Optimizing Training Cost for Scalable Graph Processing

Han Hoang[1] Joshua Li[1] (Mentor) Lindsey Kostas[2] (Mentor) Dhiman Sengupta[2]

gihoang@ucsd.edu jol029@ucsd.edu lkostas@qti.qualcomm.com dhimseng@qti.qualcomm.com

[1]UCSD [2]Qualcomm

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Introduction

Unlike traditional place-and-route (PnR) methods, which iteratively refine layouts, data-driven chip design optimization leverages machine learning to address congestion more efficiently by predicting resource bottlenecks early and guiding design decisions to reduce costly iterations. Central to this approach is the netlist, a hypergraph representation of a circuit's connectivity, where nodes represent components (e.g., logic gates) and hyperedges represent electrical connections. By analyzing and optimizing resource demand through the netlist, this method directly enhances floorplanning—the process of arranging components on a 2D chip canvas—by providing insights that minimize congestion while optimizing Power, Performance, and Area (PPA) and meeting design constraints.

DE-HNN [1] is a state-of-the-art hypergraph neural network designed to predict congestion in chip design via demand regression. It outperforms other models by effectively capturing long-range dependencies through hierarchical virtual nodes, which aggregate node features within partitioned graph neighborhoods and propagate information efficiently, enabling more robust predictions.

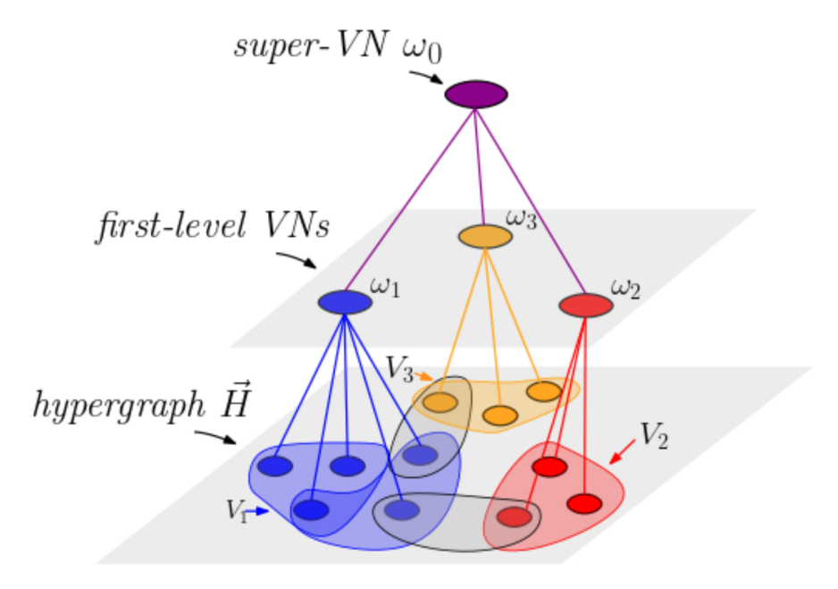


Figure 1. Illustration of DE-HNN Model Architecture

Research Motivation: While DEHNN delivers high performance, its scalability and practicality are limited by its significant computational overhead and lengthy runtimes.

Objective

This project aims to **minimize training costs** in terms of runtime and memory, while **preserving model performance** as much as possible. It focuses on **identifying cost-accuracy trade-offs** and developing a model architecture and training process that *balance computational efficiency with performance.*

Results (1)

Table 1. Performance Changes from Training Adjustments (in Percentage)

Adjustments	Node Loss	Net Loss	Runtime	Memory
ES[a]	4.9%	3.17%	84.37%	0%
ES[a]+ AA[b]	11.27%	3.65%	77.19%	38.83%
ES[a]+ AA[b]+ DLR[c]	6.4%	-17.6%	89.32%	38.83%

[*] Comparison is based against the baseline metrics
[a] Early Stopping
[b] Architecture Adjustment (Grid Search: 4 layers, 8 dimensions)
[c] Dynamic Learning Rate (Cyclical Learning Rate)

Conclusion

Our results demonstrate that DE-HNN is a highly expressive model with a fast convergence rate. We optimized DE-HNN to achieve maximum performance with minimal training using a simplified configuration **(4 layers, 8 dimensions)**, coupled with Cyclical Learning Rate scheduling and our custom Early Stopping condition.

- DE-HNN performs better on the test set with fewer training iterations.
- It has less than a 5% average performance drop with as few as 11 train iterations, compared to the baseline.
- Simpler configurations (e.g., 4 layers, 8 dimensions) outperform more complex ones (e.g., 4 layers, 32 dimensions).
- While simpler models require slightly more iterations, they are more efficient in runtime and memory, since the additional amount of training iterations is negligible.
- Cyclical Learning Rate accelerates convergence, allowing the Early Stopping condition to terminate training even earlier.

