Optimization framework for large deep neural networks (en) - Plateforme logicielle d'optimization pour réseaux de neurones profonds de très grande taille (fr)

- **Doctoral Domain**: Computer Science
- **Funding**: CIFRE schema, hired in French CDD (36 months) by Huawei France
- **Work Places**: Huawei Paris Research Center (Boulogne-Billancourt) and Maison de la Simulation (University Paris-Saclay)
- **Industrial Supervision**: Dr. Chong Li and Prof. Denis Barthou, Huawei Paris Distributed and Parallel Technologies Lab
- **Academic Supervision**: Prof. Nahid Emad, Université Versailles Saint-Quentin-en-Yvelines
- **Keywords**: Massively Parallelism, Linear Algebra, Performance Optimisation, Deep Learning, Large Models

1 Context

Very large models are now widespread and require large amount of computing power and memory capacity. In these models, large matrix multiplications are at the heart of each layer. Two factors are limiting inference and training for large models: (i) High memory usage and large memory footprint, requiring possible compression of memory models, memory layout optimization and scheduling for data movement optimization; (ii) Adaptation to the heterogeneous parallel architecture, taking into account accelerators and requiring performance modeling and load balancing

For performance modeling and load balancing, several techniques have been proposed, taking into account the time to move data across the nodes, either statically (see [WTL+22, WLT+21] for instance) or at runtime (see [HBC18] for instance). For memory optimization in the context of AI, many techniques tackle the problem of reducing the size of the matrices:

- Pruning: Unimportant model weights or connections are trimmed while preserving the model capacity. Some recent hardware support some structure pruning, where some elements are set to zero following some pattern restrictions, such as the Nvidia A100 GPU or the Google TPUv4[JKL+23]. Several pruning methods have been proposed [GEH19, LWK18, ZG17], and some pruning method consider retraining after each pruning iteration (or pruning while training). Many of the pruning techniques are mostly for inference in the industrial level.

- Sparsification: This is a more general approach compared to pruning, and can be applied to any dense operator. Sparse attention matrices such as in the Big Bird model [ZGD$^+$21] can have different forms of sparse patterns. For models based on Mixtures of Experts, the pyramidal reformulation of MoEs proposed by [RLY$^+$22] is another example of sparsification. The key for sparsification is to keep only model weights that are relevant, and it boils down to analysing the eigenvalues of the large matrices.
- Distillation: This is essentially the reformulation of the initial pretrained model into a simpler model (a student), through another more limited training. Distillation targets inference and can resort to any previous compression technique, plus some dedicated compression methods.

In addition to the reduction and sparsification of the matrices, the compression format used to represent sparse matrices has also an impact on performance, depending on the capacities of the architecture [MHDE18]. Finally, while many techniques for model compression have been proposed (see [TDBM22] for transformers for instance), the interactions between them is not well studied. This corresponds to a similar situation for compilers where many optimizations were proposed, each promising at least a 10% improvement but the gain obtained after combining optimization never correspond to the addition of the individual gains. Therefore the question "how to go beyond individual optimizations that only work for particular cases?". To find the most appropriate sequence of optimizations, this led to languages where the user defined its sequence of optimization (metaprogramming such as Halide, TVM for instance), to the polyhedral framework unifying many transformations, to cost analysis (bridging models like [LH11]), to auto-tuning and to AI-guided optimizations. As far as parallelism and memory optimization are concerned, this has not been studied in the context of model compression.

2 Objectives of the PhD

The objectives of the PhD are to propose a framework to capture many different memory optimization techniques and a methodology to guide the optimization process on large models. The main focus will be on transformer-based models, corresponding to large models for generative AI.

This research project can be decomposed into 3 steps:

- Propose a high-level formulation of the targeted models in order to allow a class of optimizations. The optimizations that will be considered are relative to sparsification, memory optimizations, including some algebraic rewriting of the model (for instance fusion). This would correspond to the main contribution of the PhD. As a first idea, it may be possible to consider matrices of operators (or lambda functions) where in the matrix product, the multiplication would be replaced by the application of a function. This would allow the representation of softmax, relu operators as matrix ones. Then techniques based on sparsification could be explored more systematically (following [PLE22] ideas). The possible composition of different optimizations expressed through this formulation is a key objective. A combination of the optimizations according to the unite and conquer approach [EP16] can bring a solution to this objective.
- The automatic exploration of some of these compositions, taking into account the specificities of the optimization in terms of memory and hardware acceleration (ratio of vector/matrix operators, constraints on memory layout, ...). The hardware is in general dedicated to run

particular models in mind [JKL+23, LTXZ19], reformulating the model by compressing it is also a way to find a model that better fit the architecture. Semantic-based performance and memory prediction would enhance solutions be portable and systematical.

