OmniScale: Scaling Any Modality Model Training with Model-Centric Distributed Recipe Zoo

Qianli Ma[1*†] Yaowei Zheng[1*], Zhelun Shi[1*], Zhongkai Zhao[1*], Bin Jia[1*], Ziyue Huang[1*], Zhiqi Lin[1], Youjie Li[1], Jiacheng Yang[1], Yanghua Peng[1†], Zhi Zhang[1†], Xin Liu[1†]

[1]ByteDance Seed

fazziekey@gmail.com, {shizhelun, zhangzhi.joshua, pengyanghua.yanghua, liuxin.ai}@bytedance.com

Abstract

Recent advances in large language models (LLMs) have driven impressive progress in omni-modal understanding and generation. However, training omni-modal LLMs remains a significant challenge due to the heterogeneous model architectures required to process diverse modalities, necessitating sophisticated system design for efficient large-scale training. Existing frameworks typically entangle model definition with parallel logic, incurring limited scalability and substantial engineering overhead for end-to-end omni-modal training. We present OmniScale, a modular and efficient training framework to accelerate the development of omni-modal LLMs. OmniScale introduces model-centric distributed recipes that decouples communication from computation, enabling efficient 3D parallelism on omni-modal LLMs. OmniScale also features a flexible configuration interface supporting seamless integration of new modalities with minimal code change. Using OmniScale, a omni-modal mixture-of-experts (MoE) model with 30B parameters can be trained with over 2,800 tokens/sec/GPU throughput and scale to 160K context lengths via 3D parallelism on 128 GPUs, showcasing its superior efficiency and scalability for training large omni-modal LLMs.

Code — https://github.com/ByteDance-Seed/VeOmni

Introduction

The evolution of large language models (LLMs) has progressed from unimodal specialization towards unified omni-modal understanding and generation (Sun et al. 2024c; Liu et al. 2025; Wu et al. 2025). Recent models such as GPT-4o (Hurst et al. 2024) and BAGEL (Deng et al. 2025a) exhibit strong performance across diverse multimodal tasks. These tasks include visual question answering (Alayrac et al. 2022; Guo et al. 2023), controllable image generation (Zhang et al. 2023; Li et al. 2025), and multimodal reasoning (Wang et al. 2024c; Su et al. 2025; Huang et al. 2025b), highlighting the growing potential of LLMs as general-purpose omni-modal agents.

As LLMs are extended to handle diverse modalities, their architectures have become increasingly heterogeneous and complicated. State-of-the-art omni-modal LLMs typically incorporate multiple modality-specific pre-trained networks to process inputs with fundamentally different properties, such as continuous signals (*e.g.*, images, audio) and discrete sequences (*e.g.*, text). In these omni-modal models, a language model often serves as the central backbone (Vaswani et al. 2017; Brown et al. 2020), connecting with vision encoders (Dosovitskiy et al. 2021), audio encoders (Dong, Xu, and Xu 2018), and image generation networks (Ho, Jain, and Abbeel 2020; Goodfellow et al. 2020) through learned bridging mechanisms.

Despite these advances, the development of omni-modal LLMs still lags behind their unimodal counterparts. This gap is largely attributed to the absence of scalable training frameworks tailored for omni-modal tasks. While numerous mature and widely adopted systems exist for training language models on text-to-text tasks (Shoeybi et al. 2019; Narayanan et al. 2019, 2021; Li et al. 2023; Shen et al. 2024; Liang et al. 2025; veScale Team 2024), few frameworks are designed to support any-to-text tasks (Huang et al. 2024a; Zhang et al. 2024b; Ji et al. 2024; Feng et al. 2025). Crucially, none of these frameworks enable end-to-end training for fully omni-modal scenarios, *i.e.*, any-to-any learning, revealing the necessity for scalable infrastructure to build the next generation of omni-modal LLMs.

Extending existing training frameworks to omni-modal LLMs is non-trivial. To address the growing parameter sizes, modern training frameworks leverage various distributed training strategies (Shoeybi et al. 2019; Narayanan et al. 2019; Li et al. 2020; Narayanan et al. 2021) to alleviate computation and memory bottlenecks. For example, Megatron-LM (Shoeybi et al. 2019; Narayanan et al. 2021) provides optimized transformer blocks (Vaswani et al. 2017) with advanced parallelization strategies like tensor parallelism (TP) and pipeline parallelism (PP). However, as omni-modal architectures become complex in both functionality and parameter size, directly applying these techniques to omni-modal LLMs often leads to load imbalance and poor scalability (Feng et al. 2025), due to the current frameworks tightly couple model definition with parallel logic. This en-

[*]These authors contributed equally.