Results (2)

Figure 2. Loss Curves: Baseline vs. Baseline Early Stop Model

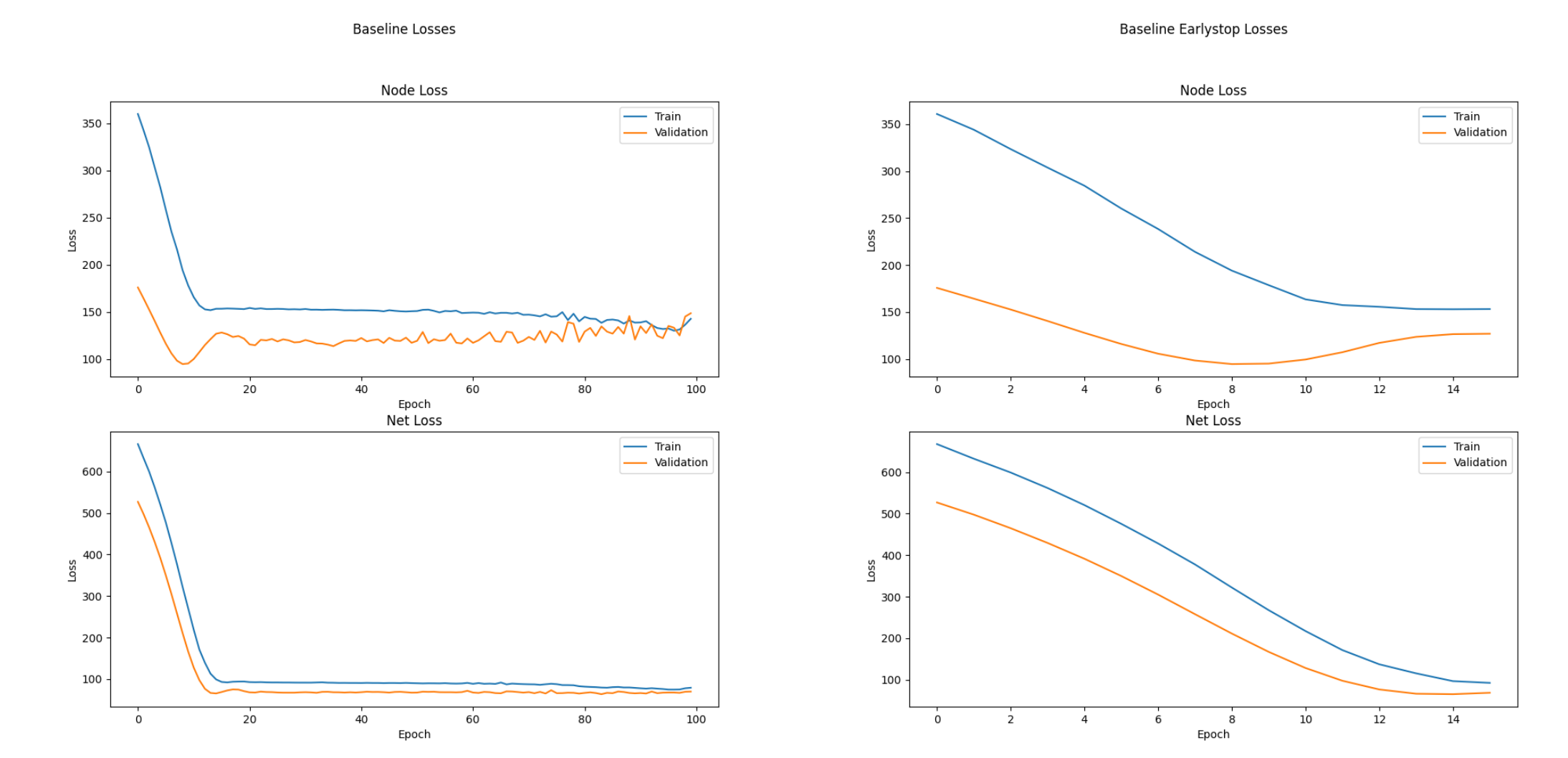
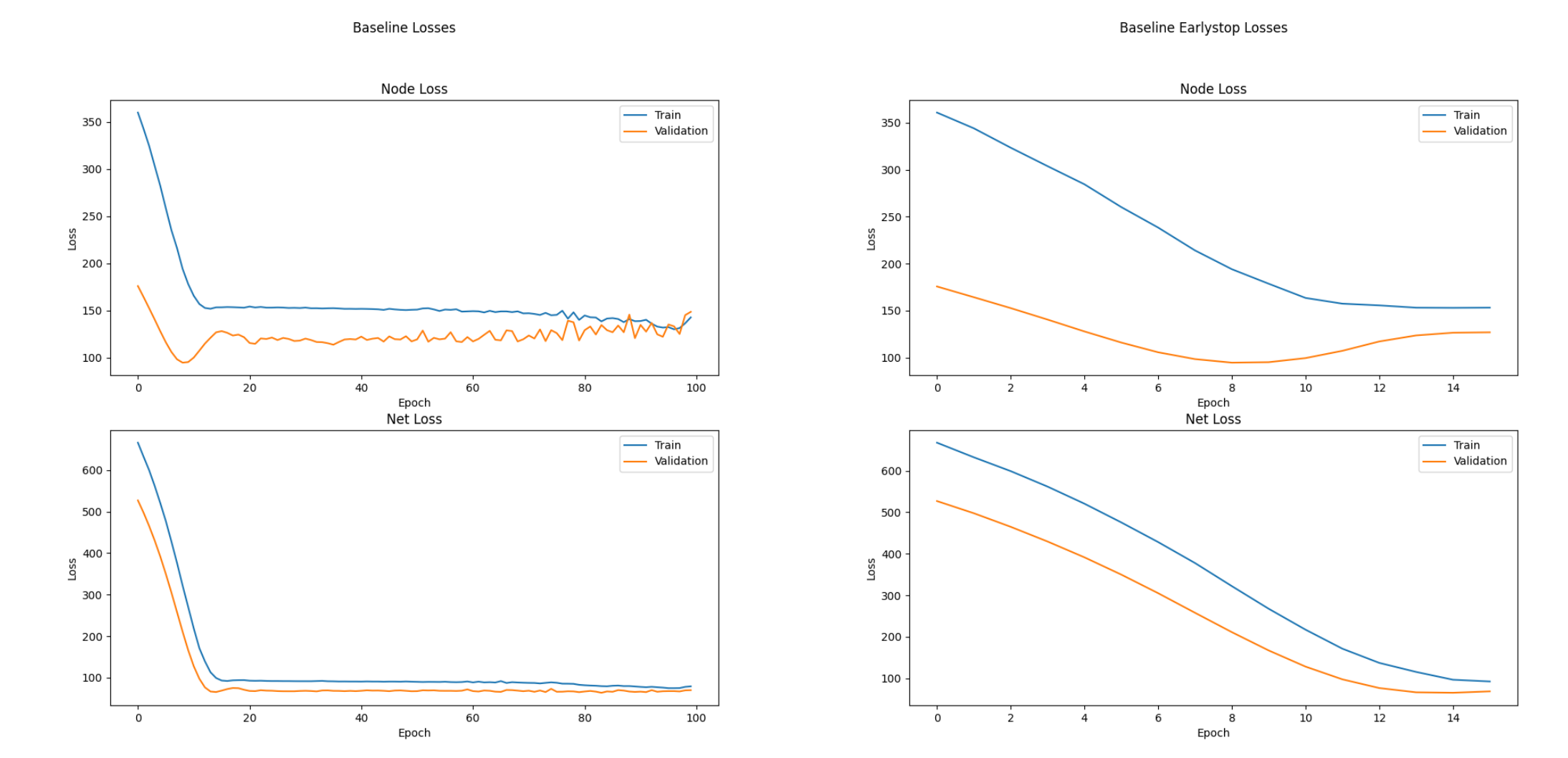


