Fast and Easy Large-scale Machine Learning via Algorithm and System Co-design

Foundation models (FMs), *e.g.*, GPT-4 and Llama2, are at the forefront of advances in machine learning (ML). These large models undergo pre-training on extensive datasets and are subsequently fine-tuned for specialized tasks, demonstrating proficiency in domains such as natural language and image processing [1]. Nonetheless, the development of FMs demands substantial computational resources and a deep understanding of distributed ML and systems. A case in point is the 70B-parameter Llama2, which required 35 days of training time on 2,048 NVIDIA A100 GPUs [2].

There is thus a pressing need to develop efficient distributed ML algorithms and systems that scale across various computing environments, such as high-performance computing clusters or decentralized commodity hardware. Current distributed ML methods often face hurdles in computation and communication efficiency, hindering ideal scalability, particularly for FMs that involve an immense count of parameters. This situation gives rise to significant challenges at the juncture of ML and systems.

Research summary. I have developed efficient distributed ML and federated learning (FL) algorithms and systems that deliver tangible computation and communication speedups in distributed clusters, *e.g.*, those on Amazon EC2. In addition to these advancements, I have explored the robustness of distributed ML and FL algorithms against malicious compute nodes and hardware failure. My research has led to several industry adoptions. My contributions can be classified into three categories, as follows:

1. **Computation and communication-efficient training by low-rank approximation.** I developed ATOMO, the first low-rank gradient factorization method for communication-efficient training [3]. My subsequent work fully bypassed gradient compression by directly training low-rank models [4, 5]. These frameworks, seamlessly compatible with existing ML systems like PyTorch, offer significant improvements in computation and communication efficiency without compromising model quality. **The aforementioned approaches have been adopted by SONY to accelerate their internal model training for product development.** My low-rank methods have also inspired subsequent research, such as the Low-Rank Adaptation (LoRA) parameter-efficient fine-tuning (PEFT) method, which essentially employs the same low-rank approach and draws partial inspiration from my frameworks [6].

2. **Hybrid-parallel distributed ML using system cost models.** It is challenging but necessary to design optimal hybrid-parallel distributed ML strategies, *e.g.*, composing data, tensor-model, and pipeline parallelism strategies for optimal training speed. The problem is more challenging on distributed clusters with various types of GPUs. I have developed pioneering system cost models to guide users to design hybrid-parallel ML strategies for any ML model and any distributed environment [7, 8, 9].

3. **Efficient and trustworthy federated learning.** I have led the development of a few efficient FL algorithms [10, 11]. Among these, the most notable is FedMA, an FL algorithm that enjoys significantly faster convergence due to a novel *matched averaging* model aggregation scheme (**ICLR 2020 Oral; 809 citations to date; FedMA has been adopted by IBM**) [10]. I also conducted extensive research on the resilience of FL against malicious clients, which has closed debates in the field [12]. Furthermore, my research has enhanced the computational efficiency of private FM inferences using secure multi-party computation (**ICLR 2023 Spotlight**) [13].

1 Accomplished Works on Large-scale Machine Learning

1.1 Computation and Communication-efficient Distributed Machine Learning

Distributed ML speeds up training but also incurs substantial communication costs for synchronizing gradients or activations. My research bridges system studies and optimizations as well as efficient algorithm designs, addressing four core research questions (RQs).

RQ 1: Communication-efficient distributed learning by gradient compression. In data-parallel training, gradients are communicated. We showed that many gradient compression approaches can be viewed as facets of a general scheme, which dictates a specific atomic decomposition. This led to the development of ATOMO, a general compression framework, and established experimentally that using low-rank gradients instead of sparse ones, can lead to significantly faster distributed training [3]. Since then, low-rank gradients have been extensively explored, and current open-source libraries for communication-efficient training implement variants of these ideas. ATOMO, however, comes with a fixed compression ratio, requiring balancing accuracy and per-iteration speedup. We demonstrated that such a trade-off is not fundamental, and adaptive compression can help. We developed an adaptive algorithm that mitigates the detrimental impact of over-compression during critical learning regimes [14]. Large-scale experiments show that

this provides accuracy comparable to uncompressed training, and a $4\times$ end-to-end speedup over static compression methods. Beyond data parallelism, for model parallel training, we introduced a learning-based method for activation compression as gradient compression is not directly applicable.

RQ 2: Utility of gradient compression. Gradient compression has significant potential for enhancing communication efficiency in data-parallel distributed ML [3]. However, its benefits are primarily observed in distributed clusters with limited communication bandwidth, while in clusters with sufficient bandwidth, it may hinder the speed of distributed training. These observed inconsistencies prompted our comprehensive evaluation of the effectiveness of all popular gradient compression methods across hundreds of real distributed setups. In typical data center setups with high communication bandwidths, compression does not always yield significant speedups [7]. To quantitatively analyze this phenomenon, we developed fine-grained cost models to evaluate the benefits of gradient compression methods across various distributed system setups [7, 9]. Our findings identify the specific conditions under which gradient compression contributes to significant speedups and guide the design of efficient future compression methods.

