

# Resource-Efficient Machine Learning

Oana Balmau<sup>\*1</sup>, Matthias Boehm<sup>\*2</sup>, Ana Klimovic<sup>\*3</sup>,  
Peter Pietzuch<sup>\*4</sup>, and Pinar Tözün<sup>\*5</sup>

1 McGill University, CA. oana.balmau@cs.mcgill.ca

2 TU Berlin, DE & BIFOLD, DE. matthias.boehm@tu-berlin.de

3 ETH Zürich, CH. aklimovic@ethz.ch

4 Imperial College London, GB. prp@imperial.ac.uk

5 IT University of Copenhagen, DK. pito@itu.dk

---

## Abstract

Machine learning (ML) enables forecasts, even in real-time, at ever lower cost and better accuracy. Today, data scientists are able to collect more data, access that data faster, and apply more complex data analysis than ever. As a result, ML impacts a variety of fields such as healthcare, finance, and entertainment.

The advances in ML are mainly thanks to the exponential evolution of hardware, the availability of the large datasets, and the emergence of machine learning frameworks, which hide the complexities of the underlying hardware, boosting the productivity of data scientists. On the other hand, the computational need of the powerful ML models has increased several orders of magnitude in the past decade. A state-of-the-art large language processing model can cost of millions dollars to train in the cloud [52] without accounting for the electricity cost and carbon footprint [17, 55]. This makes the current rate of increase in model parameters, datasets, and compute budget unsustainable. To achieve a more sustainable progress in ML in the future, it is essential to invest in more resource-/energy-/cost-efficient solutions.

In this Dagstuhl Seminar, our main goal was to reason critically about how we build software and hardware for end-to-end machine learning. The crowd was composed of experts from academia and industry across fields of data management, machine learning, compilers, systems, and computer architecture covering expertise of algorithmic optimizations in machine learning, job scheduling and resource management in distributed computing, parallel computing, and data management and processing.

During the seminar, we explored how to improve ML resource efficiency through a holistic view of the ML landscape, which includes data preparation and loading, continual retraining of models in dynamic data environments, compiling ML on specialized hardware accelerators, hardware/software co-design for ML, and serving models for real-time applications with low-latency requirements and constrained resource environments. We hope that the discussions and the work planned during the seminar will lead to increased awareness for understanding the utilization of modern hardware and kickstart future developments to minimize hardware underutilization while still enabling emerging applications powered by ML.

**Seminar** July 28 – August 02, 2024 – <https://www.dagstuhl.de/24311>

**2012 ACM Subject Classification** Hardware; Information systems → Data management systems; Computing methodologies → Artificial intelligence; Computing methodologies → Machine learning

**Keywords and phrases** Machine Learning, Modern Hardware, Sustainability, Energy-Efficiency, Benchmarking, Hardware-Software Co-Design, Data Management, Compilation

**Digital Object Identifier** 10.4230/DagRep.14.7.153

---

\* Editor / Organizer



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Resource-Efficient Machine Learning, *Dagstuhl Reports*, Vol. 14, Issue 7, pp. 153–169

Editors: Oana Balmau, Matthias Boehm, Ana Klimovic, Peter Pietzuch, and Pinar Tözün



DAGSTUHL  
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Executive Summary


*Oana Balmau (McGill University, CA oana.balmau@cs.mcgill.ca)*

*Matthias Boehm (TU Berlin & BIFOLD, DE, matthias.boehm@tu-berlin.de)*

*Ana Klimovic (ETH Zürich, CH, aklimovic@ethz.ch)*

*Peter Pietzuch (Imperial College London, GB, prp@imperial.ac.uk)*

*Pinar Tözün (IT University of Copenhagen, DK, pito@itu.dk)*

**License**  Creative Commons BY 4.0 International license  
 © Oana Balmau, Matthias Boehm, Ana Klimovic, Peter Pietzuch, and Pinar Tözün

While the capabilities of machine learning models have become more and more impressive in the last decade, one cannot overlook the computational footprint of their end-to-end lifecycle. According to the Stanford AI Index Report [52], the computational complexity of the state-of-the-art language models has increased 7 orders of magnitude since 2017. In turn, this increases the estimated costs to train these models in the cloud by 5 orders of magnitude, and the carbon footprint of training such models are equivalent to 10s of human years. Furthermore, the cost of training is only a fraction of the whole costs. After training, then comes the cost of continuously deploying these models, which depend on the way these models are used for inference and the frequency of retraining to update the models.

The participants of the Dagstuhl Seminar on Resource-Efficient Machine Learning (ML) targeted the computational efficiency challenges for machine learning, especially deep learning, from different angles and by focusing on the different stages. On the first day of the seminar, we split into four groups, each with a specific focus. The groups identified the research questions they want to focus on, delved deeper into the existing work, and identified future steps to continue collaborations beyond the seminar.

