POSTER: Diffusion-Based Data Augmentation for Multi-Label Performance Modeling

Mohammad Ali
Texas State University
San Marcos, Texas, USA
mohammad.ali@txstate.edu

Apan Qasem
Texas State University
San Marcos, Texas, USA
apan@txstate.edu

Abstract

Data scarcity remains a major challenge in machine-learning-based performance modeling, particularly in multi-label settings with underrepresented optimization outcomes. In this work, we investigate diffusion-based data augmentation for multi-label performance modeling, using compiler optimization sequence prediction as the target application. The proposed approach generates synthetic feature-label pairs that improve the representation of sparse optimization patterns while preserving the underlying relationships between program characteristics and optimization outcomes. Experimental results show that synthetic augmentation improves prediction accuracy to 92%. We further evaluate the predicted optimization sequences, which achieve an average instruction-count reduction of 46.33%, closely matching LLVM -Oz while requiring substantially shorter sequences. These results demonstrate the potential of diffusion-based augmentation to improve multi-label performance modeling while reducing dependence on costly labeled data.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Interpreters**; **Source code generation**; • **General and reference** → *Evaluation*.

Keywords

Compiler Optimization, Multi-Label Learning, Gaussian Diffusion, Performance Modeling, Label Imbalance

ACM Reference Format:
Mohammad Ali and Apan Qasem. 2026. POSTER: Diffusion-Based Data Augmentation for Multi-Label Performance Modeling. In *International Conference on Parallel Architectures and Compilation Techniques (PACT '26), October 19–22, 2026, Chicago, IL, USA*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3838684.3846859

1 Introduction

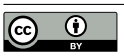
Machine-learning-based compiler optimization can be an effective alternative to expensive iterative search and autotuning by learning relationships between program characteristics and optimization decisions [1]. Prior work has demonstrated the use of learned models for selecting compiler optimization sequences and guiding optimization decisions [3, 17]. Frameworks such as MLGO [14] and CompilerGym [5] further demonstrate the potential of machine learning for compiler optimization.

A major challenge in constructing such models is the cost of obtaining high-quality labeled performance data [11]. Identifying effective optimization sequences requires costly evaluation of many compiler configurations. Prior work has explored synthetic program and benchmark generation to alleviate data scarcity [4, 15, 16]. However, preserving relationships between program characteristics and optimization outcomes remains challenging. Optimization sequence prediction is naturally multi-label, as multiple sequences can produce similar performance for the same program. Restricting prediction to a single sequence can discard valid alternatives. Furthermore, the label distribution is highly imbalanced, with a few sequences appearing much more frequently than others.

In this work, we investigate diffusion-based synthetic augmentation for multi-label compiler optimization sequence prediction. Diffusion models learn complex data distributions through iterative denoising [8]. We jointly model program features and optimization labels while emphasizing underrepresented label combinations to improve label coverage and reduce dependence on costly labeled data. Our results show that synthetic augmentation improves prediction accuracy to 92%. The predicted sequences achieve an average instruction-count reduction of 46.33%, approaching the optimization effectiveness of LLVM `-Oz` while requiring substantially shorter optimization sequences. These results indicate that synthetic augmentation not only improves prediction accuracy but also enables the model to identify compact and effective optimization sequences for unseen programs.

2 Dataset

We construct our experimental dataset from AnghaBench, a large-scale collection of C programs [6]. Since the original benchmark does not provide the feature representations or optimization labels required for supervised learning, we derive both components. Each program is represented using the AutoPhase observation space [7], which extracts static program features from LLVM IR. We retain the top 10% of programs, ranked by their initial LLVM instruction count. The optimization labels are constructed from a reduced set of representative compiler optimization sequences, selected to provide effective coverage across programs while avoiding exhaustive exploration of the optimization space [12]. We apply these sequences to each program and measure the resulting instruction counts. The label set includes all sequences whose instruction counts fall within the best 5% for that program, forming a multi-label prediction task.