RQ 3: Communication-efficient distributed training with no extra cost. We developed the PUFFERFISH framework that bypasses the cost of compression by training low-rank models [4]. Directly training low-rank models leads to accuracy loss, yet we avoid this by borrowing ideas from the Lottery Ticket Hypothesis literature [15, 16]. We showed that accurate low-rank models can be obtained by training a full-rank model, and then converting to low-rank early in training. PUFFERFISH also retains full-rank for layers sensitive to low-rank factorization. Nonetheless, these techniques necessitate extra hyperparameter tuning, such as determining the optimal transition time from full-rank to low-rank and pinpointing sensitive layers. Addressing this, we designed CUTTLEFISH to automatically estimate and adjust these hyperparameters during training [5]. Using low-rank training, we managed to achieve almost linear scalability in data-parallel training (Figure 1).

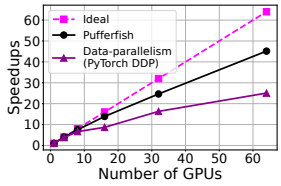

Figure 1: Scalability of low-rank training on ResNet-50 trained on ImageNet-1k dataset upto 64 GPUs.

Using the aforementioned low-rank training approach, we also pre-trained a low-rank version of Llama2 (the base model with 1.3 billion parameters), which enabled the removal of 41% of redundant parameters, resulting in a $1.4\times$ end-to-end training speedup (as shown in Table 1). **Our low-rank frameworks have been adopted by SONY for product development.**

	# Params. (B)	Valid. Loss	Days
Llama2	$1.3_{(1\times)}$	2.25	$10.33_{(1\times)}$
Low-rank Llama2	$0.79_{(0.59\times)}$	2.32	$7.57_{(1.37\times)}$

Table 1: Performance of low-rank LLAMA2 pre-training on the Pile-OpenWebText2 corpus using our methods.

RQ 4: Hybrid-parallel distributed ML on clusters with various types of GPUs. It has been shown that neither data, tensor-model, nor pipeline parallelism alone can achieve optimal FM pre-training speed. Designing hybrid-parallel ML strategies, however, often demands extensive tuning efforts. One challenge is determining how to optimally assign, for instance, one thousand GPUs along data, tensor-model, and pipeline parallelism dimensions. It becomes even more challenging for heterogeneous clusters, *i.e.,* those with various types of GPUs. To simplify this, we developed the AMP framework that automatically tunes the hybrid-parallelism strategy [8]. We developed precise cost models and an efficient search algorithm to pinpoint the best parallelism strategy. AMP performs equally well compared to human expert-tuned strategies in typical distributed clusters. Notably, on heterogeneous clusters, AMP identified strategies that achieved 1.8 times end-to-end speedup over NVIDIA's leading Megatron-LM system for GPT pre-training.

1.2 Efficient and Trustworthy Federated Learning

FL is important for learning on decentralized and private data. My research studied efficient and trustworthy FL methods, which tackles the following two RQs.

RQ 5: Better federated model aggregations. We developed the FedMA algorithm that constructs the shared global model in a layer-wise manner by matching and averaging hidden elements with similar feature extraction signatures. FedMA not only outperforms popular FL algorithms but also reduces communication costs significantly. **FedMA has been adopted by IBM.**

RQ 6: The resilience of FL against malicious clients. My research indicates that providing robustness guarantees against malicious clients in FL is highly challenging, if not impossible [12]. Specifically, we demonstrate that robustness in FL generally leads to model robustness against adversarial examples, a major open problem, while detecting a malicious FL client within a polynomial time frame is unlikely. We have also developed a novel category of training-time attacks, called *edge-case backdoors*. These backdoors, difficult to detect, are readily inserted into federated models. Edge-case backdoors cause the models to misclassify seemingly straightforward inputs that are unlikely to appear in

the training or test data. We demonstrate that skilled adversaries can exploit this backdoor across various ML tasks.

2 Future Research: Easier Foundation Model Developments and Deployments

My ultimate research goal is to **empower everyone with the capacity to develop and deploy their own FMs on personal hardware**. This involves optimizing the use of existing hardware and jointly developing efficient algorithms and systems. Such efforts are intended to bridge the gap between the constrained computing resources accessible to academic research labs and the substantial resources demanded by FMs. Specifically, I propose the following aims:

Aim 1: Efficient Distributed Training by Optimally Leveraging Existing Hardware

Modern AI accelerators are often upgraded every two years (*e.g.*, NVIDIA's V100 in 2018; A100 in 2020; H100 in 2022). However, older hardware often falls into disuse. This raises the question: "*How can we effectively reuse older hardware generations, integrating them with current versions or utilizing them independently?*"

I plan to investigate two research aspects. First, can a group of older GPUs, if optimally parallelized, match or even exceed the latest GPU's performance? Secondly, how to build a distributed training system for optimal throughput on clusters with GPUs from various generations.