The first group, **Resource-Efficient Data Selection** (Section 3), targets the efficiency of data selection methods for training deep learning models. Data selection is a preliminary step before any model training, but specifically for fine-tuning tasks, where a pre-trained model must be specialized for a specific task. The effectiveness of a data selection method is typically evaluated by the accuracy it achieves for the given task. The assumption is, as a side effect, if one can achieve a certain accuracy while using less data, this would improve the efficiency of training. This group questions this assumption and asks the following research question: *what are the trade-offs between the computational complexity of a data selection method, its effectiveness in terms of model accuracy, and the end-to-end training efficiency?*

The second group, **The Future of Portable, Extensible, and Composable Machine Learning Systems** (Section 4), aims at making the emerging ML systems support a larger diversity of applications more efficiently. At the core of this support lies a departure from the dominant reliance on dense tensor computations. Targeting a larger diversity in applications also requires looking at a larger variety of hardware devices, beyond large accelerators that are highly optimized for dense matrix computations. Targeting such diversity requires co-design and finding the right abstractions across software, compilers, and hardware. The group’s vision results in several research challenges with the following overarching research question: *how to design holistic and composable software and hardware frameworks for ML?*

The third group, **Hardware-Software Co-Design for Machine Learning** (Section 5), target similar research challenges to the second group, but with a deeper focus on hardware diversity. The group identifies that the conventional way of optimizing machine learning tasks for a certain hardware device is through tight coupling between high-level ML engineering and low-level performance optimizations. Certain high-level optimizations,

with a specific hardware in mind, in turn, hinders portability to different hardware devices, resulting in sub-optimal efficiency and missed opportunities for functionality. Therefore, the key research question here is *how does one create a hardware stack for ML that enables better portability across different devices?*

The fourth group, **Workload-Aware Machine Learning Serving** (Section 6), focus on ML serving. Serving ML models, especially large language models (LLMs), at scale is highly costly and requires substantial hardware resources. To achieve more resource- and performance-efficient model ways of serving models, one needs to adaptively determine which specialized model to serve or cache, or how to optimize a model. This adaptivity is highly dependent on the workload needs that may be dynamic. This group, therefore, aims to answer the following questions: (1) *what is the behavior and needs of the real-world serving workloads and* (2) *how does one build a framework that enables adaptive model serving based on dynamic user and workload needs?*

**2** Table of Contents

Executive Summary  
*Oana Balmau, Matthias Boehm, Ana Klimovic, Peter Pietzuch, and Pinar Tözün* . 154

Working Group 1: Resource-Efficient Data Selection . . . . . 157

Working Group 2: The Future of Portable, Extensible, and Composable  
Machine Learning Systems . . . . . 159

Working Group 3: Hardware-Software Co-Design for Machine Learning . . 161

Working Group 4: Workload-Aware Machine Learning Serving . . . . . 163

Participants . . . . . 169

### 3 Working Group 1: Resource-Efficient Data Selection

Maximilian Böther (ETH Zürich, CH, [mboether@ethz.ch](mailto:mboether@ethz.ch))

Dagmar Kainmüller (Max-Delbrück-Centrum, DE, [dagmar.kainmueller@mdc-berlin.de](mailto:dagmar.kainmueller@mdc-berlin.de))

Theodoros Rekatsinas (Apple, CH, [trekatsinas@apple.com](mailto:trekatsinas@apple.com))

Ties Robroek (IT University of Copenhagen, DK, [titr@itu.dk](mailto:titr@itu.dk))

Stefanie Scherzinger (University of Passau, DE, [stefanie.scherzinger@uni-passau.de](mailto:stefanie.scherzinger@uni-passau.de))

Pinar Tözün (IT University of Copenhagen, DK, [pito@itu.dk](mailto:pito@itu.dk))

License © Creative Commons BY 4.0 International license

© Maximilian Böther, Dagmar Kainmüller, Theodoros Rekatsinas, Ties Robroek, Stefanie Scherzinger, and Pinar Tözün

Today’s foundation models are trained on vast amounts of data from collections such as RedPajama [15], Dolma [49], FineWeb [44], or The Pile [24]. This data is typically collected by scraping the Internet and applying various ad-hoc filters, such as semantic de-duplication and filtering inappropriate content. In order to achieve high accuracy for a given task, these base language models are subsequently fine-tuned on task-specific data. The result is a process that is not resource-efficient: models are trained on huge amounts of data that are irrelevant to the downstream tasks. Therefore, recent works in the large language model community propose using gradients or attribution vectors derived from gradients to select the most relevant training data [19, 56]. Under the assumption that the training set contains a considerable share of data that is irrelevant to the task at hand, the goal is to select relevant data to *minimize* the number of training points while *maximizing* the downstream model accuracy on certain tasks. These techniques reportedly achieve downstream accuracy comparable to full data training using just 1 to 5% of the training budget. While these data selection techniques show promise, we propose to investigate their effectiveness in production settings. Firstly, there is a lack of investigation of the end-to-end computational needs; i.e., what is the overall training cost including the cost of data selection itself. Secondly, comparisons to other, lighter, data selection techniques are missing in terms of both end-to-end performance and accuracy. Thirdly, it is worthwhile to explore and improve the way gradients are stored to reduce the end-to-end overheads of data selection.

In the following paragraphs, we briefly introduce the state-of-the-art data selection works, and based on our discussions at the Dagstuhl Seminar, outline our vision on how to make data selection more resource-efficient.