This work is licensed under a Creative Commons Attribution 4.0 International License.
PACT '26, Chicago, IL, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846859

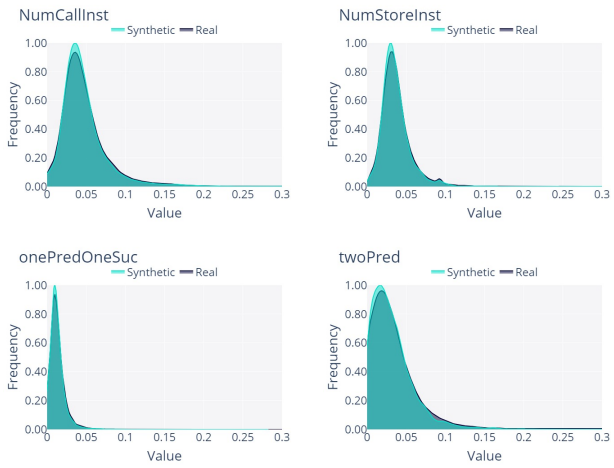


Figure 1: Comparison of real and synthetic feature distributions

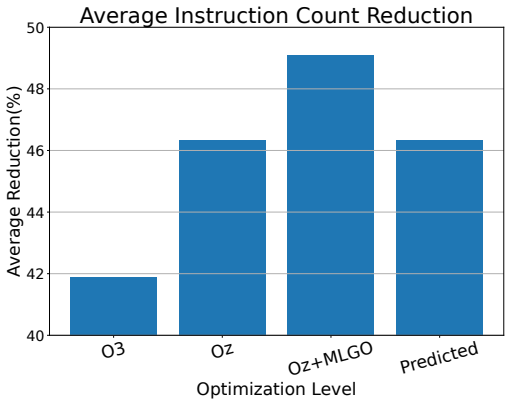


Figure 2: Average instruction-count reduction

3 Methodology

The proposed framework generates synthetic training samples for multi-label performance modeling. Starting with the original tabular dataset, we first retain the most informative program features and identify underrepresented optimization labels based on their occurrence frequency. The minority-focused sampling strategy [2] is used to emphasize sparsely represented regions of the label space during generative modeling.

To obtain a compact representation of the multi-label targets, we use the dictionary-encoding scheme [13] that maps each unique label combination to an index. The index is embedded and concatenated with the corresponding program features to form a continuous joint representation. We then employ a Gaussian diffusion model [10] to learn the dependencies between program characteristics and optimization labels. During generation, the model produces synthetic feature and label representations through iterative denoising. The label representation is decoded back to its original multi-label form, and the synthetic samples are combined with real training data to train the downstream optimization-sequence predictor. This targeted augmentation improves rare-pattern representation while preserving relationships between program characteristics and effective optimization sequences.

4 Experimental Results

The synthetic dataset achieves an average KS score of 97%, indicating strong statistical similarity with the real feature distributions. As shown in Fig. 1, the synthetic samples preserve the distributional shape, value range, and tail behavior of representative features. We next evaluate the utility of the synthetic data for optimization-sequence prediction using an MLP. Training on real data alone achieves 89% accuracy, while synthetic augmentation improves accuracy to 92% on the same real held-out test set. In comparison, augmentation using the SOTA diffusion model [10] achieves 87%.

Finally, we assess the effectiveness of the predicted optimization sequences by comparing instruction count reduction against LLVM optimization levels `-O3` and `-Oz`, as well as MLGO-guided optimization applied on top of `-Oz`. All results are reported as percentage reduction relative to the `-O0` baseline. Figure 2 summarizes the average reduction across 10,000 programs. Among the standard LLVM optimization levels, `-O3` achieves an average reduction of 41.89%, while `-Oz` achieves 46.34%. MLGO (`-Oz` + MLGO) delivers the best performance, achieving an average reduction of 49.10%. The predicted sequence model achieves an average reduction of 46.33%, consistently outperforming `-O3` and remaining competitive with `-Oz`, while staying within 2.77 percentage points of MLGO. LLVM optimization pipelines typically involve long and complex sequences of passes [9], whereas our approach uses substantially shorter optimization sequences without modifying the compiler pipeline or requiring iterative compiler interaction.

5 Conclusion

Diffusion-based synthetic augmentation improves multi-label compiler optimization sequence prediction, increasing accuracy from 89% to 92% while preserving high statistical fidelity. On unseen programs, the predicted sequences achieve a 46.33% instruction-count reduction, closely matching LLVM `-Oz` with substantially fewer optimization passes. These results show that synthetic augmentation can reduce reliance on costly labeled data while supporting effective compiler optimization.

