POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Graph Neural Network Acceleration with SODA Framework

Master of Science Thesis in
Computer Science and Engineering

Author: **Giovanni Demasi**

Student ID: 987062
Advisor: Prof. Fabrizio Ferrandi
Co-advisors: Serena Curzel
Academic Year: 2022-23

Abstract

Here goes the Abstract in English of your thesis followed by a list of keywords. The Abstract is a concise summary of the content of the thesis (single page of text) and a guide to the most important contributions included in your thesis. The Abstract is the very last thing you write. It should be a self-contained text and should be clear to someone who hasn't (yet) read the whole manuscript. The Abstract should contain the answers to the main scientific questions that have been addressed in your thesis. It needs to summarize the adopted motivations and the adopted methodological approach as well as the findings of your work and their relevance and impact. The Abstract is the part appearing in the record of your thesis inside POLITesi, the Digital Archive of PhD and Master Theses (Laurea Magistrale) of Politecnico di Milano. The Abstract will be followed by a list of four to six keywords. Keywords are a tool to help indexers and search engines to find relevant documents. To be relevant and effective, keywords must be chosen carefully. They should represent the content of your work and be specific to your field or sub-field. Keywords may be a single word or two to four words.

Keywords: here, the keywords, of your thesis

Abstract in lingua italiana

Qui va l'Abstract in lingua italiana della tesi seguito dalla lista di parole chiave.

Parole chiave: qui, vanno, le parole chiave, della tesi

Contents

1 Introduction

In recent years, deep learning has brought about a revolutionary transformation in various machine learning tasks, spanning from image classification and video processing to speech recognition and natural language understanding. Traditionally, these tasks have predominantly operated within the Euclidean space, where data is typically represented. Nevertheless, a growing number of applications now generate data from non-Euclidean domains, presenting it in the form of complex graphs with intricate relationships and interdependencies among objects. The inherent complexity of graph data has posed considerable challenges for existing machine learning algorithms. Consequently, there has been a surge of studies focusing on extending deep learning techniques to accommodate and leverage graph data.

Graph neural networks (GNNs) have been introduced in response to the growing demand for learning tasks involving graph data, which encompasses extensive relational information among its elements. These neural models effectively capture the interdependence among graph nodes by employing message passing mechanisms.

As Graph Neural Networks are increasingly employed, particularly in domains characterized by vast amounts of data, such as social networks and chemistry, a need arises to optimize and accelerate their capabilities. Inference in GNNs refers to the time the model takes to make predictions after training. The duration of the inference process determines the speed at which queries are answered, and researchers strive to minimize this time span.

In applications of deep learning that prioritize low latency, FPGAs outperform other computing devices, such as CPUs and GPUs, by providing superior performance. FPGAs offer the advantage of being fine-tuned to strike the optimal balance between power efficiency and meeting performance requirements.

Due to this reason, researchers have been actively pursuing the development of new FPGA accelerators for Graph Neural Networks (GNNs) in recent times.

The conventional approach to hardware design involves a combination of manual coding and automated processing. However, this method demands significant effort and relies heavily on the expertise of the designers, leading to varying quality of results.

To address these challenges, the objective of this thesis research study is to develop a comprehensive toolchain that, starting from PyTorch [16], a cutting-edge high-level programming framework for creating neural network algorithms based on the Python programming language, enables the automatic generation of a Graph Neural Networks (GNNs) FPGA accelerator with minimal effort required.

The suggested toolchain represents an enhancement of the SODA toolchain [2]. It operates by transforming the PyTorch model, provided as input, into a multi-level intermediate representation (MLIR) [14] utilizing Torch-MLIR [20], an MLIR based compiler toolkit for PyTorch programs. This MLIR representation is then passed to the SODA framework to conduct hardware/software partitioning of the algorithm specifications and architecture-independent optimizations. Following this, the framework generates a low-level IR (LLVM IR) specifically tailored for the hardware generation engine, PandA-Bambu [7].

In pursuit of the thesis goal, various optimizations were adopted throughout the process. Specifically, efforts were made to optimize specific computations in Graph Neural Networks during the experimental phase. As these networks often deal with massive graph sizes, the computation time and memory requirements are substantial. Consequently, a significant portion of the research focuses on optimizing the computation phase of Graph Neural Networks using tailored SODA optimizations, particularly matrix multiplication.

Furthermore, limitations and challenges have been encountered along the way. Another objective of this thesis is to analyze these limitations, ensuring they are clearly understood thoroughly. This analysis aims to provide valuable insights for future research endeavors, enabling the development of solutions to overcome these limitations and further enhance the proposed toolchain.

While the intended purpose of the toolchain is to be general, the experimental phase primarily focused on two specific types of Graph Neural Networks: Graph Isomorphism Networks (GIN) [22] and Graph Convolutional Networks (GCN) [13]. These models were sourced from reliable GitHub implementations and were modified as necessary.

The GCN model [19], designed for node classification task and written in pure PyTorch, held particular importance for the experimental phase as it served as the basis for the resulting accelerator. On the other hand, the GIN model [18], designed for graph classification task and written in PyTorch Geometric [8], a library built upon PyTorch for easier development and training of Graph Neural Networks, did not progress through the final step of the proposed toolchain. This was due to some incompatibilities between PyTorch Geometric and Torch-MLIR, which are integral parts of this thesis research.