[†]Corresponding author

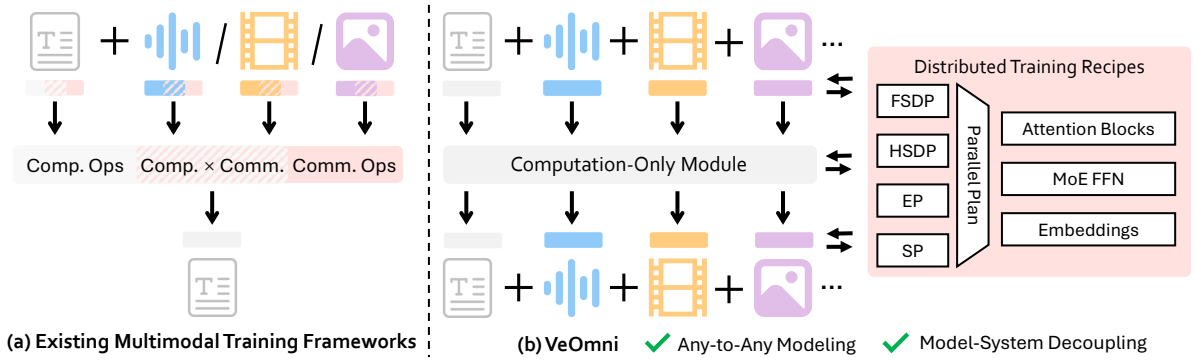

Figure 1: **Comparison between OmniScale and Existing Training Frameworks.**

tanglement hiders generalization to more diverse model architectures. Although recent efforts (Huang et al. 2024a; Zhang et al. 2024b; Feng et al. 2025) attempt to mitigate the inefficiencies of omni-modal training, they still suffer from the coupling of communication and computation, resulting in significant engineering cost and limited extensibility.

To efficiently train large omni-modal LLMs, OmniScale introduces model-centric distributed recipes that decouple model definition from parallel logic. As shown in Figure 1, distributed strategies such as fully sharded data parallel (FSDP), hybrid sharded data parallel (HSDP) (Zhao et al. 2023; Liang et al. 2025), sequence parallelism (SP) (Jacobs et al. 2023), and expert parallelism (EP) (Zhang et al. 2025) can be applied to model blocks via a high-level *parallel plan* API. This design effectively separates communication from computation and has been validated in recent LLM training systems (Liang et al. 2025). OmniScale supports flexible composition of parallel strategies (*e.g.*, FSDP+SP for 2D and FSDP+SP+EP for 3D parallelism), enabling a range of training recipes tailored to dense or MoE models. By decoupling parallel logic, new modality-specific modules can be integrated with minimal engineering effort, as computation modules need not account for distributed concerns. In addition, OmniScale provides a lightweight interface for customizing omni-modal LLMs, allowing users to easily add or remove modality-specific encoders and decoders.

Our contributions are as follows:

(1) We propose model-centric distributed recipes that efficiently scale omni-modal training using n-D parallelism with minimal engineering overhead.

(2) We introduce a light-weight configuration interface for easy customization of omni-modal LLMs.

(3) We demonstrate OmniScale's competitive efficiency and scalability across 8–128 GPUs on models ranging from 7B to 72B parameters under omni-modal scenarios.