Figure 3. Loss Curves: Early Stop + Architecture Adjustment vs. Optimized Model



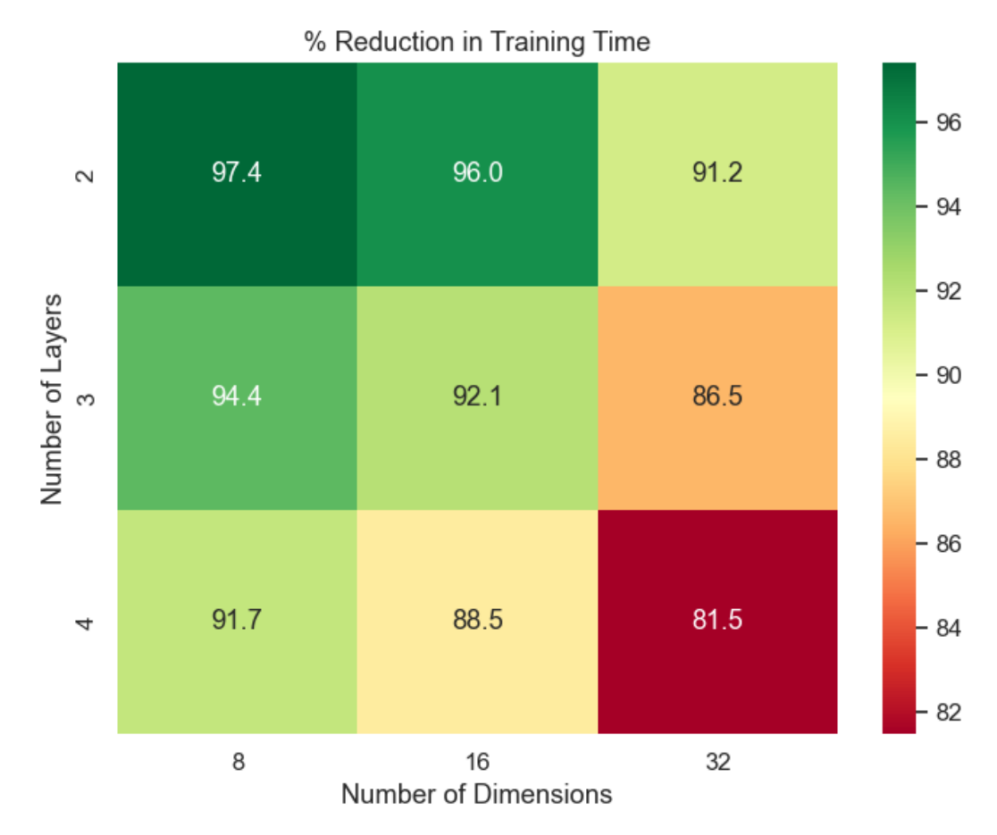


Figure 4. Reduction in Training Time (in Minutes)