1.1 Contributions

1.2 Thesis structure

2 Background

Background here...

3 Conclusions and future developments

Final chapter containing the main conclusions of my research and possible future developments.

Bibliography

[1] S. Abi-Karam, Y. He, R. Sarkar, L. Sathidevi, Z. Qiao, and C. Hao. Gengnn: A generic FPGA framework for graph neural network acceleration. *CoRR*, abs/2201.08475, 2022. URL `https://arxiv.org/abs/2201.08475`.

[2] N. B. Agostini, S. Curzel, J. J. Zhang, A. Limaye, C. Tan, V. Amatya, M. Minutoli, V. G. Castellana, J. Manzano, D. Brooks, G.-Y. Wei, and A. Tumeo. Bridging python to silicon: The soda toolchain. *IEEE Micro*, 42(5):78–88, 2022. doi: 10.1109/MM.2022.3178580.

[3] A. Auten, M. Tomei, and R. Kumar. Hardware acceleration of graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020. doi: 10.1109/DAC18072.2020.9218751.

[4] A. Bik, P. Koanantakool, T. Shpeisman, N. Vasilache, B. Zheng, and F. Kjolstad. Compiler support for sparse tensor computations in MLIR. *ACM Transactions on Architecture and Code Optimization*, 19(4):1–25, sep 2022. doi: 10.1145/3544559. URL `https://doi.org/10.1145%2F3544559`.

[5] U. Bondhugula. High performance code generation in MLIR: an early case study with GEMM. *CoRR*, abs/2003.00532, 2020. URL `https://arxiv.org/abs/2003.00532`.

[6] S. Böhm. How to optimize a cuda matmul kernel for cublas-like performance: a worklog, 2022. URL `https://siboehm.com/articles/22/CUDA-MMM`.

[7] F. Ferrandi, V. G. Castellana, S. Curzel, P. Fezzardi, M. Fiorito, M. Lattuada, M. Minutoli, C. Pilato, and A. Tumeo. Invited: Bambu: an open-source research framework for the high-level synthesis of complex applications. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1327–1330, 2021. doi: 10.1109/DAC18074.2021.9586110.

[8] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. URL `http://arxiv.org/abs/1903.02428`.

[9] L. He. Engn: A high-throughput and energy-efficient accelerator for large graph

neural networks. *CoRR*, abs/1909.00155, 2019. URL `http://arxiv.org/abs/1909.00155`.

[10] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22118–22133. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf`.

[11] Y. Hu, Y. Du, E. Ustun, and Z. Zhang. Graphlily: Accelerating graph linear algebra on hbm-equipped fpgas. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2021. doi: 10.1109/ICCAD51958.2021.9643582.

[12] K. Kiningham, C. Ré, and P. A. Levis. GRIP: A graph neural network accelerator architecture. *CoRR*, abs/2007.13828, 2020. URL `https://arxiv.org/abs/2007.13828`.

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

[14] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko. Mlir: Scaling compiler infrastructure for domain specific computation. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 2–14, 2021. doi: 10.1109/CGO51591.2021.9370308.

[15] S. Liang, C. Liu, Y. Wang, H. Li, and X. Li. Deepburning-gl: an automated framework for generating graph neural network accelerators. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2020.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL `http://arxiv.org/abs/1912.01703`.

[17] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 2021. doi: 10.23915/distill.00033. https://distill.pub/2021/gnn-intro.

[18] O. G. B. team and other contributors. Gnn models from open graph benchmark,

2020. URL https://github.com/snap-stanford/ogb/tree/master/examples/graphproppred/mol.

[19] M. W. Thomas N. Kipf and other contributors. Gcn model in pytorch, 2017. URL https://github.com/tkipf/pygcn.

[20] n. t. Torch-MLIR team and other contributors. Torch-mlir: Mlir based compiler toolkit for pytorch programs, 2021. URL https://github.com/llvm/torch-mlir.

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. URL http://arxiv.org/abs/1901.00596.

[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks?, 2019.

[23] M. Yan, L. Deng, X. Hu, L. Liang, Y. Feng, X. Ye, Z. Zhang, D. Fan, and Y. Xie. Hygcn: A GCN accelerator with hybrid architecture. *CoRR*, abs/2001.02514, 2020. URL http://arxiv.org/abs/2001.02514.

[24] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL http://arxiv.org/abs/1812.08434.

A Appendix A

If you need to include an appendix to support the research in your thesis, you can place it at the end of the manuscript. An appendix contains supplementary material (figures, tables, data, codes, mathematical proofs, surveys, . . .) which supplement the main results contained in the previous chapters.

B Appendix B

It may be necessary to include another appendix to better organize the presentation of supplementary material.

List of Figures

List of Tables

List of Symbols

Variable	**Description**	**SI unit**
$\boldsymbol{u}$	solid displacement	m
$\boldsymbol{u}_f$	fluid displacement	m

Acknowledgements

Acknowledgements here...