPU vs. multi-GPU setups), providing insights into scaling efficiency for diverse workload categories. This dual focuson ML and physics-based tasksenables us to explore synergies and trade-offs not visible in ML-only or HPC-only studies. Consequently, our work contributes a novel perspective on how to optimize HPC infrastructures for both data-driven and simulation-driven applications simultaneously.

3 Methodology

As there is a large body of benchmarking data available for certain ML tasks, this study uses the hybrid methodology: original performance results are obtained for platforms available to the authors, and meta-analysis of the publicly available data is done for other platforms.

For evaluation, we explore several benchmark suites tailored for ML tasks, ai-benchmark [13], MLPerf [9,19], other test suites [1]. The methodology for individual benchmarks is presented in the next subsections.

3.1 Methodology for ai-benchmark

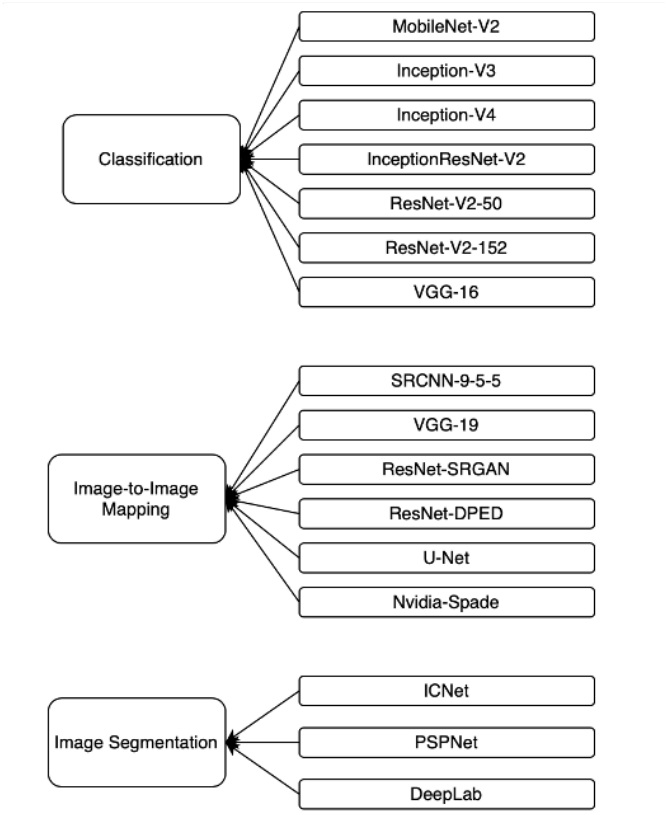


Fig. 1: Deep learning models used in ai-benchmark, divided into three main classes of problems

ai-benchmark is a set of test problems chosen to characterize the average performance of a platform for classification and computer vision tasks. That

benchmark consists of 19 different neural network models that cover most deep learning architectures used in popular tasks and allows to test both their training and inference processes. The models used in *ai-benchmark* are shown in Fig. 1 and are divided into three main classes and several subclasses. The main classes of problem-solving models are: models for object classification, models for image-to-image mapping and models for image segmentation.

For massively-parallel systems, this benchmark can be used to evaluate single-node performance. Testing with an *ai-benchmark* performance evaluation tools was performed on nodes of the supercomputer cHARISMa. That system was chosen due to the need to design and establish simple and common methods for launching, testing, and profiling in random complex environment. In Fig. 2 characteristics of node of the supercomputer cHARISMa are shown that were used for testing. The detailed characteristics and performance of those components will be shown later in the article alongside other models.

10 Type B Computing nodes	
Processor Model	2 x Intel Xeon Gold 6152 2.1-3.7 HHz (22 cores)
GPU Model	4 x NVIDIA Tesla V100 32 GB
RAM	**1536 GB**
Solid-state drive	2 x SSD 240 GB (RAID 1)
Network Adapter InfiniBand	2 x Mellanox 100Gb/s Infiniband Dual Port
Network Adapter Ethernet	Intel Ethernet 10G 4P X710/I350

Fig. 2: Supercomputer cHARISMa node Type B characteristics, on which the calculations for testing were performed

We design the method for launching the benchmark together with the procedure to test the performance of random system during benchmark launch and by using external software on the node configuration presented in Fig. 2.

The values of various metrics were also compared with their theoretical values over a given class of scientific problems to verify the assertion that belonging to a certain class of scientific problems will lead to similar performance results on different systems.

In this article we analyze only the metrics that directly affect the performance of the supercomputer components for the tasks of deep learning. They will be discussed later in details.

3.2 Methodology for MLPerf HPC Training

MLPerf HPC Training [9,19] is a specialized performance evaluation test designed to assess the computational capabilities of supercomputers in training

complex deep learning models. This standardized benchmarking suite is focused on high-performance computing and includes various scenarios that reflect key workloads in scientific and engineering research.

Scenarios of MLPerf HPC Training:

- CosmoFlow: Cosmological process modeling using convolutional neural networks. The primary computational workload involves processing large volumes of multidimensional data, requiring high memory bandwidth and significant computational resources for both CPUs and GPUs.
- DeepCAM: Satellite image analysis for climate research. This scenario employs convolutional neural networks for anomaly detection, imposing high demands on computational power and efficient utilization of parallel computing.
- OpenCatalyst: Chemical reaction modeling using density functional theory in combination with machine learning. The computations involve both traditional HPC algorithms and deep learning training, making this scenario computationally intensive.
- OpenFold: Protein structure prediction based on deep learning and transformer architectures. Introduced in MLPerf HPC Training v3.0, this scenario requires substantial computational resources, particularly from GPUs, due to intensive matrix multiplication operations.

The selection of these performance evaluation scenarios is justified by their scientific significance and high computational complexity. As MLPerf HPC Training has evolved, these scenarios have been adapted to the changing requirements of computing systems: CosmoFlow, DeepCAM, and OpenCatalyst have maintained continuity across versions, enabling performance trend analysis, while OpenFold has expanded the scope of tested workloads by incorporating more modern machine learning models.

MLPerf HPC Training was chosen as the primary evaluation tool because it is tailored for scientific computing and provides a comprehensive assessment of supercomputer performance in deep learning tasks. It is developed and maintained by MLCommons, ensuring its relevance and alignment with current HPC standards.

The performance analysis is based on data published on the official MLCommons website [20] and specifications of individual CPU and GPU models provided by manufacturers such as Intel, AMD, NVIDIA, and FUJITSU. This selection ensures data reliability, as the sources are authoritative and enable reproducibility of the analysis.

3.3 Methodology for other test suites

The data for this study was collected from https://openbenchmarking.org, a repository of Phoronix test suite benchmark results on various platforms. Specifically, we focused on machine learning (ML) test suites that provide performance metrics for different CPUs and GPUs. The benchmark results include metrics

such as inference or training time, which are critical for evaluating the performance of hardware in ML tasks.

Since benchmark results are expressed in different units of measurement, results were converted to either inverse seconds or units per second for the sake of consistency. Benchmark results data was also grouped or filtered based on the number of compute devices in the dataset or the benchmark scenarios. For CPUs following characteristics were considered for analysis — number of cores, base clock frequency, L3 cache size, memory bandwidth, peak FP32 performance. For GPUs — number of cores, clock frequency, memory bandwidth, peak FP32 performance.

3.4 Common part of methodologies

The following metrics were used to evaluate computational performance:

- System performance, expressed as the inverse execution time of a given task, independent of the contribution of CPUs or GPUs.
- Relative performance, representing the fraction of the inverse execution time that corresponds to the share of either CPU or GPU contribution to the overall system performance, proportional to the peak floating-point performance (FLOPS) of the respective device.
- Balance, calculated as the ratio of the peak performance sum (either for all CPUs or all GPUs) to the total memory bandwidth (main memory for CPUs or video memory for GPUs), ensuring a consistent comparison within the same type of computational resources.
- Average execution time, calculated as the arithmetic mean of execution times for a specific computational device characteristic.
- Potential peak Performance, computed as the product of the average execution time and the peak FLOPS performance of the given computing device.

Key hardware characteristics considered in the analysis:

- CPU: Number of cores, clock frequency, L3 cache size, memory bandwidth, peak FP32 performance.
- GPU: Number of cores, clock frequency, memory bandwidth, peak FP32 performance.

For data processing and analysis, aggregation methods were applied based on the logical grouping of datasets by benchmark version, scenario, computing device type and model. The software implementation of these principles accelerated the processing and facilitated a more detailed analysis and visualization of results. Data validation and cleaning were performed manually, with results corresponding to computing device models whose specifications could not be confirmed from reliable sources being excluded from the analysis.