**DsDm and LESS.** In the context of large language models, two state-of-the-art methods for data selection, DsDM [19] and LESS [56], emerged. Both of these methods utilize training gradients to assess which data points to select, but do so in slightly different ways.

LESS uses LoRA adapters [27] to reduce the number of trainable parameters. It follows this up by performing a forward pass to compute gradients on all input samples. Each gradient is randomly projected to lower dimensionality to compress it and stored in a simple gradient store. This concludes the preparatory step for data selection. Performing the actual data selection starts with computing the gradients of a few task-relevant validation samples. These samples indicate what sort of training data is required. Afterwards, training data is selected based on which gradients are most cosine-similar to the gradients of the task-relevant samples.

DsDM, on the other hand, relies on *datamodels* [29]. A datamodel is a function that, given a target sample and a subset of data, outputs the expected loss on the sample as if we had actually trained a model on that data. In DsDM’s case, TRAK [42] is a linear datamodel

that assumes every piece of data contributes fixed to the expected loss, i.e., independent of the other included examples. TRAK uses projected gradients from the task we want to optimize to construct the vector containing these influences. DSDM, a data selection algorithm, then uses TRAK by choosing the subset of training samples that minimizes the loss of the samples, maximizing the accuracy.

**Data selection analysis.** As mentioned before, the DSDM and LESS papers show impressive results in terms of training budget reduction. However, a performance characterization of these two data selection methods compared to previously proposed techniques is missing. For example, they are not compared to selection functions based on loss and gradient norm-based sampling [30], DLIS [30], uncertainty sampling [14], RHO-LOSS [38], CRAIG [39], and techniques such as GRAD-MATCH [32]. Therefore, we propose to conduct a thorough benchmarking study of data selection methods for large language model training to complement all these works, especially in terms of end-to-end resource-efficiency for models of different scales and complexity.

Data selection literature frequently claims that their proposed algorithms drastically improve training efficiency. These algorithms, however, come with their own computational demands and complexity, which is often under-reported. After all, scoring data samples to determine which to select for training is not a free process. While, e.g., modern production models in deep learning feature billions of parameters, the results reported in the literature are on smaller  $\approx 120$  million parameter models. Therefore, we propose to explore the scalability of all the aforementioned algorithms and whether these methods are actually resource-efficient or are impractical due to their wall-clock time overhead. In particular, we want to thoroughly compare whether these modern methods that rely on training gradients can compare to other literature, not just on accuracy, but also on total compute efficiency.

**Optimizing gradient storage.** DSDM and LESS are both gradient-based selection methods. LESS computes these gradients before selection and stores them in a gradient store, for which they utilize a file-based implementation. The DSDM paper does not elaborate on how the gradients resulting from computing TRAK are stored. We assume that this could become a bottleneck depending on the embedding sizes, especially for larger models.

Additionally, there are many more applications where storing gradients might be useful:

- For *explainability*, we can better understand the selection decisions if we keep the gradients stored, similar to the examples shown in the TRAK paper [42].
- To *maximize accuracy while minimizing inference cost*, we could derive an  $n$ -shot prompt optimization algorithm which runs a similarity search in the embedding space and then picks the minimum number of examples required to maximize task accuracy.
- To *reduce redundant compute*, we could incrementally compute attribution scores for new tasks without computing the gradients for all samples in the training set again.

Hence, from a systems perspective, we propose to explore if we can improve the state-of-the-art of storing gradients in the context of data selection. A good starting point is the very recent METASTORE [59], which proposes a compression mechanism to make the storage more efficient. An alternative option is utilizing modern hardware capabilities, such as the fast interconnects on the new NVIDIA GH200 machines, to accelerate the movement of gradients to e.g. save them in the gradient store.

**Summary.** Data selection can be valuable for resource-efficient machine learning by drastically reducing the computation required to train task-specific large language models. However, the actual wall-clock runtime and scalability of state-of-the-art methods have yet not

been properly explored. We propose investigating these methods through two proposals. With a holistic benchmarking study, we hope to step forward towards actual resource efficiency and an understanding of data selection functions. With our proposed gradient storage, we want to improve the efficiency of the state-of-the-art data selection methods. We hope that these results will contribute to the transparency of data selection techniques w.r.t. both accuracy and resource efficiency, as well as improve the scalability of techniques reliant on gradient storage.

## 4 Working Group 2: The Future of Portable, Extensible, and Composable Machine Learning Systems

*Patrick Damme (TU Berlin, DE, patrick.damme@tu-berlin.de)*

*Jens Hagemeyer (Bielefeld University, DE, jhagemey@cit-ec.uni-bielefeld.de)*

*Fredrik Kjolstad (Stanford University, USA, kjolstad@stanford.edu)*

*Tom St. John (Meta, USA, tomstjohn617@gmail.com)*

*Cliff Young (Google DeepMind, USA, cliffy@google.com)*

**License** © Creative Commons BY 4.0 International license

© Patrick Damme, Jens Hagemeyer, Fredrik Kjolstad, Tom St. John, and Cliff Young

Applications driven by machine learning (ML) are profoundly changing the world. These applications are both data-access-intensive and compute-intensive. Thus, the efficient execution of these applications on modern hardware is essential to their success. For the past decade, ML systems have been highly optimized to support dense tensor operations. However, we begin to see the limits of this approach when advancing the state of machine learning, e.g., through ever-growing model sizes. Continuing this trend will lead to an increasing divergence between system architectures and new machine learning paradigms. Even today, exponential costs are required for each increment of quality improvement, and sustainability and environmental aspects are also of concern. To support the emergence of diverse machine learning workloads, we require a new system paradigm including new specifications throughout the system stack ranging from language abstractions and software over compilers down to the computer architecture and underlying hardware.

