Enabling Large Dynamic Neural Network Training with Learning-based Memory Management on Tiered Memory

Jie Ren*, Dong Xu§, Shuangyan Yang§, Jiacheng Zhao‡, Zhicheng Li‡, Christian Navasca†, Chenxi Wang‡, Harry Xu†, and Dong Li§
* William & Mary, ‡ University of Chinese Academy of Sciences,
† University of California, Los Angeles, § University of California, Merced

1 Introduction

Deep learning (DL) is embracing dynamic neural network (NN) architectures where the NN structure changes across data samples [8]. Such dynamic neural networks (DyNN) are different from the traditional static NN where a network architecture (i.e., a dataflow graph) is defined using symbolic expressions before execution and fixed during execution. A DyNN model may select its model components (e.g., layers [11], channel [14] or sub-networks [30]) conditional on input samples, and change the structure and parameters in the dataflow graph accordingly. DyNN decouples the requirement for many parameters from computational costs, which leads to reduction of training cost. Previous works [4, 7, 26, 31, 34, 38] show that compared with static NN, DyNN reduces training cost yet improve model prediction performance. DyNNs have shown high computational efficiency over variable-length sequences [32], trees [33], and graphs [13]. They have also demonstrated strong representation capabilities and high adaptiveness in achieving desired tradeoffs between accuracy and efficiency on the fly [8]. As a result, DyNNs have been applied to many problems, such as speech recognition [36], language modeling [7, 26, 27, 37], image recognition [2, 5] and DL translation [3, 32, 34]. Recently, DyNNs are applied to large language models (such as GLaM from Google [6]), pushing the limit of scaling laws in the age of generative models. It is believed that the DyNN is one of a few techniques to improve efficiency and resource utilization of future large models [15].

2 Motivation

DyNNs, as many other NNs, are often memory hungry [19, 23, 24, 39]. This is especially the case as large models are gaining increasing popularity. For example, AlphaFold [28], a DyNN model based on evoformers (a variant of transformer) recently making breakthrough in protein structure prediction, consumes 1,024 GB memory when using 128 amino acids sequences of 256 in length [1]. As another example, a switch-based mixture-of-expert (MoE) model with the similar parameter efficiency as T5-large (a static natural language processing model) consumes at least 320 GB memory [21]. Clearly, training of large models is fundamentally limited by GPU memory capacity. Distributed parallel training techniques such as pipeline parallelism [16, 17] and tensor model parallelism [29] go beyond the memory boundary of single GPU by splitting model states across multiple GPUs, enabling training of massive models that would otherwise not fit into a single GPU's memory. However, these techniques require enough GPUs to provide large aggregated GPU memory to store the model states necessary for training; these GPUs can be extremely expensive and beyond the affordability of many small companies and organizations [22, 24].

Exploiting CPU memory to reduce the need of GPU memory for large model training has been explored [9, 10, 19, 20, 23–25]. Although tensor offloading to CPU memory is effective in training static models, it is hard to be applied to DyNNs. In particular, effectively using heterogeneous memory (CPU and GPU memories) requires minimizing the amount of communication between CPU and GPU or hiding communication. To achieve this goal, existing efforts rely on profiling-guided optimization (PGO) to record tensor access orders using a few training iterations and plan tensor prefetech between CPU and GPU for remaining iterations. PGO has a fundamental assumption: the NN model must be invariant, i.e., using a static computation graph where tensor dimensions as well as data and control flows are statically fixed, and there are no complex data structures (such as graphs and trees) in the dataflow graph. Hence, profiling a few training iterations is enough to decide tensor prefetch for upcoming operators.

However, the above assumption does not hold for DyNNs due to their inherent dynamism. Depending on the input, the DyNN selectively activates model components, introducing irregular memory accesses and invalidating profiling results collected in training iterations. As a result, communications between CPU and GPU are largely exposed to the critical path, leading to training throughput loss.

This paper presents a memory (tensor) management system, *DyNN-Offload*[1], for training large DyNNs. DyNN-Offload uses a new approach to guide tensor migration between CPU and GPU to maximize GPU memory efficiency. In particular,

[1]The full paper will appear in HPCA'24 and can be found at https://pasalabs.org/papers/2024/hpca24_dynn-offload.pdf

