Please do not send me emails asking for application help. I wouldn't be of much help – nor am I qualified to give advice. Here is my experience: https://davidyue.me/posts/graduate-application-process/. Please do not treat this as gospel either.

Statement of Purpose

I am interested in pursuing a Ph.D. in compiler design and program optimization. Having worked in the ML research industry for nearly 2 years, I am convinced that efficient utilization of hardware remains a bottle neck for high performance computing – compilers seek to bridge this gap.

Experience with Compilers My interest in compilers began during an independent study under Dr. Thierry Sans, where I explored Robert Harper's *Practical Foundations for Programming Languages* [1] and Robert Nystrom's *Crafting Interpreters* [2]. Each week, I presented insights from my readings [1], [2] and provided updates on my programming language project, which I extended to include Golang-inspired asynchronous routines. Since then I have independently continued exploring different compiler optimization techniques [3], [4], [5], [6] such as the Static Single Assignment (SSA) Form [7], [8] translation algorithms, register allocation algorithms [9] and interprocedural analysis [10]. This work culminated in a compiler IR framework project [11]. My independence, motivation, and initiative to familiarize myself with compiler theory and to learn new tools like LLVM position me well to pursue a Ph.D. in compiler optimization.

Experience with Machine Learning As a research assistant at Huawei, I worked with the MindSpore ML framework and compiler, specifically to assess the feasibility of porting PyTorch models to MindSpore for Ascend NPU chips. PyTorch's dynamic computation graphs posed a technical challenge against MindSpore's reliance on static computation graphs, which do not support varying tensor sizes during training or inference. By analyzing MindSpore's source code and IR emitted at each pipeline stage, I uncovered static vs. dynamic graph incompatibilities, leading our team to deem PyTorch model porting infeasible. This furthered my understanding of compiler architectures and deepened my interest in tackling optimization challenges specific to ML models—ideal preparation for research in compiler design and optimization.

My work has also encompassed model compression and optimization techniques, where I studied both vLLM [12] and LoRA [13]. While vLLM applies a virtual memory-inspired approach to enhance memory efficiency via paging for the Key Value cache, LoRA reduces the rank of weight matrices by storing them as low-rank matrix products. Leading a research project focused on weight decomposition, I applied LoRA's fine-tuning methodology with the objective of training models for multiple tasks. This method involved splitting model weights into a sum of task-specific matrices, allowing the sum of any subset of matrices to represent combined task weights. I initially pursued a model based on low-rank matrices and weight masks but ultimately pivoted to a singular value decomposition-based approach - yielding marked improvements in model performance and validating our strategy. This experience, centered on designing and optimizing models, parallels key challenges in ML compiler research—namely, balancing memory efficiency and performance. My experience in applying techniques like SVD and low-rank approximations demonstrates my systematic approach to tackling optimization problems. Successfully leading this project showcases my capability for independent research.

Lastly, I also contributed to developing robust data watermarking techniques for ML models, embedding imperceptible watermarks in large language models (LLMs) to establish ownership and protect intellectual property. These watermarks preserved normal model behavior, utilizing "passthrough" layers that functioned as identity mappings under standard use but triggered specific outputs when provided a secret key. I experimented with various training strategies to enhance the resilience

of these passthrough layers against adversarial attacks, such as downstream finetuning and layer deletion, finding that increasing the layer count significantly bolstered robustness. This work provided valuable insights into PyTorch's handling of backpropagation and computation graph construction. Training the passthrough layers independently of the LLM required careful layer freezing and gradient management to ensure precise updates, deepening my understanding of computation graphs and their integration with ML compilers—a good foundation for compiler research. This project culminated in the paper "Task-Agnostic Language Model Watermarking via High Entropy Passthrough Layers," accepted by AAAI'25.

Teaching and Education I have been an active member of Dr. Brian Harrington's Computer Science Education Undergraduate Research Group, leading the reading group since early 2020. In this role, I analyzed and presented five research papers, guiding discussions on methodologies and findings. I also contributed to a literature mapping paper, "Finding and Categorizing COVID-19 Papers in CS Education", accepted by SIGCSE 2023. Reviewing 37 papers, I assessed how the COVID-19 pandemic impacted computer science education, gaining valuable insights into pedagogical challenges and student outcomes during times of disruption. This research directly informed my role as a teaching assistant, where I adapted office hours and tutorials to better support student needs amid remote learning challenges. As a repeat TA for multiple courses, I applied research-driven strategies to create accessible and supportive learning environments. This effort earned me a nomination for the "Computer and Mathematical Sciences Teaching Assistant Award," and strengthened my ability to communicate complex ideas.