In that context, we propose and discuss a vision of future ML systems, perhaps in 5–10 years from today. We expect such systems will go beyond the current focus on dense, tensor arithmetic and incorporate aspects of wider computer science and high-performance computing calculations. As examples of the kinds of richness the future might hold, we draw inspiration from what we call the *Four Languages taxonomy* that spans four different algebras:

- *Tensor algebras* work on tensors (linear relationships between sets of objects) expressed as an array of values as supported by modern ML frameworks [8, 43] and numerical systems [11, 53].
- *Relational algebras* work on relations between objects (a subset of the Cartesian combination of one or more sets) as supported by relational database management systems [18, 28, 40, 45, 50].
- *Graph algebras* work on edge relationships between sets of objects (such as knowledge graphs, social network graphs, and meshes) as supported by graph engines [3, 36].
- *Spatial algebras* work on metric spaces that express distance relationships between sets of objects as supported by graphics rendering engines [4], molecular dynamics packages [1, 2] and ML-based recommender models.

We base our vision on **three suppositions** regarding the future of ML systems. We believe these suppositions to be likely, given past trends in the ML computing space:

1. **ML applications will be exciting and diverse.** ML models will not be limited to dense tensor computations composed in different ways. Instead, we expect them to evolve in unexpected ways, to incorporate components from the four algebras above. Perhaps, this involves increased reliance on hierarchies, irregular and sparse models, spatial embeddings, and computation with irregular data models such as relational models and graphs.
2. **Hardware will be exciting and diverse.** Hardware will be unexpected in various ways, and not limited to CPUs, programmable GPUs, and dense vector or matrix units. As ML models increase in complexity and embrace components from different algebras, we will see hardware to cater to those algebras. In addition, we will increasingly find ways to build adaptation and reconfigurability into hardware substrates, perhaps through programmable networks, configurable collection-oriented operations, and through support for different data-types and precision.
3. **Computation will be distributed across centers and edges.** We also expect that this heterogeneity goes far beyond the data center focus of current large language model (LLM) training and inference systems to incorporate a wide range of edge, mobile, internet of things (IoT), and other deployment scenarios. Each of these areas will have different economies and trade-offs among cost, compute and memory performance, power (both dissipated and battery capacity), and latency/bandwidth/reliability of communication links.

With these many dimensions of heterogeneity, we imagine a complex, potentially fully connected, graph of interfaces between components with various diverse strengths and capabilities, even potentially changing at runtime by adaptation and reconfigurability to deal with changing requirements and environments. Components (which combine hardware accelerators, software stacks, runtimes, and application software) may have to talk to other components that speak a different language, which makes each interface an opportunity for co-design optimization across the languages.

Each of the languages can express many of the computations of the others. However, each language is conceptually very different, lends itself to a different style of thoughts, and comes with some degree of induced awkwardness or structure in expressing particular computations. And similarly for the reductions or translations from one language to another (as is typical of theoretical reductions). But the structure of these equivalences and reductions induces interesting trade-offs and opportunities for cross-domain optimization and expressiveness. Such fused systems might have huge opportunities for whole-system efficiency. But to realize those opportunities, we need to build first the individual components, then the interfaces between them, and lastly the entire system in a way that allows us to move work fluidly across and into these future ML computing systems.

We do not yet have a strong enough signal to recommend embarking on a *four languages* commercial systems project. Due to the involved complexity this could be a too aggressive jump towards a single grand unified theory of four very different styles of computation. However, we see a range of **research challenges** to approach this vision:

1. How to design a holistic and composable framework at the software and hardware level?
2. How to design user-facing language abstractions and a system-internal unified intermediate representations for combining the different data models and algebras?
3. How to do transformations across the different data representations (i.e., tensor, relational, graph, spatial) enabling portability across the different hardware substrates?



4. How to design the interfaces between these components and how to ensure efficient data exchange?
5. How to assign components to different heterogeneous compute architectures such as CPUs, GPU, TPUs, or even more specialized substrates, and use heterogeneous communication links between them?
6. How to compose domain-specific hardware blocks and at what degree of programmability?
7. How to make hardware reconfigurable and adaptable at run-time, and exploit such adaptability throughout the framework (i.e., language, compiler, runtime)?
8. How to allow experts to fine-tune the system by extending it with hand-written code where automatic approaches are still missing or insufficient?