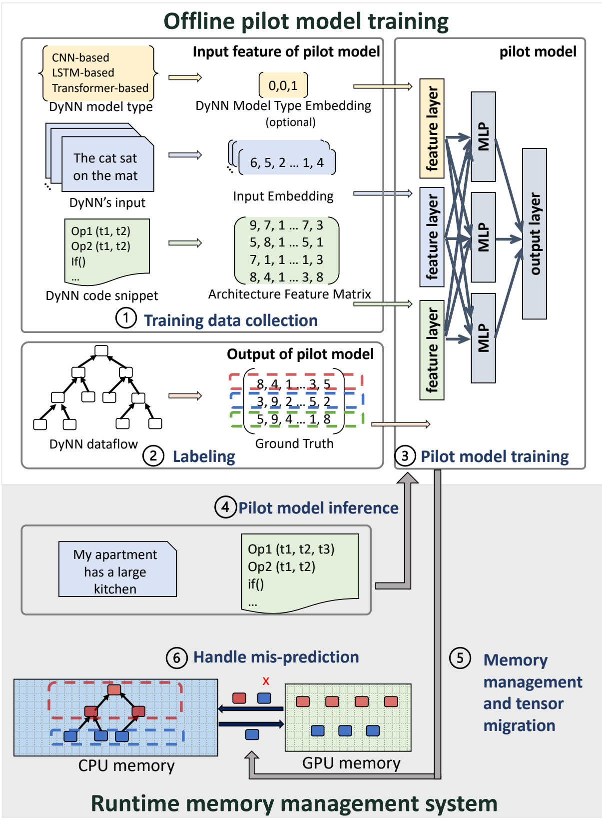

Figure 1. The workflow of DyNN-Offload.

we explore the extent to which a pilot model, such as an NN, can be used to increase predictability of tensor accesses during the training process of a large DyNN. We use the pilot model to timely prefetch tensors from CPU memory to GPU memory to hide communication overheads.

Research challenges. Developing a model for GPU memory management requires overcoming a number of challenges. The first is how to minimize the performance impact of querying the model (referred to as *pilot model*) for memory management. The inference using the pilot model introduces performance overheads to the critical path of DyNN training. The second challenge is how exactly to use the pilot model. DyNN-Offload queries the pilot model to decide *when to prefetch tensors* from CPU to GPU memory with the goal to maximize the overlapping between tensor migration and DyNN training. Tensor prefetching is critical in minimizing the overheads incurred from tensor migration. A possible idea is to build the pilot model to predict the exact execution order of operators. If this can be done, we could come up with a prefetch plan in a similar way to using PGO-guided tensor prefetch for static NNs. However, this approach requires rich output from the pilot model and high prediction accuracy, which leads to high inference overhead of the pilot model. Hence, there is an important tradeoff between the usefulness (to guide tensor prefetch) and performance overhead.

3 Overview

The overall architecture of DyNN-Offload comprises three main components shown in Figure 1.

The design of the pilot model centers around how to enable efficient enforcement and yet provide high pilot-model accuracy. We achieve this goal based on two observations: (1) operators in machine learning (ML), though rich in interfaces and algorithms, can be identified by a combination of *six pervasive and expressive memory access patterns*. (2) Tensors typically migrate in batches in order to fully utilize interconnect bandwidth. For those tensors that migrate together, there is no need to predict the exact execution order of the operators that reference the migrating tensors. This observation relaxes the requirement of using fine-grained execution order to plan tensor prefetch, which is the central technique used in all PGO-based solutions for static NNs [20, 23, 25, 35].

Based on the first observation, the input features and output of the pilot model can benefit from a compact representation based on six program idioms to *encode* the DyNN's architecture and indicate execution order of operators. This compact representation reduces the input feature space, leading to a simpler pilot model. Based on the second observation, the pilot model implicitly partitions a DyNN with resolved dynamism into multiple execution blocks, and only predicts the execution order of these blocks. This leads to an easier prediction task, and hence a lighter pilot-model and higher prediction accuracy. The above techniques address the challenge on the performance overhead of the pilot model.

To address the challenge in the planning of tensor prefetching, DyNN-Offload learns how to hide tensor migration through the training of the pilot model. During the pilot model training, the DyNN is transformed to a static one and then an existing PGO solution is used to decide execution blocks. Such transformation allows DyNN-Offload to create training samples with the knowledge of optimal DyNN partitioning for the pilot model to learn.

4 Evaluation

DyNN-Offload supports a variety of DyNNs and works on real production datasets without the need of refactoring DyNNs. DyNN-Offload significantly improves GPU memory efficiency: given a constraint on GPU memory consumption, DyNN-Offload enables 8× larger DyNN training on a single GPU compared with using PyTorch alone (unprecedented with any existing solution); Evaluating with AlphaFold (a production-level, large-scale DyNN), we show that DyNN-Offload outperforms unified virtual memory (UVM) [18] and dynamic tensor rematerialization (DTR) [12], the most advanced solutions for DyNN, by 3× and 2.1× respectively in terms of maximum batch size. DyNN-Offload also reduces training time of the DyNN by 35% (up to 1.38×) compared to UVM and DTR, while other solutions (e.g., ZeRO-Infinity [22]) cannot work for DyNNs.