Related Work

Multi-Modal and Omni-Modal LLMs

Recent advances in large language models (LLMs) have increasingly focused on developing multi-modal and omni-modal capabilities. The prevailing approaches include incorporating modality-specific encoders into the input space of the LLMs for multimodal understanding (Liu et al. 2023; Wang et al. 2024b; Yin et al. 2023; Lin et al. 2023), and attaching generative decoders to the output space for multimodal generation (Huang et al. 2024b; Tian et al. 2025) or embodied action prediction (Black et al. 2024; Kim et al. 2024). More recently, omni-modal LLMs that support unified understanding and generation across modalities have emerged, aiming to align arbitrary modality features with language in a shared latent space. Based on a unified paradigm of auto-regressive modeling over multimodal embeddings, these models differ significantly in how the modality features are encoded and decoded. Some (Team 2024; Wu et al. 2025) adopt discrete-token generation pipelines based on VQ-VAE series (Esser, Rombach, and Ommer 2020; Razavi, Van den Oord, and Vinyals 2019). Some (Wang et al. 2024a; Huang et al. 2025a; Sun et al. 2024b) employ continuous-token generation via latent diffusion models (Rombach et al. 2022; Podell et al. 2023). Others (Deng et al. 2025b; Yang et al. 2025; Zhou et al. 2024) explore hybrid architectures that combine diffusion-based generation with auto-regressive decoding. In addition, some works proposed alternative next-token prediction schemes, such as masked generation (Li et al. 2024), next patch prediction (Pang et al. 2024), next scale prediction (Tian et al. 2024), next block prediction (Ren et al. 2025), offering increased flexibility beyond standard auto-regressive decoding. Given this diverse landscape of model architectures, OmniScale offers a flexible and extensible framework for building and scaling new modeling paradigms.

LLM Training Frameworks

The landscape of large language model training frameworks has evolved significantly to address the computational demands of scaling. For pure text training, several mature frameworks have established themselves as industry standards. Megatron-LM (Shoeybi et al. 2019; Narayanan et al. 2021) pioneered optimized transformer blocks with advanced parallelization strategies including tensor parallelism (TP) and pipeline parallelism (PP), becoming the foundation for many subsequent frameworks. Colossal-AI (Li et al. 2023) extends this paradigm by offering comprehensive 3D parallel training strategies that combine data, tensor, and pipeline parallelism for enhanced scalability. NeMo (Shen et al. 2024) provides an end-to-end cloud solu-

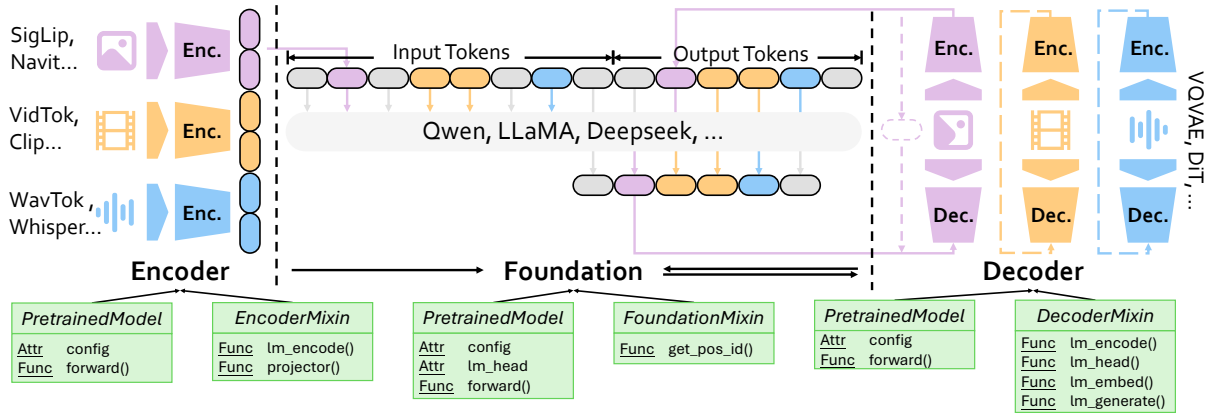

Figure 2: **Composite Architecture for Omni-Modal LLMs**. The architecture consists of three fully decoupled modules: Encoder, Foundation, and Decoder.

tion specifically designed for training large-scale LLMs with enterprise-grade features, while TorchTitan (Liang et al. 2025) and veScale (veScale Team 2024) represent the newer generation of PyTorch-native frameworks that emphasize auto-parallelization and simplified programming models.

In the multi-modal domain, specialized frameworks have emerged to handle the unique challenges of any-to-text training. DistMM (Huang et al. 2024a) introduces optimizations specifically tailored for multimodal LLM training, while DistTrain (Zhang et al. 2024b) addresses model and data heterogeneity through disaggregated training approaches. Align Anything (Ji et al. 2024) focuses on cross-modality model training with feedback mechanisms, and Optimus (Feng et al. 2025) accelerates large-scale multi-modal LLM training through bubble exploitation techniques. However, these existing frameworks primarily target either text-to-text or any-to-text scenarios, leaving a significant gap in supporting comprehensive omni-modal training (any-to-any) scenarios. Our framework addresses this limitation by providing a scalable and modular solution specifically designed for the complexities of omni-modal training, enabling seamless integration of diverse modalities within a unified training pipeline.