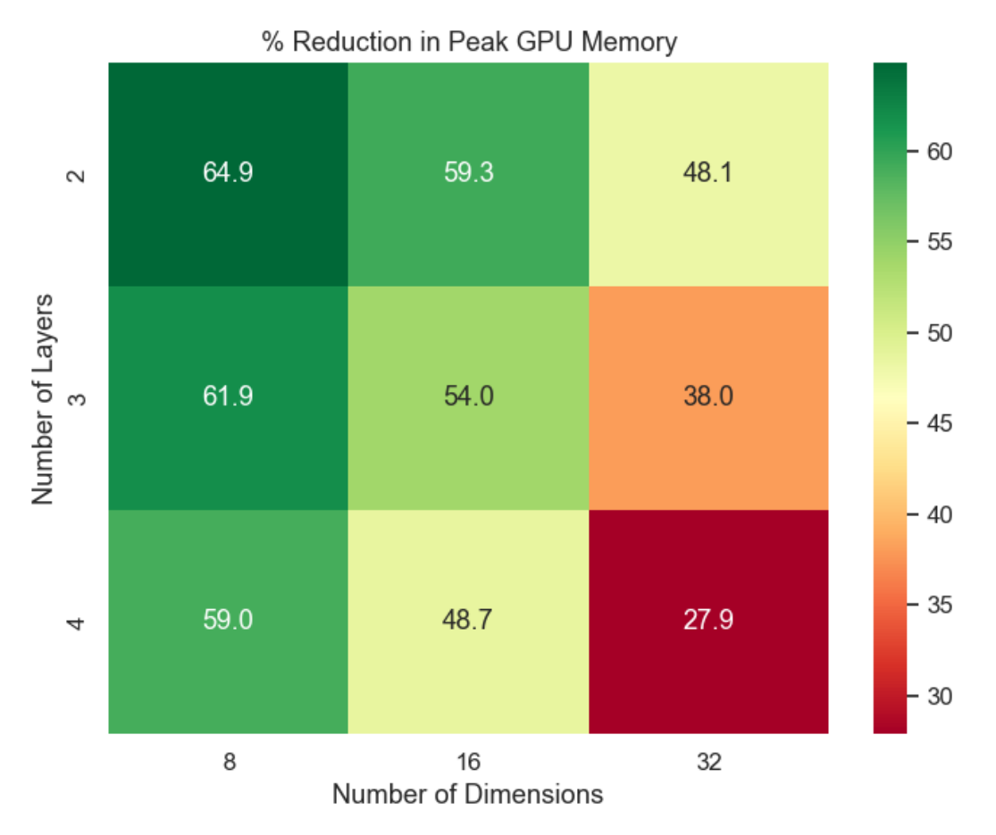


Figure 5. Reduction in Memory Usage

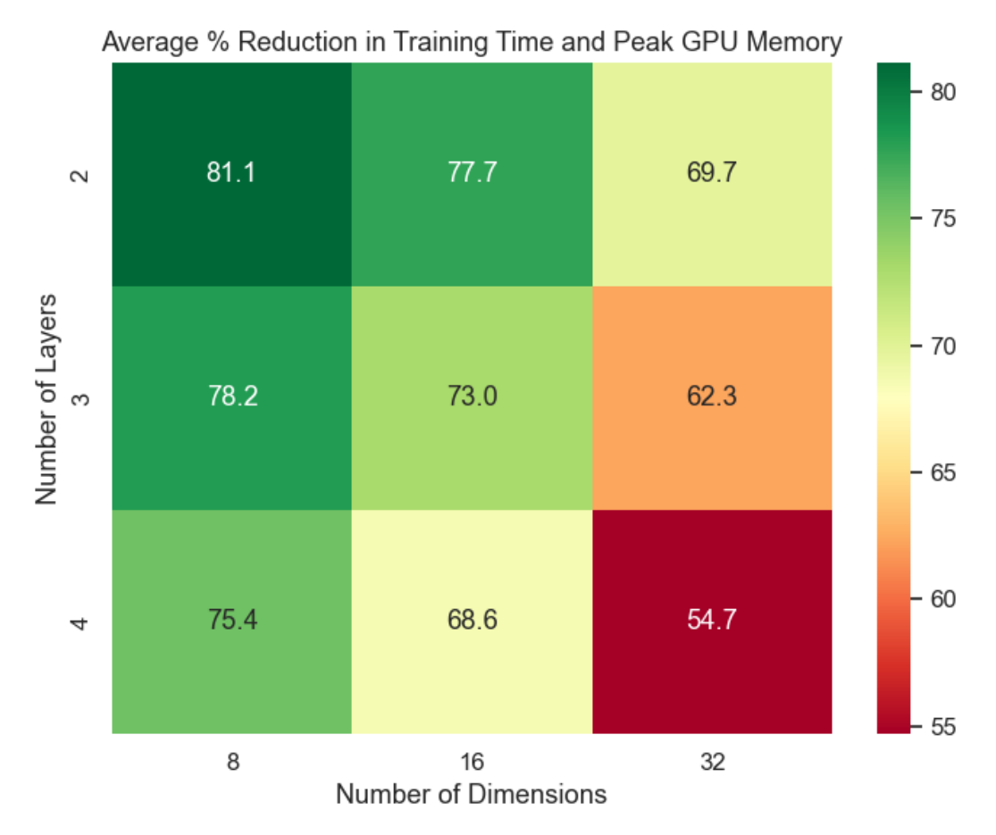


Figure 6. Average Reduction in Cost

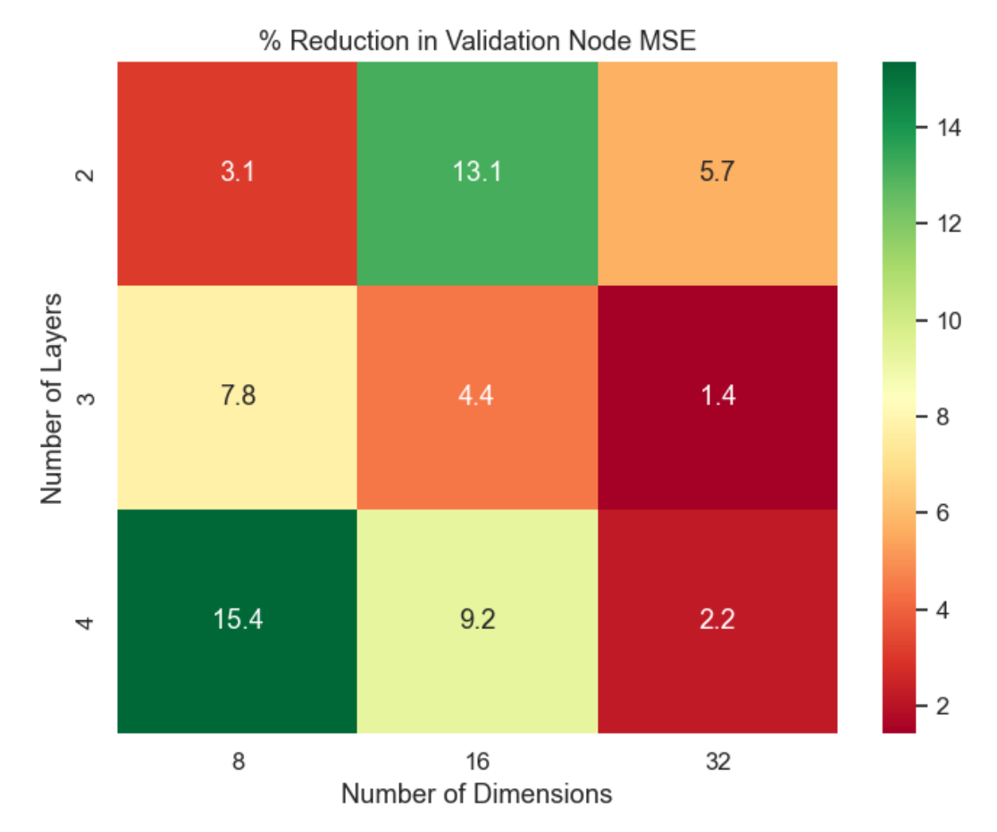


Figure 7. Reduction in Validation Node MSE

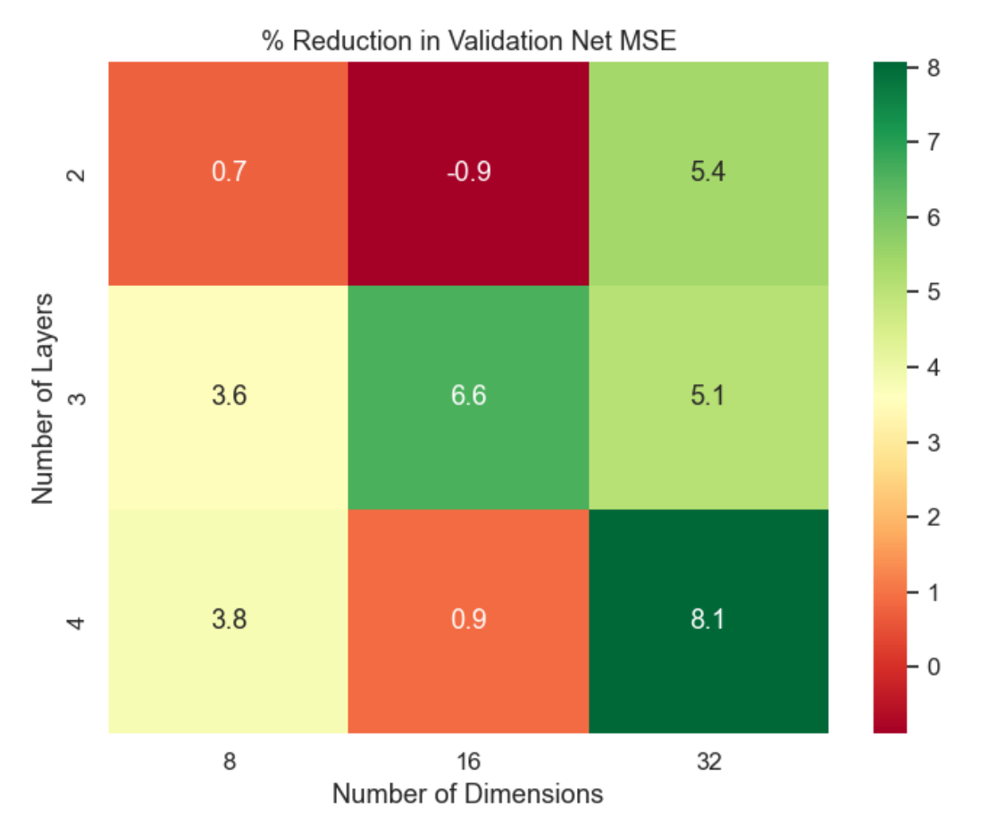


Figure 8. Reduction in Validation Net MSE

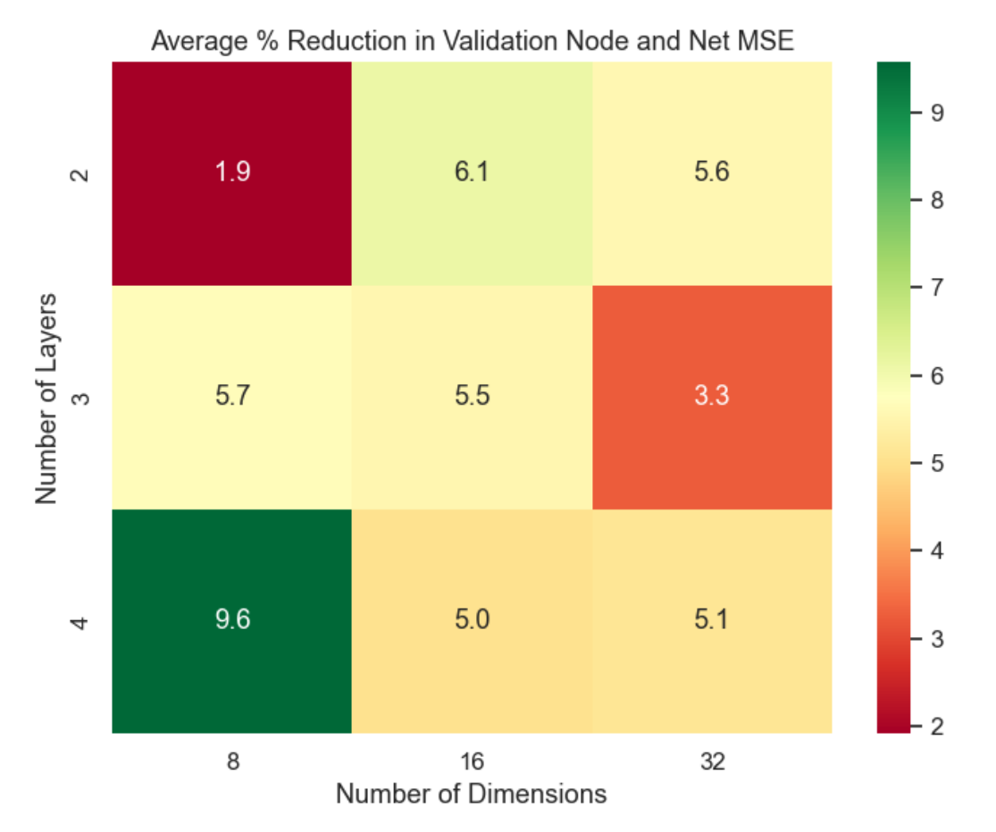


Figure 9. Average Reduction in Performance

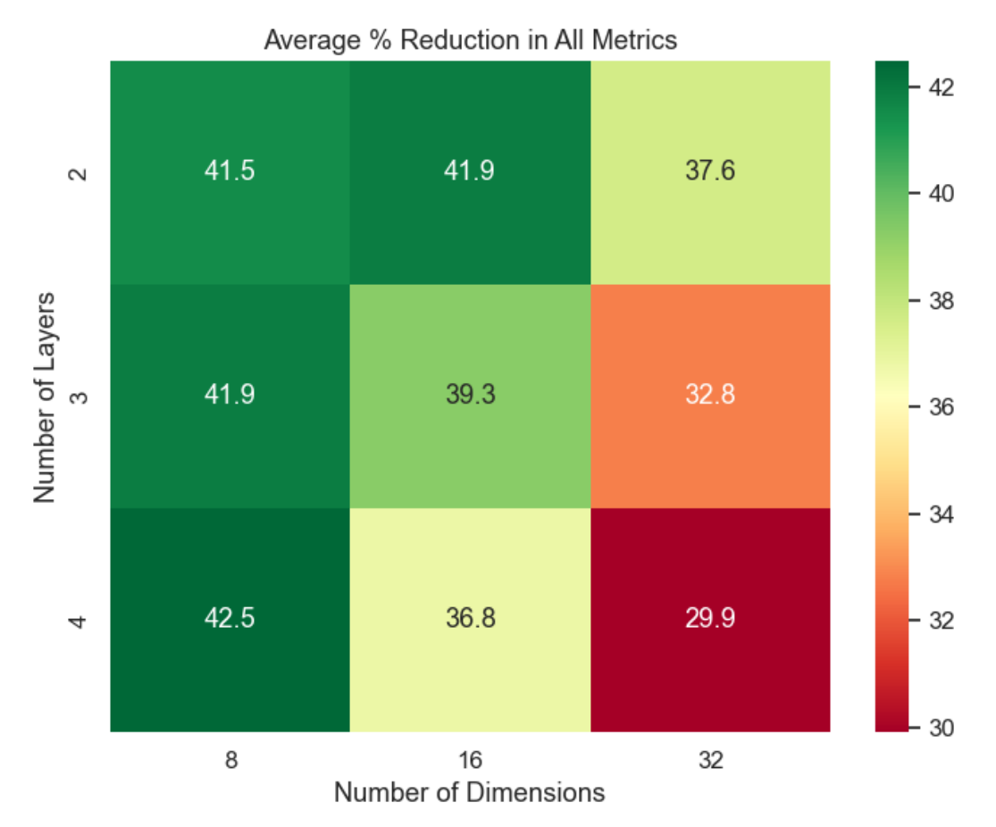


Figure 10. Average Reduction in All Metrics

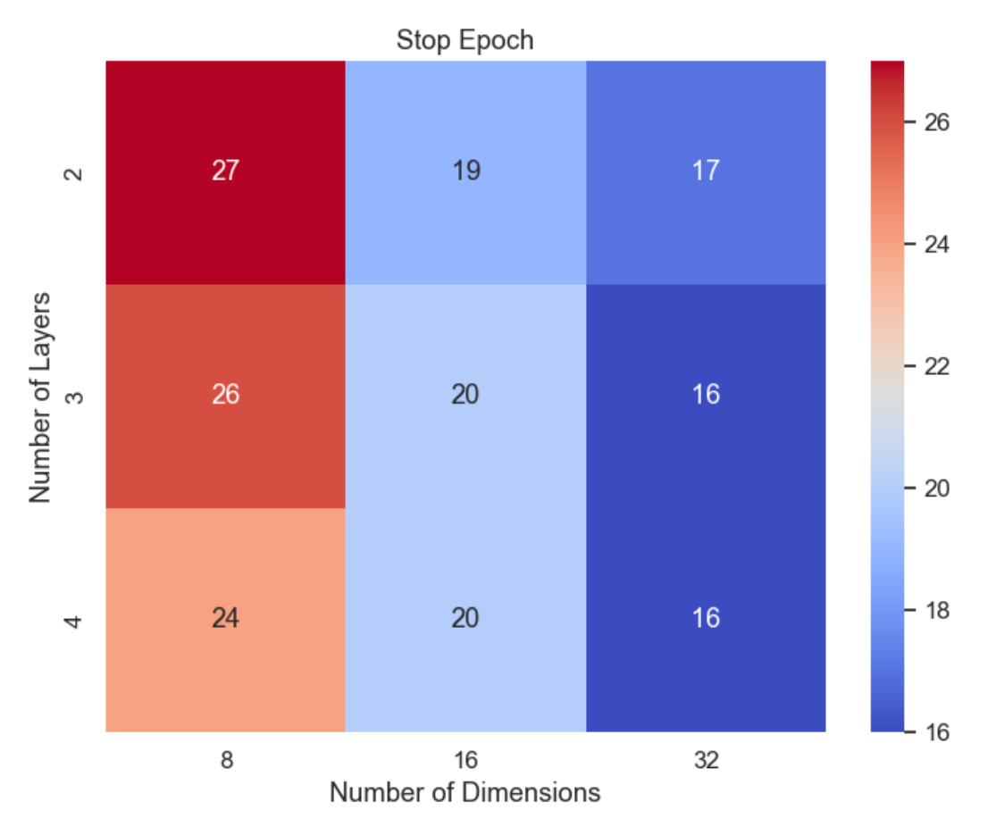


Figure 11. Early Stop Epoch

Methodology

Environment Setup: All experiments were conducted on a UCSD DSMLP cloud system with NVIDIA RTX A5000 GPU (24 GB VRAM), using PyTorch 2.2.2 and CUDA 12.2.