To conclude, we think there is value in the vision of uniting the four languages and the rich, effective, and expressive future for ML that such a unification might support. Even if we only partially unify some of the areas (as physics did with the electromagnetic and weak forces), it could still be valuable to ML in both theory and practice.

## 5 Working Group 3: Hardware-Software Co-Design for Machine Learning

*Marco Canini (KAUST - Thuwal, SA, marco@kaust.edu.sa)*

*Jeronimo Castrillon (TU Dresden, DE, jeronimo.castrillon@tu-dresden.de)*

*Steven Hand (Google, USA sthand@google.com)*

*Peter R. Pietzuch (Imperial College London, GB, prp@imperial.ac.uk)*

*Foteini Strati (ETH Zurich, CH, foteini.strati@inf.ethz.ch)*

*Shinya Takamaeda-Yamazaki (The University of Tokyo, JP, shinya@is.s.u-tokyo.ac.jp)*

*Lluís Vilanova (Imperial College London, GB vilanova@imperial.ac.uk)*

*Eiko Yoneki (University of Cambridge, GB, eiko.yoneki@cl.cam.ac.uk)*

**License** © Creative Commons BY 4.0 International license

© Marco Canini, Jeronimo Castrillon, Steven Hand, Peter R. Pietzuch, Foteini Strati, Shinya Takamaeda-Yamazaki, Lluís Vilanova, and Eiko Yoneki

The current ML stack is structured across multiple levels. At the top, an ML engineer writes code in a high-level framework such as PyTorch [5] or Tensorflow [6], defining an abstract dataflow graph. To achieve high performance, the high-level graph undergoes a compilation process, involving transformations at multiple levels (in the form of Intermediate Languages [7]), that target a *specific* hardware setup (i.e.  $N$  machines of type  $X$ ). High-level optimizations such as operator fusion are applied first, followed by low-level, hardware-dependent operations such as buffer materialization and padding [60, 13, 46]. As a result, the compiler outputs operators and libraries that target only the specific hardware setup.

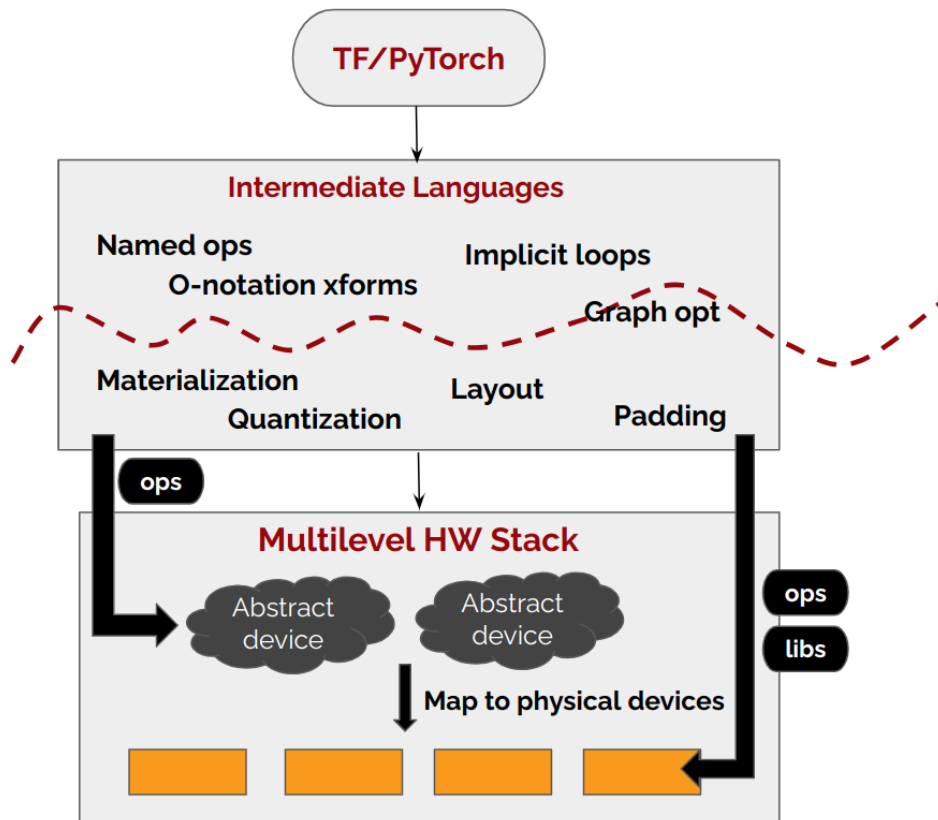
Although the current compilation process leads to high performance, we identify some key issues that make it incompatible with the current and future landscape of heterogeneous ML models and hardware. First, we observe an excessive coupling between high-level ML engineering and low-level performance optimization, hindering portability, both for functionality and performance. ML engineers often perform premature optimization at the model definition stage to maximize efficiency, without having full knowledge of the hardware and its attributes (for example deciding on a specific memory layout or operator fusing). However,

fully decoupling high-level model definitions from low-level implementation and optimizations is impractical since the compiler needs some guidance to navigate the large search space [13] and discover efficient plans in a reasonable time. Additionally, as ML is moving at such a fast pace, the current ML stack fails to keep up with the recent advances in a principled way, compromising correctness and soundness. Finally, the tight coupling between ML model definition and hardware setup, and the focus on real-world performance, makes us unable to reason about performance at different scales and limits access for academic researchers.