OmniScale: Scalable Omni-Modal Training

OmniScale designs a model-centric framework that is natively suitable for training omni-modal models. This framework includes a diverse range of distributed training strategies, enabling any modality to be easily scaled up for large-scale clusters.

Light-Weight Omni-Modal Customization

To support training across arbitrary modalities, we propose a plug-and-play architecture that allows any combination of multimodal encoders and decoders to be flexibly attached to the input and output sides of a foundation model, as shown in Fig. 2. This modular and fully decoupled design decouples system-level operations from model-specific computation, allowing the streaming pipeline to operate independently of the model pipeline. Therefore, OmniScale enables easy extension and scaling of diverse omni-modal LLMs.

Our core design centers around a lightweight protocol tailored for each component. Specifically, encoder and decoder modules implement a unified interface by extending the `PreTrainedModel` class from HuggingFace along with a task-specific mixin. This design ensures compatibility with standard training workflows and enables all modules to be registered, initialized, and executed consistently.

During training, for input modalities, each encoder implements an `lm_encode` function that encodes raw modality data into token embeddings, which are then inserted into the input embeddings of the foundation model. Similarly, decoder modules implement the `lm_encode` function to provide training-time inputs representing the target output tokens. The final outputs from the foundation model are decoded into the target modality using `lm_head` function, completing the end-to-end mapping.

In the inference phase, for each prediction step, the output hidden states is passed through the corresponding modality decoder's `lm_embed` function, which generates embeddings that are used as inputs for the foundation model's next token prediction. Once all tokens have been predicted, the decoder's `lm_generate` function is invoked to generate the final modality-specific output.

Model-Centric Distributed Recipe Zoo

To support large-scale distributed training of omni-modal LLMs, OmniScale offers a modular suite of distributed training strategies, including Fully Sharded Data Parallelism (FSDP), Sequence Parallelism (SP), and Expert Parallelism (EP), designed to handle challenges such as training with ultra-long sequences and efficiently scaling large mixture-of-experts (MoE) models. OmniScale enables users to freely compose distributed training recipes tailored to their model architectures and modality-specific requirements, without needing to manage low-level implementation details. The framework is designed with the following principles:

1. ***Non-intrusive distributed training APIs.*** OmniScale decouples the model architecture from the underlying dis-

tributed training mechanisms, enabling users to compose multimodal models flexibly without modifying low-level parallelization code.

2. ***Support for long context training.*** OmniScale accommodates training workloads involving extremely long sequences across different modalities, which is a common challenge in omni-modal training.
3. ***Efficient scaling for MoE models.*** OmniScale facilitates efficient training of large-scale MoE models through integrated expert parallelism, achieving both scalability and compute efficiency.

In the following sections, we introduce each of these distributed technologies in OmniScale and demonstrate how they are integrated to support scalable omni-modal training.

Fully and Hybrid Sharded Data Parallel Fully Sharded Data Parallel (FSDP) is a highly efficient implementation of the Zero Redundancy Optimizer (ZeRO-3) (Rajbhandari et al. 2020) in PyTorch. Its primary advantage is the significant reduction in memory required on each GPU during training. By sharding a model's parameters, gradients, and optimizer states across all available devices, FSDP allows for the training of extremely large models that would otherwise exceed the memory capacity of a single GPU. A key advantage of FSDP is its non-intrusive design, which decouples the model's architecture from the parallelization strategy. This means developers can focus on designing their models without needing to modify the underlying code to accommodate the distributed training setup. This non-intrusive nature makes FSDP exceptionally well-suited for training omni-modal LLMs. Since these models are defined by their complex and heterogeneous architectures, FSDP's ability to parallelize training without requiring any changes to the model's code makes it an ideal solution. OmniScale integrates both FSDP1 (Zhao et al. 2023) and FSDP2 (Liang et al. 2025) as fundamental components of its distributed training stack, and provides a unified API for easy configuration.