4 Results

4.1 Results for ai-benchmark

The *ai-benchmark* score is calculated as geometric mean of the inverse runtime values of different models in the benchmark, so the higher the score is, the better model performed. For the convenience of the experiment, the averages among all presented models for which information was available are being calculated. Those averages are presented in Figures 3 and 5 by solid lines. We decided to divide results of CPU testing shown on the Figure 3 into two different classes depending of manufacturers: Intel or AMD. Tables with the data on the basis of which those graphs were constructed are presented later in the article.

Let us now directly show and discuss the results of CPU testing. Figure 3 represents the dependencies of score from chosen metrics: number of CPU cores, amount of threads, L3 Cache size (MB), theoretical performance (Flops) and recommended retail price (USD).

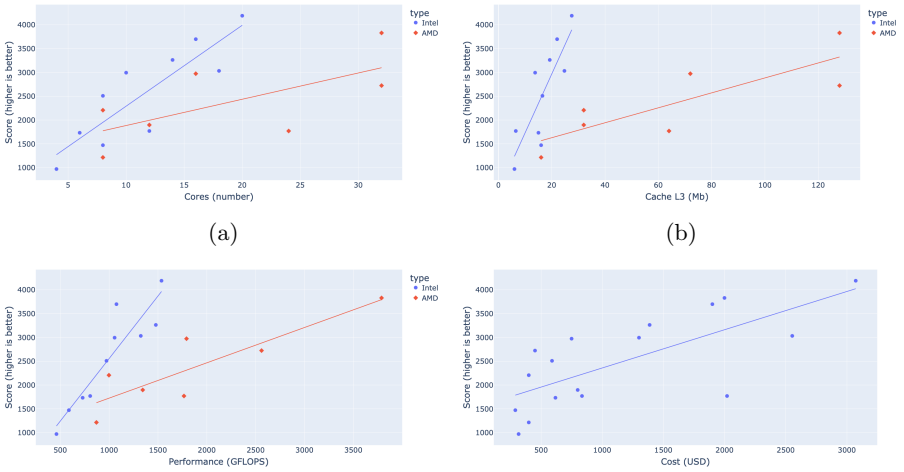


Fig. 3: Dependency of *ai-benchmark* score from the number of cores in CPU (a), the CPU L3 cache size (b), rated CPU FLOPS (c), CPU cost (d) (launch date prices were taken when possible). Red lines and symbols correspond to the AMD CPU models and blue lines and symbols to Intel CPU models

Figure 4 represents the existing dependency between L3Cache size and performance of different CPU models. That proves visible dependencies between figures shown at 3

Figure 5 represents the dependencies of score on GPU platforms from selected metrics: number of GPU cores, clock speed, memory bandwidth, peak perfor-

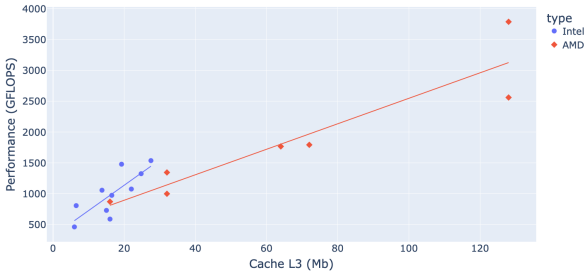
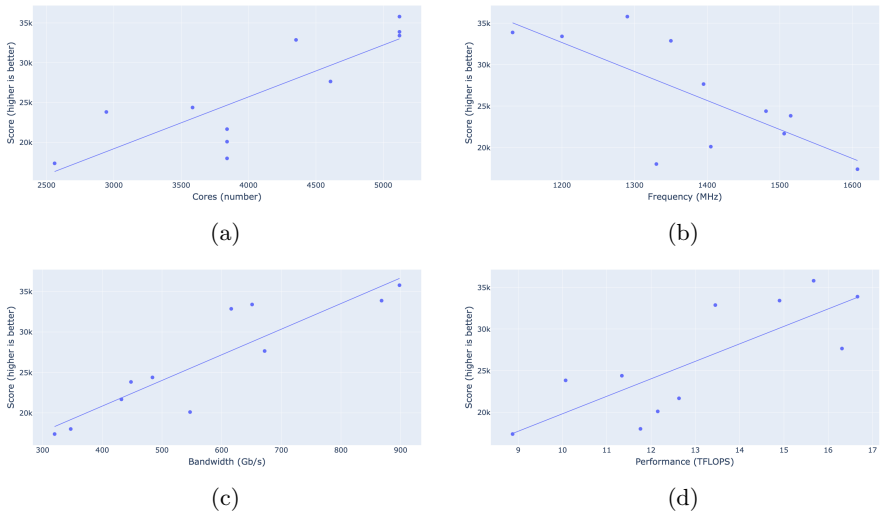


Fig. 4: Dependence of evaluated CPU performance from L3Cache size. Red line and symbols correspond to the AMD CPU models and blue line and symbols to Intel CPU models



(a) (b)

(c) (d)

Fig. 5: Dependency of *ai-benchmark* score from the number of GPU cores (a), the GPU clock speed (b), GPU memory bandwidth (c), GPU peak performance in FP32 (d)

mance in FP32 FLOPS in graphics processing unit (GPU). From the figures we see that the primary characteristic determining the benchmark performance on CPU the peak floating-point performance, suggesting that the problems chosen for the benchmark are compute-bound on the tested platforms. For GPU-accelerated platforms, the performance increases with the number of GPU cores, peak flops and memory bandwidth. However, those characteristics typically increase simultaneously with each new GPU generation, so that it's hard to decide what plays the most significant role.

To identify the bottleneck characteristic, we have conducted a roofline analysis for the benchmark running on a cHARISMa Type B node. Figure 6 shows the obtained result. We can see that the benchmark tasks (red-circled marker) lie within the compute-bound region, meaning that *ai-benchmark* tasks are compute-bound both on CPU-only and on CPU+GPU platforms. We should also note that the performance is significantly below the peak GPU performance, meaning that such tasks have a significant potential for improved performance by algorithmic changes.

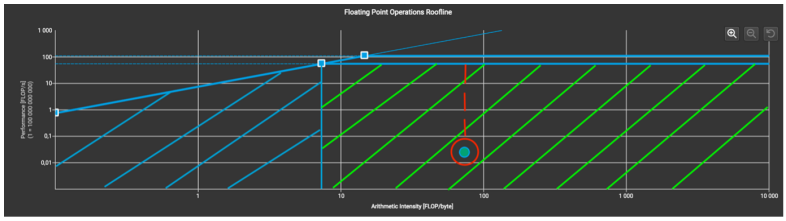
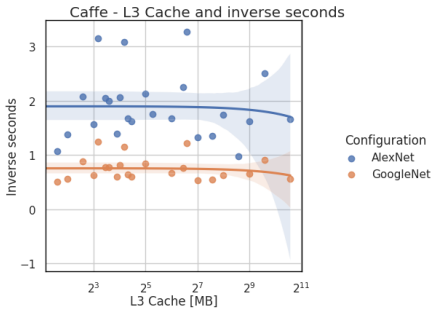


Fig. 6: Roofline analysis of *ai-benchmark* on cHARISMa Type B computing node. Blue shading denotes the bandwidth-bound zone, green shading denotes the compute-bound zone for NVidia V100 GPU. The red-circled marker is the position of *ai-benchmark*

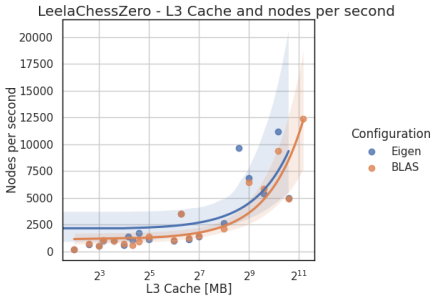
4.2 Results for other CPU test suites

This subsection provides graphs showing the dependencies between CPU specifications and benchmark results within each test suite.

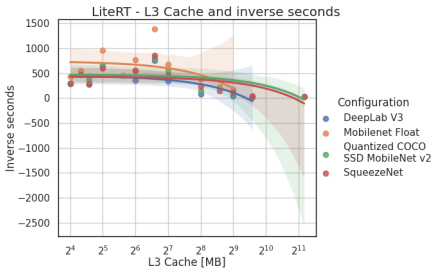
L3 Cache On the Figure 7 there are plots with results in benchmarks and cache sizes. In some packages, processors with larger L3-cache have better performance (e.g., LeelaChessZero, MLPack, Numenta and Whisper.cpp). However, it is difficult to distinguish the dependence between them.