References

[1] AlphaFold Performance: Molecule Size, Speed, Memory, and GPU. https://www.rbvi.ucsf.edu/chimerax/data/alphafold-jan2022/afspeed.html.

[2] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves Stoyanov. Efficient large scale language modeling with mixtures of experts, 2022.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*, 2015.

[4] Dehao Chen, Dmitry (Dima) Lepikhin, HyoukJoong Lee, Maxim Krikun, Noam Shazeer, Orhan Firat, Yanping Huang, Yuanzhong Xu, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. 2020.

[5] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

[6] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In *International Conference on Machine Learning*, 2022.

[7] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[8] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *CoRR*, abs/2102.04906, 2021.

[9] Mark Hildebrand, Jawad Khan, Sanjeev Trika, Jason Lowe-Power, and Venkatesh Akella. AutoTM: Automatic Tensor Movement in Heterogeneous Memory Systems Using Integer Linear Programming. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020.

[10] Chien-Chin Huang, Gu Jin, and Jinyang Li. SwapAdvisor: Pushing Deep Learning Beyond the GPU Memory Limit via Smart Swapping. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[11] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[12] Marisa Kirisame, Steven Lyubomirsky, Altan Haan, Jennifer Brennan, Mike He, Jared Roesch, Tianqi Chen, and Zachary Tatlock. Dynamic tensor rematerialization, 2021.

[13] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph LSTM. *CoRR*, abs/1603.07063, 2016.

[14] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime Neural Pruning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[15] Azalia Mirhoseini. Pushing the Limits of Scaling Laws in the Age of Generative Models. Keynote at Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2023.

[16] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *Symposium on Operating System Principles*, 2019.

[17] Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. Memory-Efficient Pipeline-Parallel DNN Training. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[18] Nvidia. Unified Memory. https://devblogs.nvidia.com/unified-memory-in-cuda-6/, 2019.

[19] Xuan Peng, Xuanhua Shi, Hulin Dai, Hai Jin, Weiliang Ma, Qian Xiong, Fan Yang, and Xuehai Qian. Capuchin: Tensor-based GPU Memory Management for Deep Learning. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[20] Bharadwaj Pudipeddi, Maral Mesmakhosroshahi, Jinwen Xi, and Sujeeth Bharadwaj. Training large neural networks with constant memory using a new execution algorithm. *CoRR*, abs/2002.05645, 2020.

[21] 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. *CoRR*, abs/2201.05596, 2022.

[22] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. *CoRR*, abs/2104.07857, 2021.

[23] Jie Ren, Jiaolin Luo, Kai Wu, Minjia Zhang, Hyeran Jeon, and Dong Li. Sentinel: Efficient Tensor Migration and Allocation on Heterogeneous Memory Systems for Deep Learning. In *International Symposium on High Performance Computer Architecture (HPCA)*, 2020.

[24] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *USENIX Annual Technical Conference*, 2021.

[25] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016.

[26] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8583–8595. Curran Associates, Inc., 2021.

[27] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models, 2021.

[28] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[29] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. Mesh-TensorFlow: Deep Learning for Supercomputers. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[30] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations (ICLR)*, 2017.

[31] Liang Shen, Zhihua Wu, WeiBao Gong, Hongxiang Hao, Yangfan Bai, HuaChao Wu, Xinxuan Wu, Jiang Bian, Haoyi Xiong, Dianhai Yu, and Yanjun Ma. Se-moe: A scalable and efficient mixture-of-experts distributed training and inference system, 2023.

[32] Ilya Sutskever, Oriol Vinyals, and Quocviet Le. Sequence to Sequence Learning with Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[33] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks, 2015.

[34] NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation, 2022.

[35] Linnan Wang, Jinmian Ye, Yiyang Zhao, Wei Wu, Ang Li, Shuaiwen Leon Song, Zenglin Xu, and Tim Kraska. Superneurons: Dynamic GPU Memory Management for Training Deep Neural Networks. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2018.

[36] Zhaofeng Wu, Ding Zhao, Qiao Liang, Jiahui Yu, Anmol Gulati, and Ruoming Pang. Dynamic sparsity neural networks for automatic speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[37] Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. Go wider instead of deeper. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8779–8787, 2022.

[38] An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, Di Zhang, Wei Lin, Lin Qu, Jingren Zhou, and Hongxia Yang. M6-t: Exploring sparse expert models and beyond, 2021.

[39] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2019.