To further improve training efficiency, OmniScale integrates Hybrid Sharded Data Parallel (HSDP), an extension of FSDP designed to minimize communication overhead. HSDP utilizes a 2D device mesh, combining FSDP within "shard groups" and Distributed Data Parallel (DDP) across "replicate groups". This hybrid approach drastically cuts down on inter-node communication, enabling even greater scalability. The switch to the more efficient HSDP is as simple as changing one parameter in the configuration, with no modifications to the underlying code required.

Sequence Parallelism for Long Context Training With the advancement of state-of-the-art omni-modal LLMs, the sequence lengths required for training have grown significantly, particularly in domains such as high-resolution image or video understanding and generation (Wang et al. 2024b; Zhang et al. 2024a; Gao et al. 2025). This trend toward longer sequences poses substantial challenges in terms of both computational cost and memory consumption. Efficiently addressing the demands of long context training is therefore critical for scaling model capacity and unleashing the full potential of omni-modal architectures. To address this, OmniScale adopts DeepSpeed Ulysses (Jacobs et al. 2023), a highly efficient sequence parallelism technique specifically designed for transformer training on ultra-long sequences. DeepSpeed Ulysses splits activations along the sequence dimension and strategically orchestrates all-to-all communications during attention computation, ensuring that communication volume remains constant when both sequence length and device count are scaled proportionally. This design enables high efficiency and scalability for long-context model training. A core advantage of OmniScale is its provision of a non-intrusive interface for deploying Ulysses. Developers can seamlessly leverage DeepSpeed Ulysses without needing to modify model-building code or directly interact with the underlying implementation details. In particular, OmniScale enhances the implementation of Flash Attention (Dao et al. 2022) by integrating DeepSpeed Ulysses while keeping fully compatible with the default attention, enabling the straightforward adoption of advanced attention acceleration methods within sequence parallel workflows.

Furthermore, to optimize training throughput, we introduce Async-Ulysses, an enhanced implementation designed to overlap communication and computation. This version schedules the all-to-all communication operations to execute concurrently with the linear projection computations. By effectively hiding a significant portion of the communication latency behind computation, this approach yields substantial improvements in training performance.

Expert Parallelism for MoE Model Scaling Mixture-of-Experts (MoE) architectures have emerged as a mainstream solution for scaling large models efficiently (DeepSeek-AI 2025), particularly due to their ability to sparsely activate subsets of parameters during inference and training, enabling significant improvements in both computational efficiency and model capacity. In the context of omni-modal foundation models, adopting MoE architectures as the backbone not only reduces training cost but also enhances model performance across diverse modalities by dynamically allocating expert capacity based on input content.

To support the scalable training of MoE-based omni-modal LLMs, OmniScale introduces a user-friendly and flexible interface for expert parallelism, allowing users to easily apply expert sharding across devices without manual configuration. This interface is compatible with widely used MoE variants and supports plug-and-play integration, thereby significantly lowering the barrier to implementing distributed expert parallelism.

A major bottleneck in large-scale MoE training lies in all-to-all communication required for routing tokens to their assigned experts, introducing substantial latency. To mitigate this, OmniScale incorporates fine-grained communication-computation overlapping techniques inspired by recent advances (Zhang et al. 2025; Chang et al. 2024). These works hide communication latency by scheduling collective operations concurrently with local expert computation, eliminating the need for complex pipeline-level solutions such