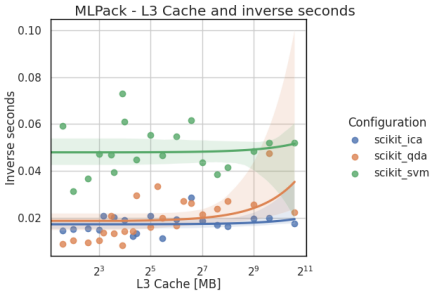
(a)



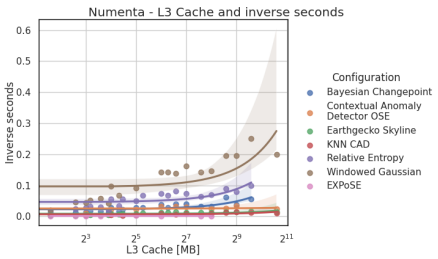
(b)


LiteRT - L3 Cache and inverse seconds
Inverse seconds
L3 Cache [MB]
Configuration
DeepLab V3
Mobilenet Float
Quantized COCO SSD MobileNet v2
SqueezeNet

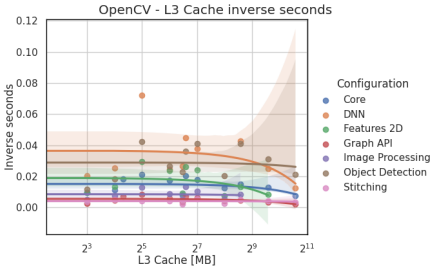
(c)


MLPack - L3 Cache and inverse seconds
Inverse seconds
L3 Cache [MB]
Configuration
scikit_ica
scikit_qda
scikit_svm

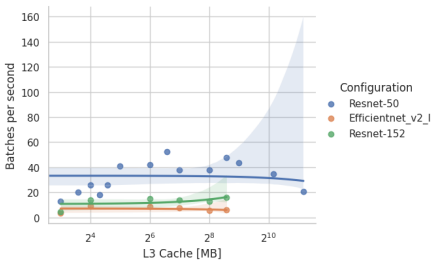
(d)


Numenta - L3 Cache and inverse seconds
Inverse seconds
L3 Cache [MB]
Configuration
Bayesian Changepoint
Contextual Anomaly Detector OSE
Earthgecko Skyline
KNN CAD
Relative Entropy
Windowed Gaussian
EXPoSE

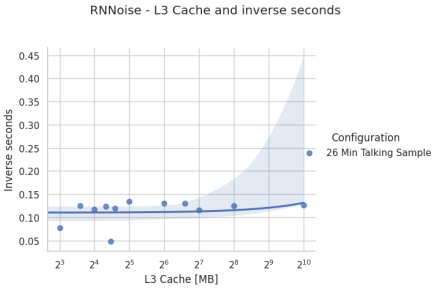
(e)


OpenCV - L3 Cache inverse seconds
Inverse seconds
L3 Cache [MB]
Configuration
Core
DNN
Features 2D
Graph API
Image Processing
Object Detection
Stitching

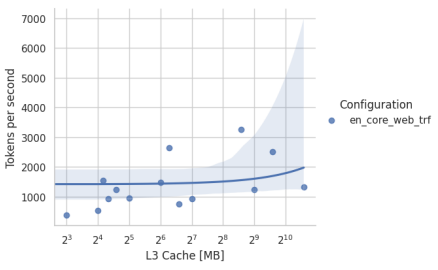
(f)


Pytorch - L3 Cache and batches per second
Batches per second
L3 Cache [MB]
Configuration
Resnet-50
Efficientnet_v2_l
Resnet-152

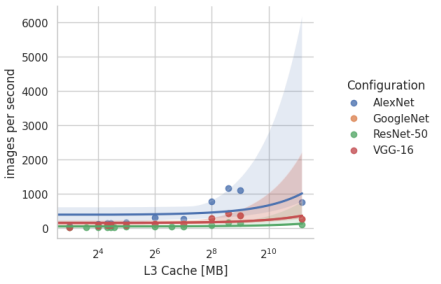
(g)


RNNoise - L3 Cache and inverse seconds
Inverse seconds
L3 Cache [MB]
Configuration
26 Min Talking Sample

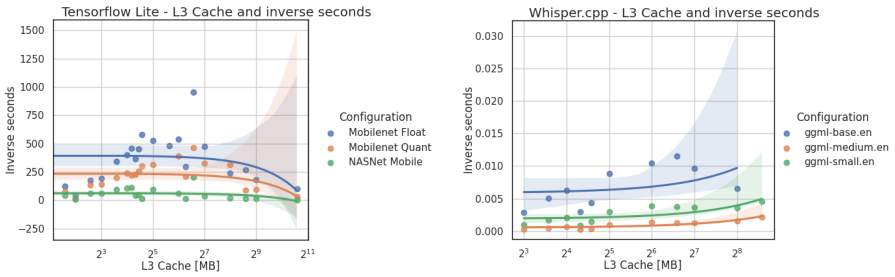
(h)


spaCy - L3 Cache and tokens per second
Tokens per second
L3 Cache [MB]
Configuration
en_core_web_trf

(i)


Tensorflow - L3 Cache and images per second
images per second
L3 Cache [MB]
Configuration
AlexNet
GoogleNet
ResNet-50
VGG-16

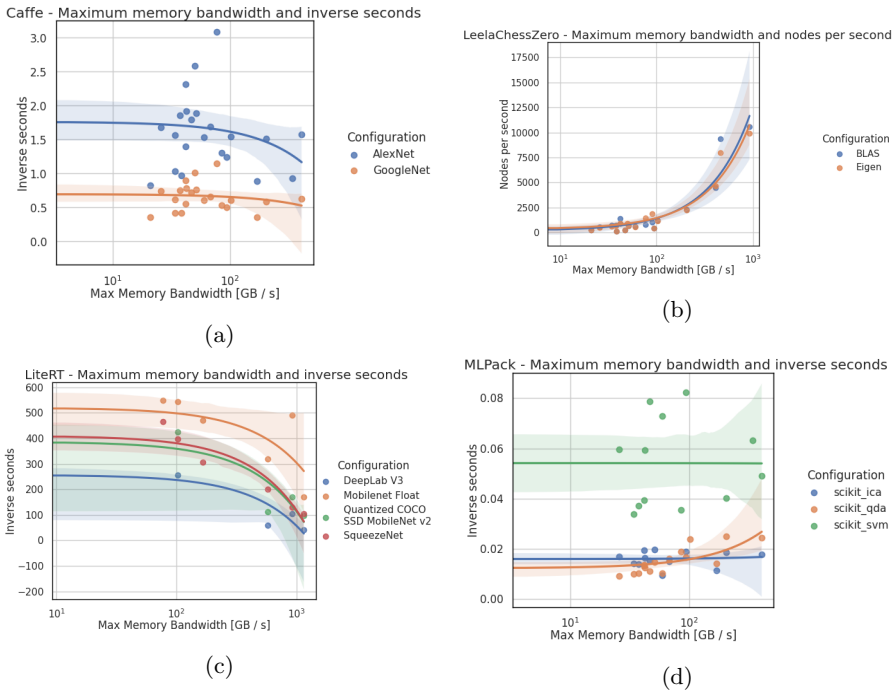
(j)



(k) (l)

Fig. 7: Dependency of CPU performance from L3-cache in ML tasks within each test suite

Memory Bandwidth On the Figure 8 there are plots with results in benchmarks and maximum memory bandwidth. In most test suites, larger memory bandwidth gives better performance in machine learning tasks.