The theoretical teraflops of four RTX 2080 Ti GPUs from 2017 are comparable to those of a single A6000 GPU from 2021, assuming ideal scaling. For the first aspect, we could co-optimize system and algorithm designs for parallelization while maintaining model quality. Distributed training on mixed-generation GPU clusters faces challenges like uneven GPU performance and memory capacities. For the second aspect, I propose the development of distributed training compilers that consider the specifics of FM pre-training tasks and the particularities of hardware across different generations. Additionally, I plan to deploy quantization and sparsification algorithms to address the bandwidth heterogeneity among hardware of various generations.

Aim 2: System-friendly Algorithms for Pre-training Foundation Models

I am a deep believer that FMs will drive major breakthroughs in AI. However, the computational requirements they entail, *e.g.*, thousands of GPUs, surpass what academia can offer. A solution could be to deploy efficient algorithms for optimal hardware use. I have noticed many inefficiencies in FM pre-training. My goal is to reduce computational redundancy in FMs' parameters, activations, and key-value caches, and to echo this with system enhancements.

I have used low-rank approximation to trim redundant model parameters [4, 5]. Yet, it is unlikely that every model's parameter weights are low-rank. Theory suggests that matrices often have both low-rank and sparse elements. With this in mind, I intend to unify low-rank and sparsity concepts. I aim to investigate a system-friendly N:M sparsity and find the best methods to arrange model weights to maximize the advantages of low-rank approximations.

On the system front, I intend to implement efficient CUDA kernels to support sparsity and low-rank structures. For instance, the low-rank model weights can be executed using a fused GPU kernel to boost IO and memory efficiency, which is similar to the core idea of FlashAttention [17].

Aim 3: Efficient Fine-tuning and Inference for Foundation Models

FMs are often fine-tuned for specific applications, including code generation and fundamental science, before their deployment for inference tasks. However, both these stages are resource-intensive. I aim to develop automatic and efficient algorithms and systems to speed up FM's fine-tuning and inference.

For the fine-tuning phase, while PEFT techniques such as LoRA show potential, they often require extensive hyperparameter tuning [6]. For instance, determining the specific model layers where LoRA should be applied and fine-tuning the rank of the LoRA adapters are critical for achieving high accuracy. I intend to develop AutoPEFT, a system designed to alleviate the need for such extensive tuning while maximizing resources like GPU memory. I also propose deploying AutoPEFT in FL applications where clients conduct federated training using devices with heterogeneous resources (*e.g.*, computing power and memory capacity).

For efficient inference without compromising the model's generation quality, I intend to merge model approximation with speculative decoding (SD). SD primarily uses a smaller *draft model* to generate, with a larger *target model* for occasional verification. The key challenge lies in designing a compact draft model that closely mimics the target one. I propose a holistic algorithmic framework that unifies techniques like low-rank, sparsification, quantization, and knowledge distillation. I also plan to study how to jointly fine-tune the target and draft models for varied applications.

I am excited to identify and address the challenges in the development and deployment of FM via comprehensive solutions combining algorithms and systems inspired by real-world application requirements and constraints.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[3] **Hongyi Wang***, Scott Sievert*, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. Atomo: Communication-efficient learning via atomic sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] **Hongyi Wang**, Saurabh Agarwal, and Dimitris Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems (MLSys)*, 3, 2021.

[5] **Hongyi Wang**, Saurabh Agarwal, Yoshiki Tanaka, Eric Xing, Dimitris Papailiopoulos, et al. Cuttlefish: Low-rank model training without all the tuning. *Proceedings of Machine Learning and Systems (MLSys)*, 5, 2023.

[6] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022.

[7] Saurabh Agarwal, **Hongyi Wang**, Shivaram Venkataraman, and Dimitris Papailiopoulos. On the utility of gradient compression in distributed training systems. *Proceedings of Machine Learning and Systems (MLSys)*, 2022.

[8] Dacheng Li, **Hongyi Wang**, Eric Xing, and Hao Zhang. AMP: Automatically finding model parallel strategies with heterogeneity awareness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[9] Song Bian, Dacheng Li, **Hongyi Wang**, Eric P Xing, and Shivaram Venkataraman. Does compressing activations help model parallel training? *arXiv preprint arXiv:2301.02654*, 2023.

[10] **Hongyi Wang**, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *International Conference on Learning Representations (ICLR,* **Oral***)*, 2020.

[11] Junbo Li, Ang Li, Chong Tian, Qirong Ho, Eric Xing, and **Hongyi Wang**. Fednar: Federated optimization with normalized annealing regularization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[12] **Hongyi Wang**, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] Dacheng Li*, Rulin Shao*, **Hongyi Wang***, Han Guo, Eric Xing, and Hao Zhang. Mpcformer: fast, performant and private transformer inference with mpc. *The Eleventh International Conference on Learning Representations (***Spolight***)*, 2023.

[14] Saurabh Agarwal, **Hongyi Wang**, Kangwook Lee, Shivaram Venkataraman, and Dimitris Papailiopoulos. Adaptive gradient communication via critical learning regime identification. *Proceedings of Machine Learning and Systems (MLSys)*, 2021.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew Grinde, Alliot Nagle, **Hongyi Wang**, Eric Xing, Kangwook Lee, and Dimitris Papailiopoulos. Rare gems: Finding lottery tickets at initialization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.