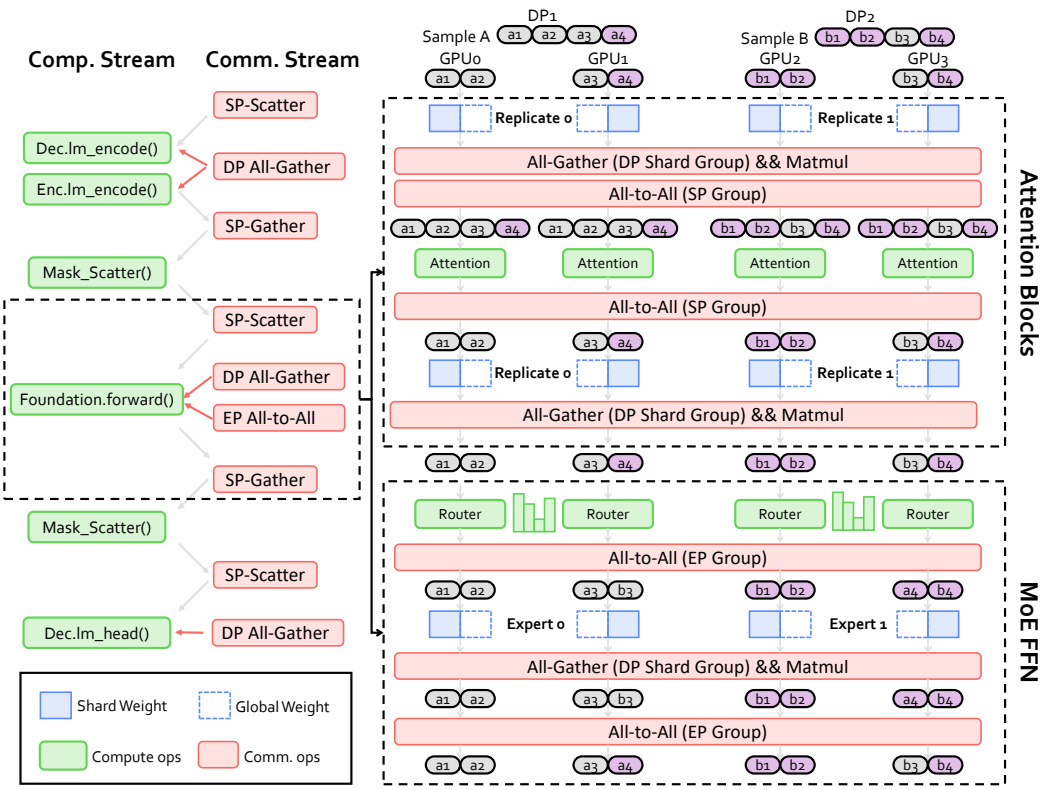

Figure 3: **Overview of OmniScale's n-D Parallelism Design.** The left figure illustrates the data flow of OmniScale during the training of omni-modal LLMs. The encoder for each modality processes its respective input, and then scatters the features to the corresponding ranks through an all-to-all communication operation. The right figure shows the 3D parallelism system of OmniScale. Attention blocks apply HSDP with data sharded size 2 and data replicate size 2, input apply Ulysses with sequence parallel size 2, Mixture-of-Experts blocks apply expert parallel and FSDP with experts parallel size 2 and data sharded size 2.

as DualPipe (DeepSeek-AI 2025). Unlike pipeline-centric designs, which are often rigid and sensitive to modality-specific imbalance, OmniScale's operator-level approach is lightweight, model-agnostic, and particularly well-suited to multi-modal settings. This results in higher utilization and faster iteration during large-scale MoE training.

n-D Parallelism Training Strategies In OmniScale, core parallelism strategies (*i.e.*, Fully Sharded Data Parallel (FSDP), Sequence Parallelism (SP), and Expert Parallelism (EP)) are designed to be fully composable and can be flexibly applied to different components of an omni-modal LLM. This composability is particularly advantageous for multimodal training, where different modalities or architectural components may benefit from distinct parallelism techniques. For example, vision encoders can adopt FSDP or standard data parallelism depending on their scale, while language backbones can leverage a combination of EP for MoE layers and SP to support long-context processing. This fine-grained flexibility enables efficient and scalable training across diverse model architectures.

Figure 3 provides an overview of OmniScale's n-D parallelism design. The left figure illustrates the communication and computation flow for a unified multimodal model, where different modality-specific encoders process their respective inputs and distribute intermediate features to downstream modules. The right side visualizes the application of hybrid parallelism strategies across different parts of the model, demonstrating how various parallelism techniques coexist within a single training configuration. To support this flexible parallelism composition, OmniScale introduces a unified abstraction for distributed training based on a global device mesh. Unlike approaches (Shoeybi et al. 2019; Narayanan et al. 2021) that require manually managing multiple process groups, OmniScale significantly simplifies the configuration and coordination of n-D parallelism. This not only reduces the complexity of process group management but also improves extensibility, making it easier to adapt and extend to future parallelism strategies or new model components.