(a) (b)

(c) (d)

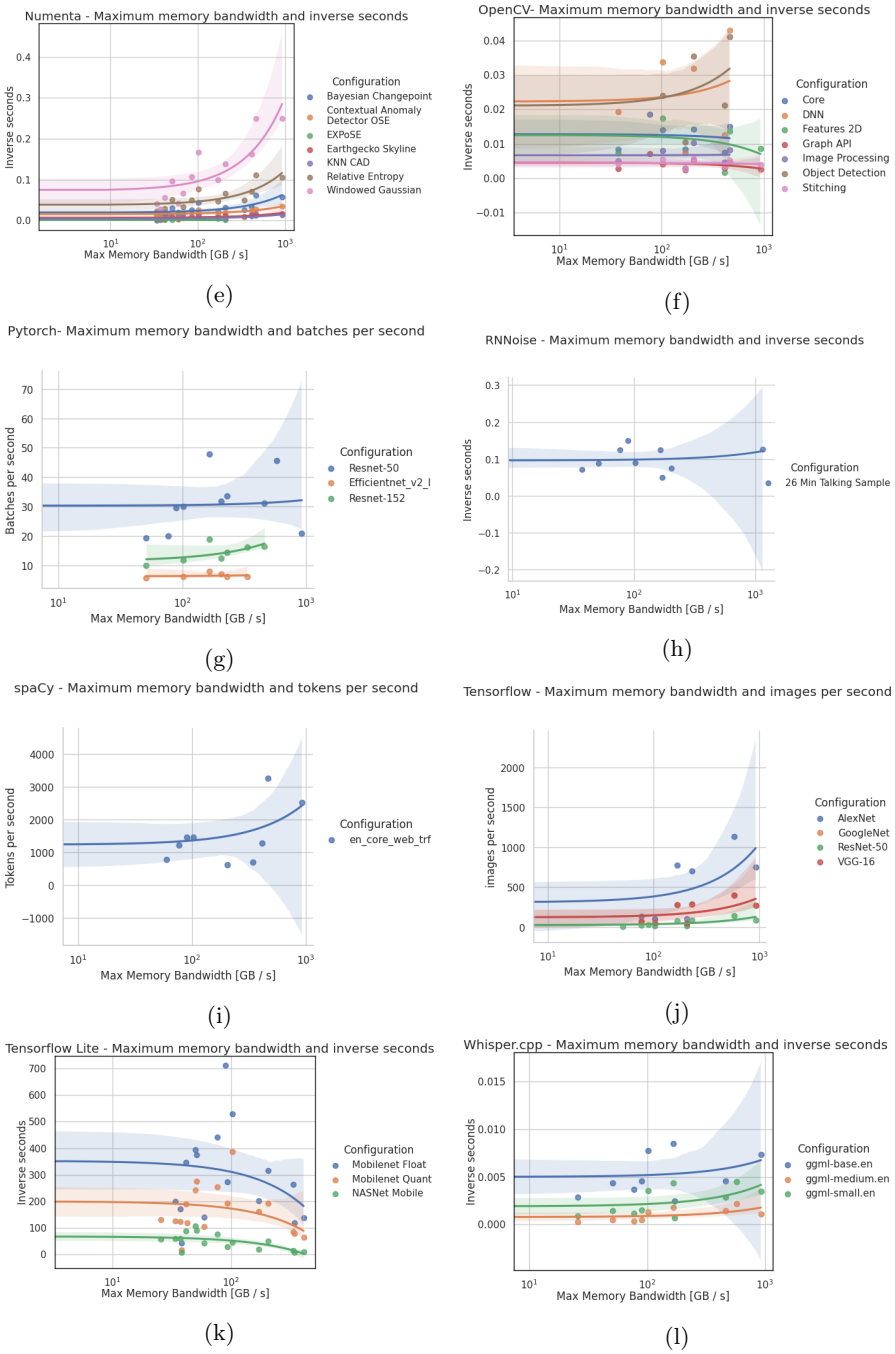
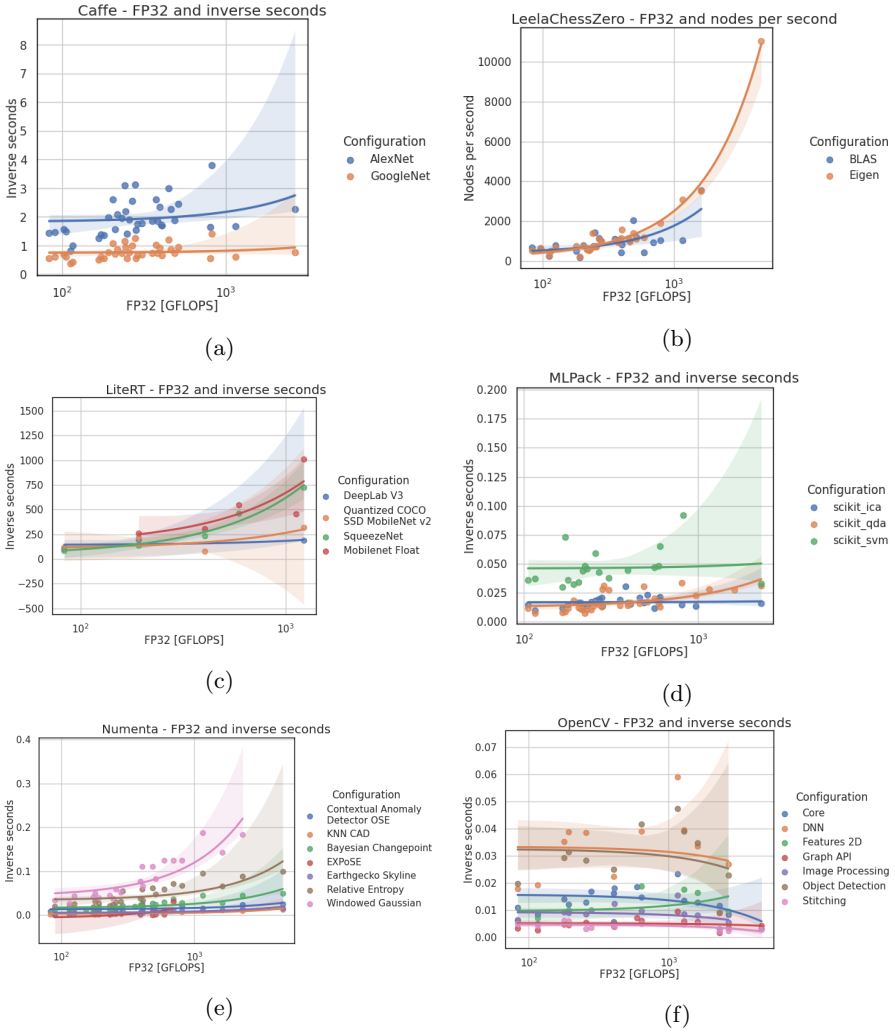


Fig. 8: Dependency of CPU performance from memory bandwidth in ML tasks within each test suite

FP32 Performance On the Figure 9 there are plots with results in benchmarks and FP32 performance. In most cases, processors with higher FP32 performance show better results in ML tasks.



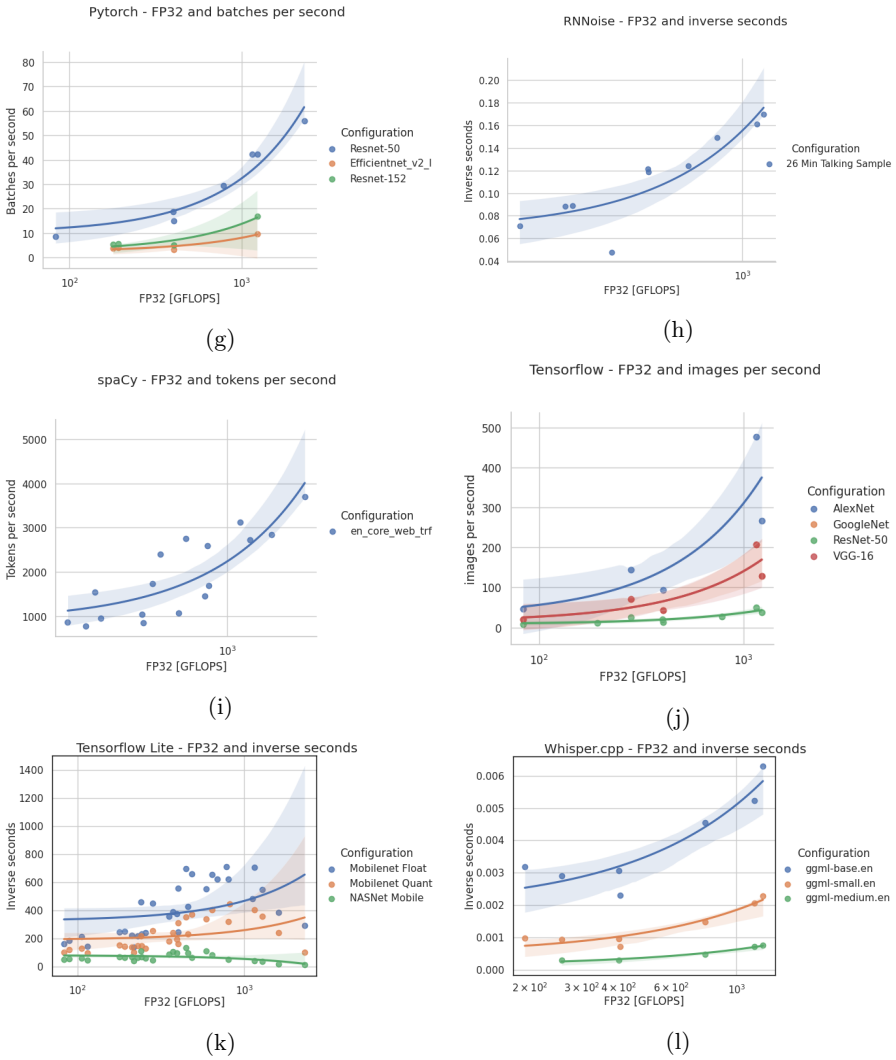
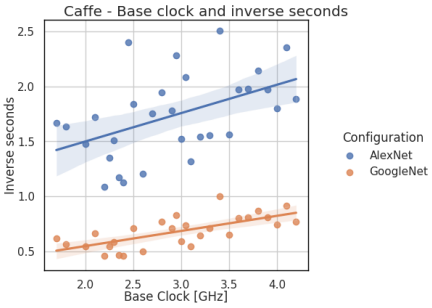
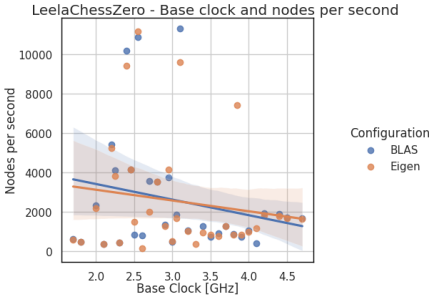