We propose a principled redesign of the ML software stack and identify key research questions that need to be addressed at each level. The proposed ML stack is illustrated in Figure 1. At the top level, ML engineers focus on abstract model design (i.e. mathematical formulation) without considering hardware specifications. A critical first step here is defining the input at this high level (i.e. is it the whole ML model graph, or several such that interact via some programming constructs). We also argue that there should be restrictions on which high-level operations are supported. Next, as in the current ML stack, multiple transformations are performed. However, instead of targeting a specific hardware setup, we propose targeting an *abstract multi-level SPMD execution model*, representing a virtual, multi-level Hardware stack. This would allow the compiler to generate “executables” at different levels of abstraction and understand tradeoffs and optimizations at different levels. It would also enable modeling of existing and future hardware in an abstract way, enabling portability in terms of functionality and performance, accelerating the design of new hardware and software in a principled manner, and allowing correctness guarantees at different levels.

For this Hardware stack to be effective, we need to identify what types of information we need to pass to the compiler at different levels. We propose three key levels. At the first level, we define distinct classes of abstract heterogeneous devices, each with a specific memory hierarchy and access granularity, number of compute units, supported functions, and compute-to-memory ratio. At a second level, we view each possible system setup as a collection of these heterogeneous abstract devices, that can communicate with a specific set of primitives. An example of a step in this direction are the abstractions for operations and communication primitives in the CINM and C4CAM [31, 20] multi-level compiler frameworks for emerging near and in-memory accelerators. Finally, the last level handles runtime optimizations, mapping abstract (virtual) to physical devices. At this stage, the compiler decides on specific memory layouts and operator mapping.

In conclusion, we propose reimagining the ML stack in a principled way with explicit consideration of semantics and correctness guarantees across multiple abstract hardware layers. To implement this, it is essential to clearly define interfaces and intermediate representations required at each level of the ML stack, and evaluate the tradeoffs between performance and adaptability in the fast-moving ML space.



■ **Figure 1** The proposed ML software-hardware stack, consisting of various levels of abstractions.

## 6 Working Group 4: Workload-Aware Machine Learning Serving

Matthias Boehm (TU Berlin & BIFOLD, DE, [matthias.boehm@tu-berlin.de](mailto:matthias.boehm@tu-berlin.de))

Khuzaima Daudjee (University of Waterloo, CA, [khuzaima.daudjee@uwaterloo.ca](mailto:khuzaima.daudjee@uwaterloo.ca))

Pamela Delgado (HES-SO, CH, [pamela.delgado@hevs.ch](mailto:pamela.delgado@hevs.ch))

Thaleia Dimitra Doudali (IMDEA Software Institute – Madrid, ES, [thaleia.doudali@imdea.org](mailto:thaleia.doudali@imdea.org))

Ana Klimovic (ETH Zurich, CH, [aklimovic@ethz.ch](mailto:aklimovic@ethz.ch))

James Kwok (HKUST, HK, [jamesk@cse.ust.hk](mailto:jamesk@cse.ust.hk))

Manisha Luthra (TU Darmstadt & DFKI, DE, [manisha.luthra@dfki.de](mailto:manisha.luthra@dfki.de))

Tilmann Rabl (HPI & University of Potsdam, DE, [tilmann.rabl@hpi.de](mailto:tilmann.rabl@hpi.de))

Ehsan Yousefzadeh-Asl-Miandoab (IT University of Copenhagen, DK, [ehyo@itu.dk](mailto:ehyo@itu.dk))

License © Creative Commons BY 4.0 International license

© Matthias Boehm, Khuzaima Daudjee, Pamela Delgado, Thaleia Dimitra Doudali, Ana Klimovic, James Kwok, Manisha Luthra, Tilmann Rabl, and Ehsan Yousefzadeh-Asl-Miandoab

Serving machine learning (ML) models at scale, particularly modern large language models (LLMs), is extremely resource-intensive and costly. We observe two major trends that increase resource requirements of ML serving workloads. First, there is increasing scale in terms of number of client requests and model sizes. Increasing context-lengths in conversational LLM interactions also generates large state per client. Second, we see increasing

model personalization for different domains and user groups (through fine tuning and prompt engineering), model specialization for different tasks (through custom models), as well as multi-modal and multi-component models. There are, however, new opportunities to mitigate these challenges. Weight pruning and other forms of sparsity exploitation are becoming more practical. Combined with new hardware features for reconfiguration and tuning, these increase the optimization potential for ML serving.

**Limitations of Existing Work.** Many ML serving systems have been developed to optimize latency and throughput, such as TensorFlow Serving [41], TorchServe, Clockwork [26], vLLM [33], ServerlessLLM [22], Clipper [16], Pretzel [34], and Rafiki [54]. Optimizations for different throughput-latency-accuracy trade-offs include data/request batching [9, 58], model pruning and quantization [21, 35], result caching [51, 23], multi-model optimizations [47], and specialized or fine-tuned model serving [37, 48, 12, 57]. Despite promising performance impact, these strategies are mostly applied in a heuristic manner, in isolation, and/or in a static offline step before deployment. Recent work explores dynamically tuning key-value cache compression [25], context pruning [10], and sparsity [61]. We see an opportunity to build on prior work to design an algorithm and system to *adaptively* tune these optimization knobs *holistically* on a *per-request* basis.