Academics As a co-op student, I balanced an academic workload alongside the search for and completion of work terms. Even beyond my official co-op placements, I sought additional internships, adding further pressure to my studies. In my second-to-last semester, to graduate quickly, I took on an especially challenging course load—a choice that, in hindsight, impacted my grades. However, these experiences taught me to face adversity head-on and reinforced the importance of resilience. I improved my Linear Algebra grade from a D in the first course to an A- in the second. I initially received a low grade in Artificial Intelligence, but later demonstrated my competence through research and publications in the field. Likewise, despite earning a low grade in Computer Organization, I excelled in Operating Systems, a continuation course, with an A+.

Future Work I am interested in working with Dr. John Regehr, Dr. Mary Hall, and Dr. Ponnuswamy Sadayappan. Dr. Regehr's work on automating peephole optimizations [14], CSmith [15], and CReduce focuses on identifying and addressing elusive bugs in compiler code generation. His recent use of program synthesis for SIMD instruction optimization [16] is particularly relevant to my interests. I would be excited to collaborate with Dr. Regehr to expand synthesis techniques beyond expression equivalences and integrate AI-guided program synthesis, leveraging my background in LLMs. Dr. Hall's research on BrickDL [17], which addresses GPU data movement bottlenecks, resonates with my experience in model compression and optimization. I am eager to work with Dr. Hall to develop ML compilers that minimize GPU data transfers and enhance memory efficiency. Her work on code synthesis for sparse tensor format conversions also presents an opportunity to explore LLMs for automating such tasks. Dr. Sadayappan's work on low-rank matrix factorization [18] aligns with my experience in LoRA-inspired weight decomposition for model compression. His research on accelerator data flow co-design for convolutional neural networks complements my work on ML compilers for Ascend NPUs. I am enthusiastic about collaborating with Dr. Sadayappan to design ML compilers that optimize memory usage and improve inference times for deep learning models. After completing my Ph.D., I intend to pursue an academic career and land a professorship at a well-renowned university like the University of Utah.

Bibliography

[1] R. Harper, *Practical Foundations for Programming Languages*, 2nd ed. Cambridge University Press, 2016.

[2] N. Robert, "Crafting Interpreters." 2021.

[3] T. VanDrunen and A. L. Hosking, "Value-Based Partial Redundancy Elimination," in *Compiler Construction*, E. Duesterwald, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 167–184.

[4] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[5] L. Torczon and K. Cooper, *Engineering A Compiler*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[6] P. Briggs, K. D. Cooper, and L. Torczon, "Rematerialization," *SIGPLAN Not.*, vol. 27, no. 7, pp. 311–321, July 1992, doi: 10.1145/143103.143143.

[7] M. Braun, S. Buchwald, S. Hack, R. Leißa, C. Mallon, and A. Zwinkau, "Simple and Efficient Construction of Static Single Assignment Form," in *Compiler Construction*, R. Jhala and K. De Bosschere, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 102–122.

[8] F. Rastello, *SSA-based Compiler Design*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[9] L. George and A. W. Appel, "Iterated register coalescing," *ACM Trans. Program. Lang. Syst.*, vol. 18, no. 3, pp. 300–324, May 1996, doi: 10.1145/229542.229546.

[10] "CS 6120: Advanced Compilers: The Self-Guided Online Course."

[11] Y. David, "Compiler IR Framework." GitHub, 2024.

[12] W. Kwon *et al.*, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[13] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." [Online]. Available: https://arxiv.org/abs/2106.09685

[14] M. Mukherjee and J. Regehr, "Hydra: Generalizing Peephole Optimizations with Program Synthesis," *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, Apr. 2024, doi: 10.1145/3649837.

[15] X. Yang, Y. Chen, E. Eide, and J. Regehr, "Finding and understanding bugs in C compilers," in *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, in PLDI '11. San Jose, California, USA: Association for Computing Machinery, 2011, pp. 283–294. doi: 10.1145/1993498.1993532.

[16] Z. Liu, S. Mada, and J. Regehr, "Minotaur: A SIMD-Oriented Synthesizing Superoptimizer." [Online]. Available: https://arxiv.org/abs/2306.00229

[17] M. Lakshminarasimhan, M. Hall, S. Williams, and O. Antepara, "BrickDL: Graph-Level Optimizations for DNNs with Fine-Grained Data Blocking on GPUs," in *Proceedings of the 53rd International Conference on Parallel Processing*, in ICPP '24. Gotland, Sweden: Association for Computing Machinery, 2024, pp. 576–586. doi: 10.1145/3673038.3673046.

[18] A. Gupta, S. M. Saravani, P. Sadayappan, and V. Srikumar, "An Empirical Investigation of Matrix Factorization Methods for Pre-trained Transformers." [Online]. Available: https://arxiv.org/abs/2406.11307