Fig. 9: Dependency of CPU performance from FP32 performance in ML tasks within each test suite

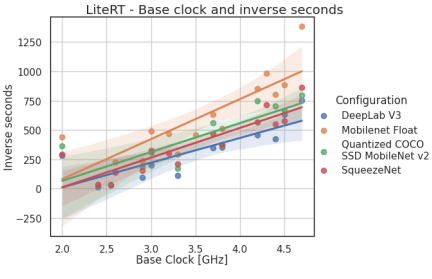
Base clock On the Figure 10 there are plots with results in benchmarks and base clock. In most cases higher clock rate gives better performance, but there are exceptions (e.g.Tensorflow and LeelaChessZero).


Caffe - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
AlexNet
GoogleNet

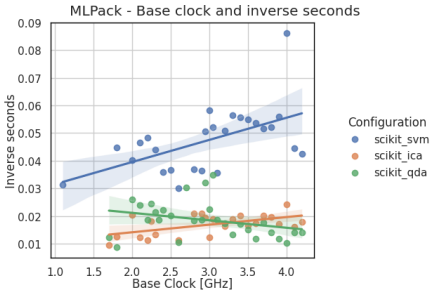
(a)


LeelaChessZero - Base clock and nodes per second
Nodes per second
Base Clock [GHz]
Configuration
BLAS
Eigen

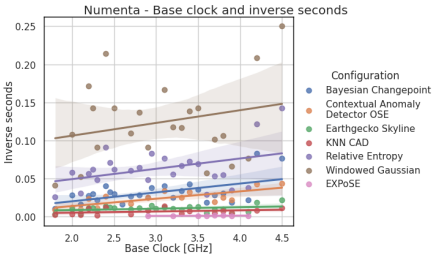
(b)


LiteRT - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
DeepLab V3
Mobilenet Float
Quantized COCO SSD MobileNet v2
SqueezeNet

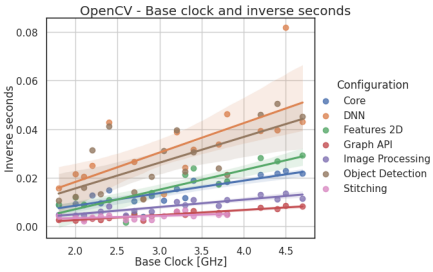
(c)


MLPack - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
scikit_svm
scikit_ica
scikit_qda

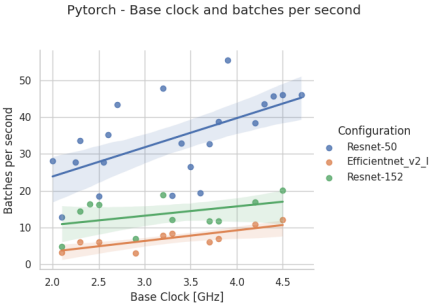
(d)


Numenta - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
Bayesian Changepoint
Contextual Anomaly Detector OSE
Earthgecko Skyline
KNN CAD
Relative Entropy
Windowed Gaussian
EXPoSE

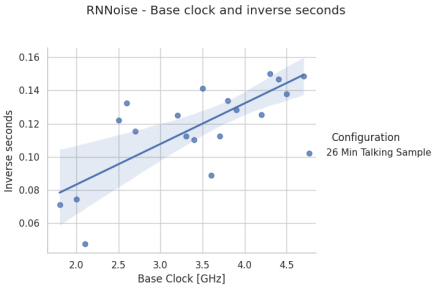
(e)


OpenCV - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
Core
DNN
Features 2D
Graph API
Image Processing
Object Detection
Stitching

(f)


Pytorch - Base clock and batches per second
Batches per second
Base Clock [GHz]
Configuration
Resnet-50
Efficientnet_v2_l
Resnet-152

(g)


RNNoise - Base clock and inverse seconds
Inverse seconds
Base Clock [GHz]
Configuration
26 Min Talking Sample

(h)

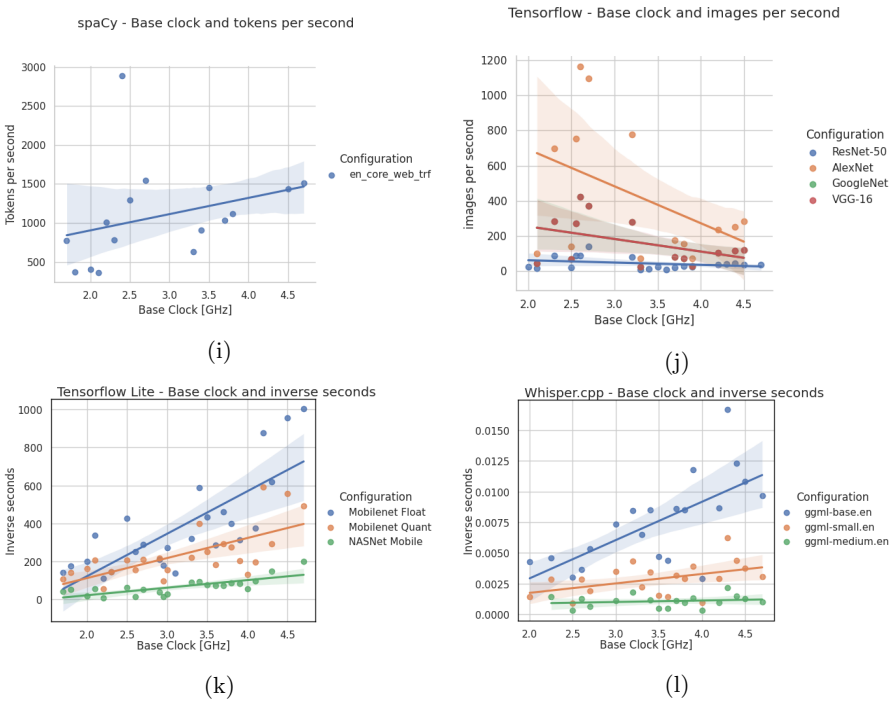
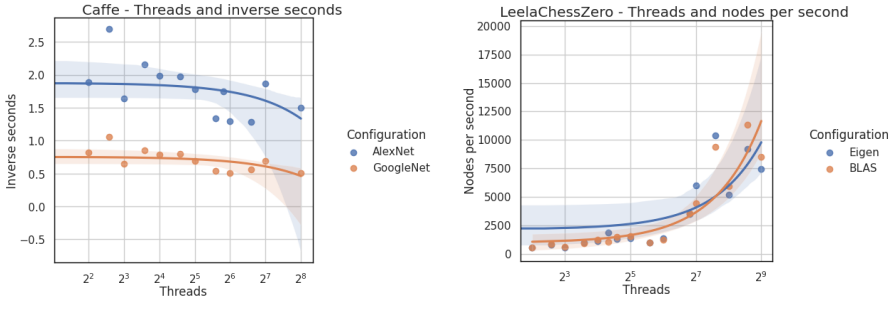
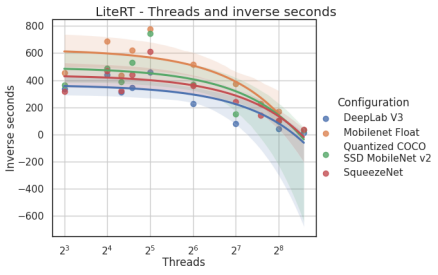


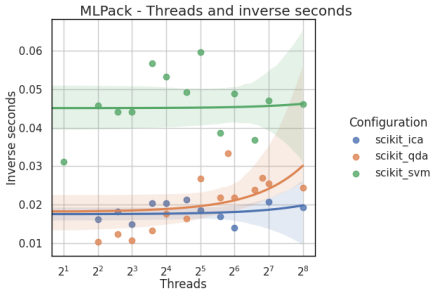
Fig. 10: Dependency of CPU performance from base clock in ML tasks within each test suite

Threads number On the Figure 11 there are plots with results in benchmarks and threads number. In some suites, more threads give better performance, but there are opposite cases. This may be due to the fact that the tests may have been run in single-threaded or multithreaded mode. However, not all suites specify testing mode.