**Vision.** These opportunities motivate a *new paradigm* for serving ML models: adaptive and workload-aware ML serving. Inspired by two decades of research on adaptive query processing, we make a case for adaptive, workload-aware optimization and reconfiguration of ML serving pipelines. To this end, we envision deploying multiple variants of sparsified and specialized models, periodically profiling samples of client requests (with several variants) for accuracy and resource consumption, and dynamically reconfiguring serving configurations (variant selection, placement, and parameter tuning) to maximize throughput subject to accuracy, energy, and latency constraints. Working towards this vision, we aim to (1) share a workload characterization of real serving traces, the potential of sparsification, and opportunities of adaptation; (2) devise a holistic reconfiguration framework including objectives, tuning knobs, and optimization algorithms; and (3) conduct preliminary experiments on two different prototypes for LLMs and traditional ML models.

## References

- 1 GROMACS. <https://www.gromacs.org/>, 2024.
- 2 NAMD. <https://www.ks.uiuc.edu/Research/namd/>, 2024.
- 3 Neo4j. <https://neo4j.com/>, 2024.
- 4 OpenGL. <https://www.opengl.org/>, 2024.
- 5 Pytorch. <https://pytorch.org/>, 2024.
- 6 Tensorflow. <https://www.tensorflow.org/>, 2024.
- 7 The torch-mlir project. <https://github.com/llvm/torch-mlir>, 2024.
- 8 Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283. USENIX Association, 2016.
- 9 Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134, Santa Clara, CA, July 2024. USENIX Association.

- 10 Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA, 2024. Curran Associates Inc.
- 11 Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017.
- 12 Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant lora serving. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 1–13, 2024.
- 13 Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated End-to-End optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, Carlsbad, CA, October 2018. USENIX Association.
- 14 Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.
- 15 Together Computer. Redpajama: an open dataset for training large language models, 2023.
- 16 Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *NSDI*, pages 613–627, 2017.
- 17 Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. Measuring the Carbon Intensity of AI in Cloud Instances. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22*, page 1877–1894, New York, NY, USA, 2022. Association for Computing Machinery.
- 18 Markus Dreseler, Jan Kossmann, Martin Boissier, Stefan Klauck, Matthias Uflacker, and Hasso Plattner. Hyrise re-engineered: An extensible database system for research in relational in-memory data management. In Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irimi Fundulaki, Carsten Binnig, and Zoi Kaoudi, editors, *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, pages 313–324. OpenProceedings.org, 2019.
- 19 Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- 20 Hamid Farzaneh, João Paulo Cardoso de Lima, Mengyuan Li, Asif Ali Khan, Xiaobo Sharon Hu, and Jeronimo Castrillon. C4cam: A compiler for cam-based in-memory accelerators. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'24), Volume 3, ASPLOS '24*, pages 164–177, New York, NY, USA, May 2024. Association for Computing Machinery.
- 21 Elias Frantar and Dan Alistarh. Sparsegpt: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning, ICML'23*. JMLR.org, 2023.
- 22 Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. ServerlessLLM: Low-Latency serverless inference for large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 135–153, Santa Clara, CA, July 2024. USENIX Association.
- 23 Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient large language model serving

- for multi-turn conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126, Santa Clara, CA, July 2024. USENIX Association.
- 24 Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint*, 2020.
  - 25 Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms, 2024.
  - 26 Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving DNNs like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.
  - 27 Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
  - 28 Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.
  - 29 Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.
  - 30 Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.
  - 31 Asif Ali Khan, Hamid Farzaneh, Karl F. A. Friebel, Lorenzo Chelini, and Jeronimo Castrillon. Cinnm (cinnamon): A compilation infrastructure for heterogeneous compute in-memory and compute near-memory paradigms. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’25)*, ASPLOS ’25. Association for Computing Machinery, March 2025.
  - 32 KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh K. Iyer. GRAD-MATCH: gradient matching based data subset selection for efficient deep model training. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, 2021.
  - 33 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP ’23*, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery.
  - 34 Yunseong Lee, Alberto Scolari, Byung-Gon Chun, Marco Domenico Santambrogio, Markus Weimer, and Matteo Interlandi. PRETZEL: opening the black box of machine learning prediction serving systems. In *OSDI*, pages 611–626, 2018.
  - 35 Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100, 2024.
  - 36 Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD Conference*, pages 135–146. ACM, 2010.