- Evaluation of large models such as BERT, GPT, LLaMA or Pangu with two metrics in mind: performance and memory consumption. Several models will be evaluated, with the transformations proposed, along the PhD. A performance model will be first designed to evaluate the amount of computation and memory gains after optimization. The generation of an optimized low-level code for different architectures is not part of this PhD and will reuse existing works. The effective implementation of targeted large models on high-performance architectures is a delicate task requiring a mastery of these architectures and the associated multi-level programming models. However, this efficient exploitation can be facilitated by programming and execution environments for parallel and/or large-scale distributed systems such as the YML software platform (yml.prism.uvsq.fr) [Del08]. YML is a solution for high-level heterogeneous programming implementations, integrating asynchronous communications, in this thesis.

3 Research Work plan

The program below can be rolled out according to the following schedule:

- 1st year:
  - Bibliography on parallel programming and performance analysis, numerical compression methods, large neural network models, new parallel and/or distributed architectures, hardware accelerators and parallel architectures dedicated to artificial intelligence.
  - Study and evaluation of algorithms of parallel computing and of eigenvalue analysis.
  - Prepare very large datasets and experiments on clusters of accelerators.
  - Identify and analyze limitations of the state of the art.
- 2nd year:
  - Create high-level models for performance analysis.
  - Design and develop methods and algorithms of performance and memory footprint optimization.
  - Experiment and evaluate the proposals on clusters of accelerators or on cloud.
  - Publication of the results obtained.
- 3rd year
  - Propose and develop parallel evolution on the second year's results.
  - Validate the methodological and computational results by experimentation on industrial case studies.
  - Publication of the results obtained.
  - Defense of the thesis.

4 How to apply

Please send your candidature (CV, cover letter, school transcripts, recommendation letters, and possibly a school project or internship report) to Prof. Nahid Emad (nahid.emad@uvsq.fr), Dr. Chong Li (ch.l@huawei.com) and Prof. Denis Barthou (denis.barthou@huawei.com).

References

[Del08] O Delannoy. *YML: a scientific workflow for high performance computing.* PhD thesis, Ph. D. Thesis, 2008.

[EP16] Nahid Emad and Serge Petiton. Unite and conquer approach for high scale numerical computing. *Journal of computational science*, 14:5–14, 2016.

[GEH19] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019.

[HBC18] Pierre Huchant, Denis Barthou, and Marie Christine Counilh. Adaptive partitioning for iterated sequences of irregular opencl kernels. In *30th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2018, Lyon, France, September 24-27, 2018*, pages 262–265. IEEE, 2018.

[JKL$^+$23] Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings, 2023.

[LH11] Chong Li and Gaétan Hains. A simple bridging model for high-performance computing. In *2011 International Conference on High Performance Computing Simulation*, pages 249–256, 2011.

[LTXZ19] Heng Liao, Jiajin Tu, Jing Xia, and Xiping Zhou. DaVinci: A Scalable Architecture for Neural Network Computing. In *2019 IEEE Hot Chips 31 Symposium (HCS)*, pages 1–44, 2019.

[LWK18] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization, 2018.

[MHDE18] Ichrak Mehrez, Olfa Hamdi-Larbi, Thomas Dufaud, and Nahid Emad. Machine Learning for Optimal Compression Format Prediction on Multiprocessor Platform. In *2018 International Conference on High Performance Computing & Simulation, HPCS 2018, Orleans, France, July 16-20, 2018*, pages 213–220. IEEE, 2018.

[PLE22] Quentin R. Petit, Chong Li, and Nahid Emad. Distributed and parallel sparse computing for very large graph neural networks. In *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*, pages 6796–6798. IEEE, 2022.

[RLY$^+$22] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale, 2022.

[TDBM22] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2022.

[WLT$^+$21] Haoran Wang, Chong Li, Thibaut Tachon, Hongxing Wang, Sheng Yang, Sébastien Limet, and Sophie Robert. Efficient and systematic partitioning of large and deep neural networks for parallelization. In *Euro-Par 2021: Parallel Processing*, pages 201–216. Springer International Publishing, 2021.

[WTL$^+$22] Haoran Wang, Thibaut Tachon, Chong Li, Sophie Robert, and Sébastien Limet. SMSG: profiling-free parallelism modeling for distributed training of DNN. *Int. J. Parallel Program.*, 51(2–3):109–127, dec 2022.

[ZG17] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017.

[ZGD$^+$21] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.