References

[1] Amir H. Ashouri, William Killian, John Cavazos, Gianluca Palermo, and Cristina Silvano. 2018. A Survey on Compiler Autotuning using Machine Learning. *ACM Comput. Surv.* 51, 5, Article 96 (Sept. 2018), 42 pages. doi:10.1145/3197978

[2] Francisco Charte, Miguel Ángel Dávila, María Dolores Pérez-Godoy, and María José del Jesus. 2025. Addressing Multilabel Imbalance with an Efficiency-Focused Approach Using Diffusion Model-Generated Synthetic Samples. arXiv:2501.10822 [cs.LG] https://arxiv.org/abs/2501.10822

[3] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. End-to-End Deep Learning of Optimization Heuristics. In *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 219–232. doi:10.1109/PACT.2017.24

[4] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. Synthesizing benchmarks for predictive modeling. In *Proceedings of the 2017 International Symposium on Code Generation and Optimization, CGO 2017, Austin, TX, USA, February 4-8, 2017*. IEEE, 86–99. http://dl.acm.org/citation.cfm?id=3049843

[5] Chris Cummins, Bram Wasti, Jiadong Guo, Brandon Cui, Jason Ansel, Sahir Gomez, Somya Jain, Jia Liu, Olivier Teytaud, Benoit Steiner, Yuandong Tian, and

Hugh Leather. 2022. CompilerGym: robust, performant compiler optimization environments for AI research. In *Proceedings of the 20th IEEE/ACM International Symposium on Code Generation and Optimization* (Virtual Event, Republic of Korea) *(CGO '22)*. IEEE Press, 92–105. doi:10.1109/CGO53902.2022.9741258

[6] Anderson Faustino da Silva, Bruno Conde Kind, José Wesley de Souza Magalhães, Jerônimo Nunes Rocha, Breno Campos Ferreira Guimarães, and Fernando Magno Quinão Pereira. 2021. ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 378–390. doi:10.1109/CGO51591.2021.9370322

[7] Ameer Haj-Ali, Qijing (Jenny) Huang, John Xiang, William Moses, Krste Asanovic, John Wawrzynek, and Ion Stoica. 2020. AutoPhase: Juggling HLS Phase Orderings in Random Forests with Deep Reinforcement Learning. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. MLSys, 70–81.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 12 pages.

[9] Shalini Jain, Utpal Bora, Prateek Kumar, Vaibhav B. Sinha, Suresh Purini, and Ramakrishna Upadrasta. 2019. An analysis of executable size reduction by LLVM passes. *CSI Transactions on ICT* 7, 2 (2019), 105–110. doi:10.1007/s40012-019-00248-5

[10] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. TabDDPM: modelling tabular data with diffusion models. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 725, 16 pages.

[11] Hugh Leather and Chris Cummins. 2020. Machine Learning in Compilers: Past, Present and Future. In *2020 Forum for Specification and Design Languages (FDL)*. 1–8. doi:10.1109/FDL50818.2020.9232934

[12] Suresh Purini and Lakshya Jain. 2013. Finding good optimization sequences covering program space. *ACM Trans. Archit. Code Optim.* 9, 4, Article 56 (Jan. 2013), 23 pages. doi:10.1145/2400682.2400715

[13] Jacob Si, Zijing Ou, Mike Qu, and Yingzhen Li. 2024. TabUnite: Efficient Encoding Schemes for Flow and Diffusion Tabular Generative Models. https://openreview.net/forum?id=Zoli4UAQVZ

[14] Mircea Trofin, Yundi Qian, Eugene Brevdo, Zinan Lin, Krzysztof Choromanski, and David Li. 2021. MLGO: a Machine Learning Guided Compiler Optimizations Framework. arXiv:2101.04808 [cs.PL] https://arxiv.org/abs/2101.04808

[15] Foivos Tsimpourlas, Pavlos Petoumenos, Min Xu, Chris Cummins, Kim Hazelwood, Ajitha Rajan, and Hugh Leather. 2023. BenchPress: A Deep Active Benchmark Generator. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques* (Chicago, Illinois) *(PACT '22)*. Association for Computing Machinery, New York, NY, USA, 505–516. doi:10.1145/3559009.3569644

[16] Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. 2011. Finding and understanding bugs in C compilers. *SIGPLAN Not.* 46, 6 (June 2011), 283–294. doi:10.1145/1993316.1993532

[17] André Felipe Zanella, Anderson Faustino da Silva, and Fernando Magno Quintão. 2020. YACOS: a Complete Infrastructure to the Design and Exploration of Code Optimization Sequences. In *Proceedings of the 24th Brazilian Symposium on Context-Oriented Programming and Advanced Modularity* (Natal, Brazil) *(SBLP '20)*. Association for Computing Machinery, New York, NY, USA, 56–63. doi:10.1145/3427081.3427089