- 37 Daniel Mendoza, Francisco Romero, and Caroline Trippel. Model selection for latency-critical inference serving. In *Proceedings of the Nineteenth European Conference on Computer Systems*, EuroSys '24, page 1016–1038, New York, NY, USA, 2024. Association for Computing Machinery.
- 38 Sören Mindermann, Jan Markus Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.
- 39 Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.
- 40 Thomas Neumann and Michael J. Freitag. Umbra: A disk-based system with in-memory performance. In *CIDR*. [www.cidrdb.org](http://www.cidrdb.org), 2020.
- 41 Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. Tensorflow-serving: Flexible, high-performance ML serving. In *ML-Systems@NeurIPS Workshop*, 2017.
- 42 Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: attributing model behavior at scale. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.
- 43 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.
- 44 Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint*, 2024.
- 45 Mark Raasveldt and Hannes Mühleisen. Duckdb: an embeddable analytical database. In *SIGMOD Conference*, pages 1981–1984. ACM, 2019.
- 46 Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *SIGPLAN Not.*, 48(6):519–530, June 2013.
- 47 Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. INFaaS: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411. USENIX Association, 2021.
- 48 Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph Gonzalez, and Ion Stoica. Slora: Scalable serving of thousands of lora adapters. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 296–311, 2024.
- 49 Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxin Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh

- Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024.
- 50 Michael Stonebraker and Lawrence A. Rowe. The design of postgres. In *SIGMOD Conference*, pages 340–355. ACM Press, 1986.
- 51 Foteini Strati, Sara Mcallister, Amar Phanishayee, Jakub Tarnawski, and Ana Klimovic. Déjàvu: KV-cache streaming for fast, fault-tolerant generative LLM serving. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 46745–46771. PMLR, 21–27 Jul 2024.
- 52 The AI Index Report. The AI Index Report - Measuring trends in AI. <https://aiindex.stanford.edu/report/>, 2024.
- 53 Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, 13(2):22–30, 2011.
- 54 Wei Wang, Jinyang Gao, Meihui Zhang, Sheng Wang, Gang Chen, Teck Khim Ng, Beng Chin Ooi, Jie Shao, and Moaz Reyad. Rafiki: Machine learning as an analytics service system. *Proc. VLDB Endow.*, 12(2):128–140, 2018.
- 55 Carole-Jean Wu, Bilge Acun, Ramya Raghavendra, and Kim Hazelwood. Beyond efficiency: Scaling ai sustainably. *IEEE Micro*, pages 1–8, 2024.
- 56 Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: selecting influential data for targeted instruction tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- 57 Xiaozhe Yao and Ana Klimovic. Deltazip: Multi-tenant language model serving via delta compression, 2023.
- 58 Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.
- 59 Huayi Zhang, Binwei Yan, Lei Cao, Samuel Madden, and Elke Rundensteiner. Metastore: Analyzing deep learning meta-data at scale. *Proceedings of the VLDB Endowment*, 17(6), 2024.
- 60 Zhen Zheng, Zaifeng Pan, Dalin Wang, Kai Zhu, Wenyi Zhao, Tianyou Guo, Xiafei Qiu, Minmin Sun, Junjie Bai, Feng Zhang, Xiaoyong Du, Jidong Zhai, and Wei Lin. Bladedisc: Optimizing dynamic shape machine learning workloads via compiler approach. *Proc. ACM Manag. Data*, 1(3), November 2023.
- 61 Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, Dahua Lin, and Chao Yang. Sampleattention: Near-lossless acceleration of long context LLM inference with adaptive structured sparse attention. *CoRR*, abs/2406.15486, 2024.



## Participants

- Matthias Böhm  
TU Berlin, DE
- Maximilian Böther  
ETH Zürich, CH
- Marco Canini  
KAUST – Thuwal, SA
- Jerónimo Castrillón-Mazo  
TU Dresden, DE
- Patrick Damme  
TU Berlin, DE
- Khuzaima Daudjee  
University of Waterloo, CA
- Pamela Delgado  
HES-SO Valais-Wallis –  
Siders, CH
- Thaleia Dimitra Doudali  
IMDEA Software Institute –  
Madrid, ES
- Jens Hagemeyer  
Universität Bielefeld, DE
- Steven Hand  
Google – Mountain View, US
- Dagmar Kainmüller  
Max-Delbrück-Centrum –  
Berlin, DE
- Fredrik Kjolstad  
Stanford University, US
- Ana Klimovic  
ETH Zürich, CH
- James Kwok  
HKUST – Hong Kong, HK
- Manisha Luthra Agnihotri  
TU Darmstadt, DE & DFKI –  
Darmstadt, DE
- Peter R. Pietzuch  
Imperial College London, GB
- Tilmann Rabl  
Hasso-Plattner-Institut,  
Universität Potsdam, DE
- Theo Rekatsinas  
Apple – Zürich, CH
- Ties Robroek  
IT University of  
Copenhagen, DK
- Stefanie Scherzinger  
Universität Passau, DE
- Tom St. John  
Meta – Menlo Park, US
- Foteini Strati  
ETH Zürich, CH
- Shinya Takamaeda-Yamazaki  
The University of Tokyo, JP
- Pinar Tözün  
IT University of  
Copenhagen, DK
- Lluís Vilanova  
Imperial College London, GB
- Eiko Yoneki  
University of Cambridge, GB
- Cliff Young  
Google – Mountain View, US
- Ehsan  
Yousefzadeh-Asl-Miandoab  
IT University of  
Copenhagen, DK