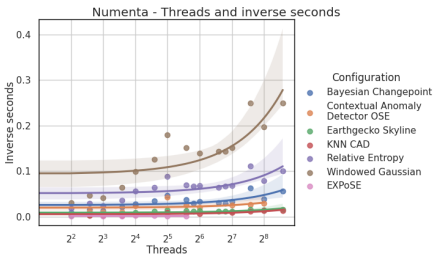



LiteRT - Threads and inverse seconds
Inverse seconds
Threads
Configuration
DeepLab V3
Mobilenet Float
Quantized COCO SSD MobileNet v2
SqueezeNet

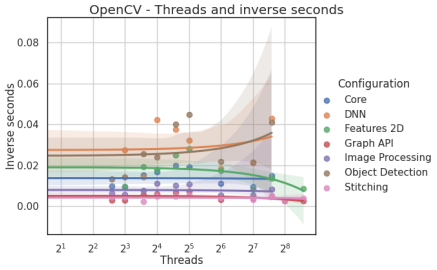
(c)


MLPack - Threads and inverse seconds
Inverse seconds
Threads
Configuration
scikit_ica
scikit_qda
scikit_svm

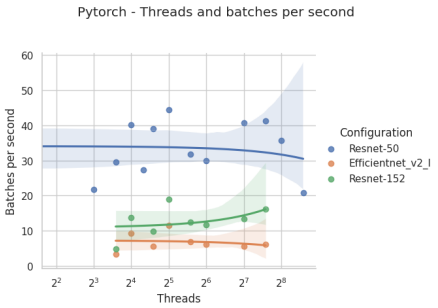
(d)


Numenta - Threads and inverse seconds
Inverse seconds
Threads
Configuration
Bayesian Changepoint
Contextual Anomaly Detector OSE
Earthgecko Skyline
KNN CAD
Relative Entropy
Windowed Gaussian
EXPoSE

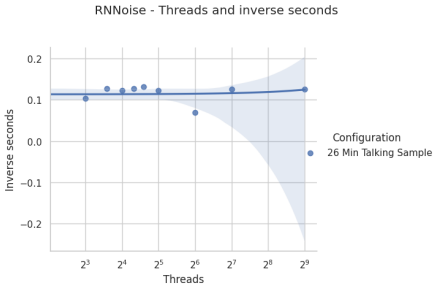
(e)


OpenCV - Threads and inverse seconds
Inverse seconds
Threads
Configuration
Core
DNN
Features 2D
Graph API
Image Processing
Object Detection
Stitching

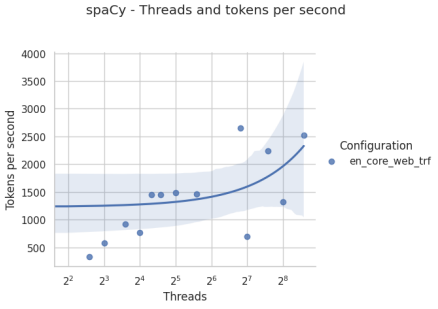
(f)


Pytorch - Threads and batches per second
Batches per second
Threads
Configuration
Resnet-50
Efficientnet_v2_l
Resnet-152

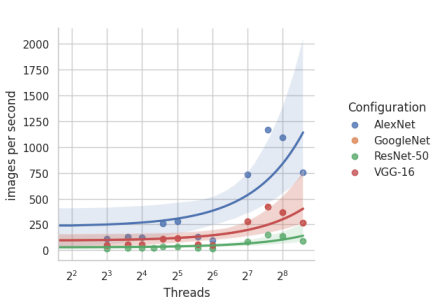
(g)


RNNoise - Threads and inverse seconds
Inverse seconds
Threads
Configuration
26 Min Talking Sample

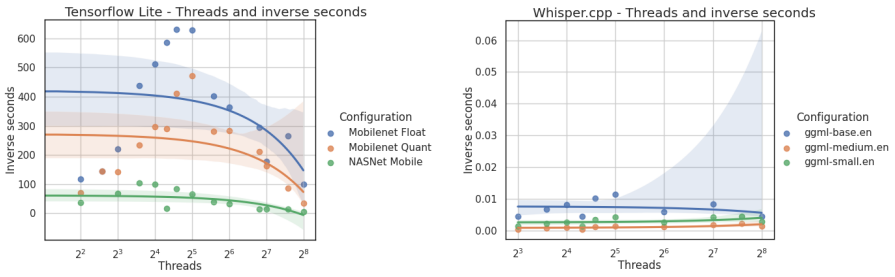
(h)


spaCy - Threads and tokens per second
Tokens per second
Threads
Configuration
en_core_web_trf

(i)


Tensorflow - Threads and images per second
images per second
Threads
Configuration
AlexNet
GoogleNet
ResNet-50
VGG-16

(j)

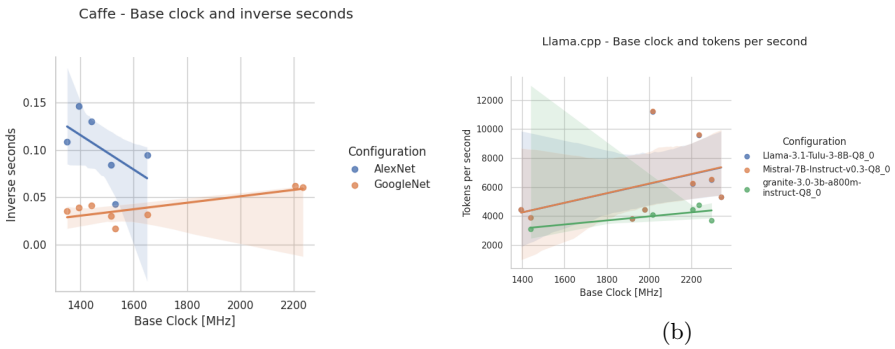


(k) (l)

Fig. 11: Dependency of CPU performance from threads number in ML tasks within each test suite

4.3 Results for other GPU test suites

This subsection provides graphs showing the dependencies between graphics card specifications and benchmark results within each test suite. When comparing GPU benchmark results in Caffe and in Llama.cpp, some dependencies are observed. For example, the higher the number of cores in GPU, the higher the performance in ML tasks. The same is true for clock rate, FP32 performance and memory bandwidth of GPU.