Dataset Description: The training dataset includes netlists [1-5], while netlist 6 is used for validation and testing. The chips vary in the number of nodes, nets (hyperedges), and edges (connections between nodes and nets). Each chip provides structural, spectral, connectivity, and virtual node features as inputs, with node and net demand as the target variables.

Netlist	# Nodes	# Nets	# Edges
1	797,938	821,523	2,950,019
2	923,355	954,144	3,459,373
3	604,921	627,036	2,358,662
4	671,284	696,983	2,532,180
5	459,495	468,888	1,942,114
6	810,812	830,308	3,107,242

Table 2. Dataset Characteristics

Training and Evaluation Strategy: Minimize mean squared error (MSE) on the training set for regression. Performance is monitored on the validation set during training, with final evaluation on the test set to assess generalization to unseen data.

Optimization Strategy: We apply **iterative optimization** by first using early stopping, followed by architecture adjustments based on Grid Search, and finally incorporating a dynamic learning rate, while using a fixed random seed throughout for weight initialization ensures reproducibility by controlling the effects of each optimization component.

- **Early Stopping (ES):** A custom condition using two parameters: *patience* and *tolerance*. Training is halted when the validation loss surpasses the average loss of the preceding epochs specified by patience. This helps prevent overfitting while optimizing training efficiency.