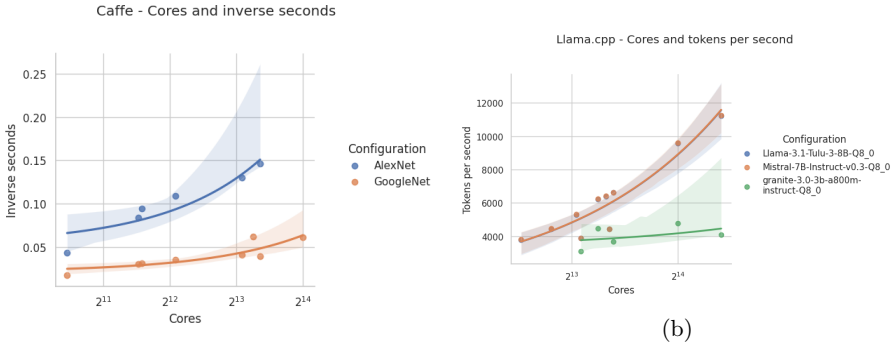


(a) (b)

Fig. 12: Dependency of GPU performance from cores number in ML tasks in Caffe and Llama.cpp suites

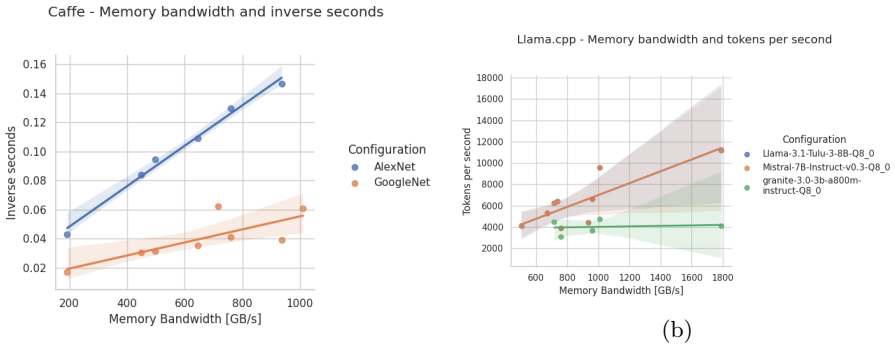
4.4 Results for MLPerf HPC Training

This section presents the key dependencies of computational device characteristics on the performance metrics defined in the previous section. The analysis



(a) (b)

Fig. 13: Dependency of GPU performance from clock rate in ML tasks in Caffe and Llama.cpp suites

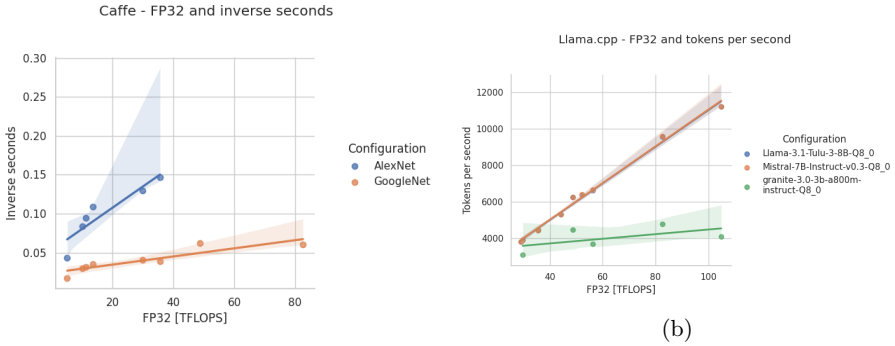


(a) (b)

Fig. 14: Dependency of GPU performance from memory bandwidth in ML tasks in Caffe and Llama.cpp suites

includes graphical representations illustrating the relationships between CPU and GPU parameters. The presented graphs provide a comparative evaluation of computing platforms used in MLPerf HPC Training benchmarks, demonstrating performance scaling trends across different supercomputing configurations.

The open dataset provided by MLCommons [20] contains performance measurements obtained from supercomputers, which may either rely solely on CPU resources or employ a hybrid configuration combining both CPU and GPU components. Due to the limited number of runs conducted exclusively on CPUs, a comprehensive analysis based solely on such data would be insufficient. Therefore, in the presented diagrams, these data points are considered alongside results from hybrid configurations. For hybrid systems, an approximate share of CPU performance is estimated based on the ratio of FLOPS contributions within the system. Similarly, in the case of GPUs, all recorded runs inherently involve a hy-

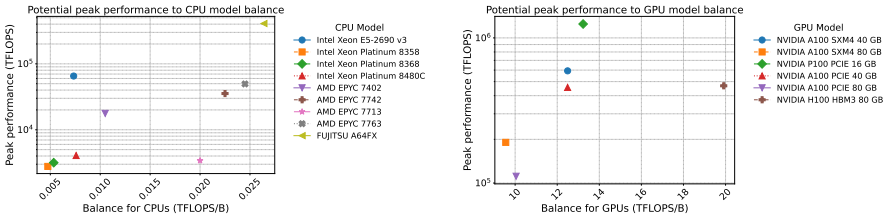


(a) (b)

Fig. 15: Dependency of GPU performance from FP32 performance in ML tasks in Caffe and Llama.cpp suites

brid setup utilizing both CPU and GPU resources. Thus, an analogous approach is applied to estimate the GPUs contribution to the overall system performance.

Fig. 16 presents performance diagrams for CPU and GPU, showing their dependency on the balance metric. This metric is defined as the ratio of computational performance (for CPU or GPU) to the peak memory bandwidth of the respective device. It is important to note that, due to the lack of exact memory bandwidth data for the devices used in the performance test, an assumption was made that the maximum possible bandwidth, based on the specifications provided by the manufacturers for computing devices, can be used as a proxy. This assumption may affect the precision of the balance calculations but provides a reasonable baseline for comparative analysis.



(a) (b)

Fig. 16: Dependencies of computing device model potential peak performance to balance metric based on results gained for all MLPerf HPC Training scenarios for CPU (a), for GPU (b)

The diagrams on the Fig. 17 showing the dependencies of CPU potential performance on the balance, now divided by the MLPerf HPC Training scenarios

help to highlight more clearly the relationships between CPU performance and memory bandwidth. The data is aggregated across all MLPerf HPC Training versions, from v1.0 to v3.0. The scenario-specific separation may be crucial, as different scenarios impose varying levels of computational demand on the system. This division allows for a more precise understanding of how each scenario affects CPU performance.

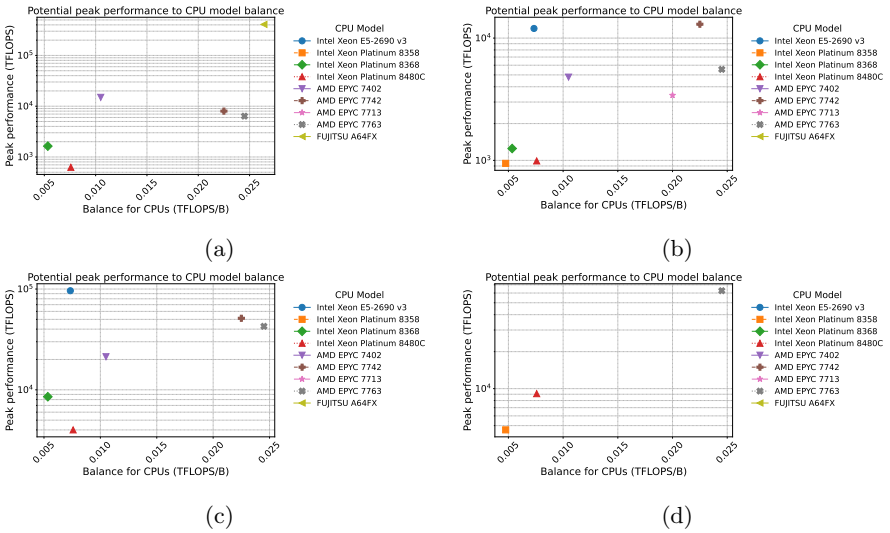


Fig. 17: Dependencies of CPU model potential peak performance to balance metric based on results gained for separate MLPerf HPC Training scenarios Cosmoflow (a), DeepCAM (b), OpenCatalyst (c), OpenFold (d)

When analyzing systems where all CPU-based configurations exhibit Balance 0.025 TFLOPS/B, it can be inferred that the evaluated workloads in benchmarks like MLPerf HPC are likely compute-bound rather than memory-bound. A low Balance value implies that the systems peak computational throughput is small relative to its memory bandwidth. The Y-axis in the discussed plots ranks CPUs by theoretical compute power, with higher values indicating superior performance for compute-bound tasks.

Similarly, Fig. 18 shows diagrams depicting the dependencies of GPU potential performance on the balance, with data separated by the MLPerf HPC Training scenarios. This separation offers a clearer view of how GPU performance relates to memory bandwidth. The results are aggregated across all MLPerf HPC Training versions, from v1.0 to v3.0. By distinguishing between the scenarios, the diagrams provide a more detailed picture of how GPU performance scales with memory bandwidth under different workloads.

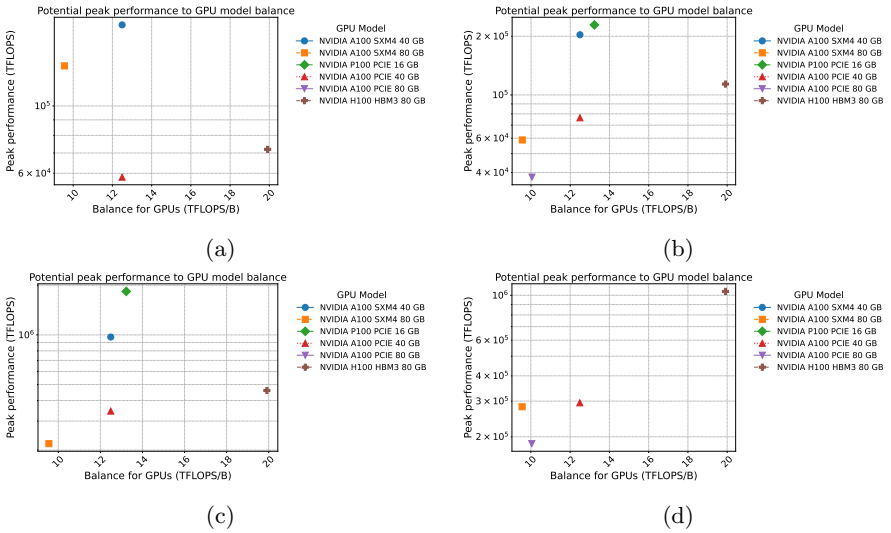


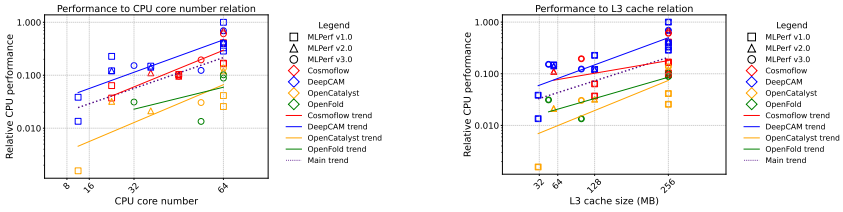
Fig. 18: Dependencies of GPU model potential peak performance to balance metric based on results gained for separate MLPerf HPC Training scenarios Cosmoflow (a), DeepCAM (b), OpenCatalyst (c), OpenFold (d)

A Balance value of 20 TFLOPS/B and lower can be treated as exceptionally low for GPUs, indicating severe memory-bound limitations. When Balance falls to 20 or lower, the memory subsystem becomes a critical bottleneck, leaving compute resources underutilized due to insufficient data supply. On a Peak Performance vs. Balance graph, the most capable GPUs for all scenarios whether compute-heavy training or memory-sensitive inference cluster in the upper-right quadrant. This region represents optimal hardware balance: high computational power (Y) paired with efficient memory bandwidth utilization (X).

The diagrams presented in Fig. 19 illustrate the relationships between CPU characteristics and the relative performance of a single CPU, as calculated according to the methodology outlined in the previous section. The results are aggregated by MLPerf HPC Training scenarios and versions, with a global trend displayed for each individual scenario. Notably, these diagrams highlight a significant increase in performance as the number of CPU cores or the size of the L3 cache grows.

5 Conclusion

In this work, we investigated the efficiency of a highly demanded class of machine learning (ML) tasks on different types of processors and graphics accelerators. As benchmarks, we employed the AI-Benchmark suite, the MLPerf HPC benchmarks (including the training scenarios Cosmoflow, DeepCAM, OpenCatalyst,



(a) (b)

Fig. 19: Dependencies of relative performance impact of a single CPU based on its characteristics, such as core count (a) and L3 cache size (b)

and OpenFold), as well as a variety of tasks from LeelaChessZero, MLPack, Numenta, Whisper, Caffe, and Llama. Separate analyses of processor-based computation and GPU-based computation were conducted to provide a comprehensive performance overview.

From our extensive analysis of a wide range of ML tasks and software packages, we demonstrated that these workloads typically fall into the compute-bound category across both central processing units (CPUs) and graphics processing units (GPUs). In particular, the AI-Benchmark results revealed stark differences in the efficiency of Intel versus AMD processors, also highlighting that processor characteristics have a substantial impact on performance. A Roofline analysis of the AI-Benchmark further confirmed that these ML tasks are compute-bound when run on GPUs; however, it also showed that GPU efficiency still requires significant improvement to fully realize their potential for ML workloads.

In contrast, hybrid benchmarks from MLPerf, which handle large datasets from diverse domains, tend to shift more toward memory-bound behavior. Our analysis also demonstrated very decent scalability with respect to both the number of CPUs and the number of GPUs across all considered MLPerf HPC benchmarks, emphasizing the importance of resource scalability for achieving optimal performance in HPC-based machine learning.

Understanding the interplay between different compute devices and the nature of ML tasks is crucial for selecting the optimal hardware platform to achieve rapid and efficient high-performance computing for ML. The findings on task type, device-specific performance, and overall efficiency provide valuable insights for guiding hardware choices in future HPC-based ML deployments.

6 Acknowledgments

The research was financially supported by the Russian Science Foundation (project No. 20-71-10127), https://rscf.ru/project/20-71-10127/. The part of work performed on the computational resources of HPC facilities at HSE University [17] were carried out by G.Promyslov within the framework of the Basic Research Program at HSE University.

References

1. Machine Learning Test Suite Collection - OpenBenchmarking.org. https://openbenchmarking.org/suite/pts/machine-learning
2. Abadi, M., Barham, P., Chen, J., et al.: TensorFlow: a system for Large-Scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283 (2016)
3. Behler, J.: Representing potential energy surfaces by high-dimensional neural network potentials. J. Phys.: Condens. Matter **26**(18), 183,001 (2014)
4. Benedict, S.: Energy-aware performance analysis methodologies for HPC architectures - An exploratory study. Journal of Network and Computer Applications **35**(6), 1709–1719 (2012)
5. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794 (2016)
6. Dongarra, J., Heroux, M.A., Luszczek, P.: High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. The International Journal of High Performance Computing Applications **30**(1), 3–10 (2016)
7. Dongarra, J.J., Luszczek, P., Petitet, A.: The LINPACK benchmark: past, present and future. Concurrency and Computation: practice and experience **15**(9), 803–820 (2003)
8. Dunn, B.: Optimizing high performance computing systems, resource utilization and throughput by leveraging machine learning. Ph.D. thesis (2021)
9. Farrell, S., Emani, M., Balma, J., et al.: MLPerf HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems. In: 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC), pp. 33–45 (2021). doi:10.1109/MLHPC54614.2021.00009
10. Gigli, L., Goscinski, A., Ceriotti, M., Tribello, G.A.: Modeling the ferroelectric phase transition in barium titanate with DFT accuracy and converged sampling. Phys. Rev. B **110**(2), 024,101 (2024)
11. Goodfellow, I.: Deep learning, vol. 196. MIT press (2016)
12. How, W.B., Chong, S., Grasselli, F., et al.: Adaptive energy reference for machine-learning models of the electronic density of states. Phys. Rev. Mater. **9**(1), 013,802 (2025)
13. Ignatov, A.: Ai-benchmark website. https://ai-benchmark.com
14. Ignatov, A., Timofte, R., Chou, W., et al.: Ai benchmark: Running deep neural networks on android smartphones. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, pp. 0–0 (2018)
15. Jouppi, N.P., Young, C., Patil, N., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture, pp. 1–12 (2017)
16. Kocer, E., Ko, T.W., Behler, J.: Neural network potentials: A concise overview of methods. Ann. Rev. Phys. Chem. **73**(1), 163–186 (2022)
17. Kostenetskiy, P., Chulkevich, R., Kozyrev, V.: HPC resources of the higher school of economics. In: Journal of Physics: Conference Series, vol. 1740, p. 012050. IOP Publishing (2021)
18. Liu, Y., Niu, C., Wang, Z., Gan, Y., Zhu, Y., Sun, S., Shen, T.: Machine learning in materials genome initiative: A review. J. Mater. Sci. Tech. **57**, 113–122 (2020)

19. Mattson, P., Reddi, V.J., Cheng, C., et al.: MLPerf: An industry standard benchmark suite for machine learning performance. IEEE Micro **40**(2), 8–16 (2020)
20. MLCommons: MLPerf HPC Training Benchmarks (2025). URL https://mlcommons.org/benchmarks/training-hpc/. Accessed: 2025-02-04
21. Paszke, A., Gross, S., Massa, F., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
22. Raju, M., Gulhane, K., Gulshan, B.A., et al.: Architectures of high-performance computing systems for machine learning workloads. In: Integrating Machine Learning Into HPC-Based Simulations and Analytics, pp. 435–460. IGI Global Scientific Publishing (2025)
23. Wang, H., Zhang, L., Han, J., Weinan, E.: DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. Comp. Phys. Comm. **228**, 178–184 (2018)
24. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM **52**(4), 65–76 (2009)