- **Architecture Adjustments (AA):** Use Grid search to explore combinations of two hyperparameters, the number of layers and dimensions, and identify cost-accuracy trade-offs.
- **Dynamic Learning Rate (DLR):** Uses Cyclical Learning Rate (CLR) scheduler, which adjusts the learning rate cyclically between a minimum and maximum value to accelerate convergence and avoid local minima.

Baseline Model: The DE-HNN baseline consists of 3 layers with 32 dimensions, a learning rate of 0.001, included virtual nodes, and trained for 100 epochs.

Discussion

To improve the robustness of our findings, future work could involve applying the optimization techniques across multiple random seeds and averaging the results. This approach would help mitigate the variability introduced by stochastic processes, providing a more reliable evaluation of the optimization strategies

Acknowledgement

We gratefully acknowledge the invaluable guidance and support of our primary mentors, Lindsey Kostas and Dhiman Sengupta. We also extend our thanks to the other mentors in our section for their expertise and assistance:

- Alex Hagen (`alexhage@qti.qualcomm.com`)
- Animesh Basak Chowdhury (`abasakch@qti.qualcomm.com`)
- Arnav Ballani (`aballani@qti.qualcomm.com`)
- Elahe Rezaei (`erezaei@qti.qualcomm.com`)
- Masoud Sadeghian (`msadeghi@qti.qualcomm.com`)

References

[1] Zhishang Luo, Truong Son Hy, Puoya Tabaghi, Donghyeon Koh, Michael Defferrard, Elahe Rezaei, Ryan Carey, Rhett Davis, Rajeev Jain, and Yusu Wang. DE-HNN: An effective neural model for Circuit Netlist representation. *arXiv preprint arXiv:2404.00477*, 2024.