Other System Optimization Strategies In addition to the aforementioned parallelism strategies, OmniScale also incorporates a wide range of other system-level optimizations, following the key design principle of OmniScale, which is the decoupling of these optimization implementations from the model's core logic. This modular architecture allows for seamless integration, enabling these system-level enhancements to be readily applied across various models with minimal to no modification of the model code. These optimizations include, but are not limited to, the following:

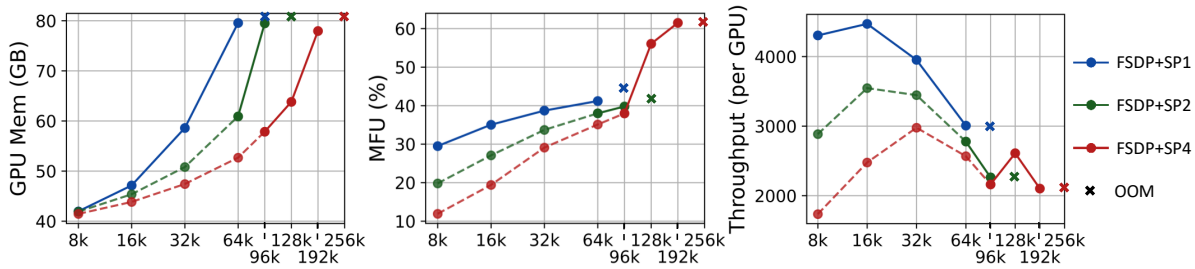

Figure 4: **2D Parallelism (FSDP+SP) on Qwen2-VL 7B with 8 GPUs.** The maximum sequence length varies from 8K to 256K tokens, with supported sequence parallel sizes ranging from 1 to 4.

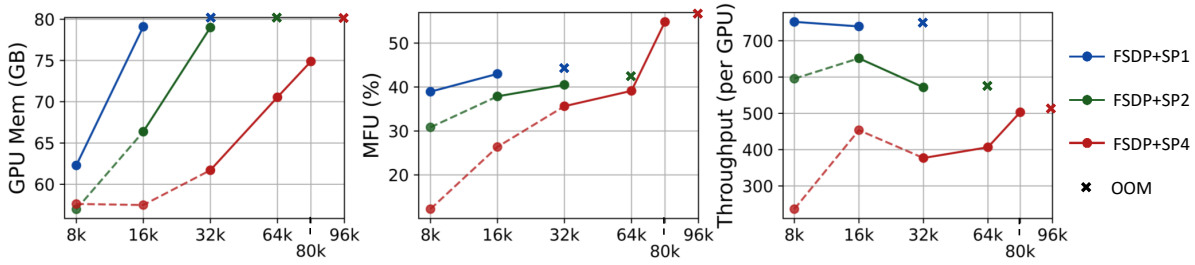

Figure 5: **2D Parallelism (FSDP+SP) on Qwen2-VL 72B with 128 GPUs.** The maximum sequence length varies from 8K to 96K tokens, with supported sequence parallel sizes ranging from 1 to 8.

Dynamic Batching. Padding all samples in a batch to a fixed sequence length often leads to substantial computational inefficiency, particularly when there is significant variation in sequence lengths across samples. To mitigate this issue and improve training efficiency, OmniScale employs Dynamic Batching that accumulates samples in a buffer and strategically packs them to approximate a target sequence length. Leveraging FlashAttention (Dao 2023), this packing strategy enables efficient utilization of the batch budget with minimal padding overhead, while preserving the correctness of attention computation across samples.

Efficient Kernels. To maximize training throughput, OmniScale incorporates a suite of highly optimized operator kernels, including RMSNorm, LayerNorm, RoPE, SwiGLU, and in-place CrossEntropy from Liger-kernel (Hsu et al. 2025), along with FlashAttention (Dao et al. 2022; Dao 2023; Shah et al. 2024) and MoE-specific operations (Chang et al. 2024; Zhang et al. 2025). These kernels are carefully engineered for both high performance and broad compatibility, enabling efficient execution across diverse transformer-based architectures and model variants.

Memory Optimization. OmniScale incorporates layer-wise recomputation (Chen et al. 2016), activation offloading, and optimizer state offloading to substantially reduce memory consumption during training. These memory-saving techniques enable the use of larger micro-batch sizes per GPU, which in turn improves the amortization of communication costs and facilitates better overlap with the communication overhead introduced by Fully Sharded Data Parallel (FSDP), ultimately enhancing overall training efficiency.

Efficient Distributed Checkpointing. OmniScale leverages ByteCheckpoint (Wan et al. 2024) to enable efficient checkpoint saving and resumption across varying distributed configurations with minimal overhead. Beyond facilitating elastic training and enhancing cluster utilization, our framework further extends ByteCheckpoint to support multimodal models. This extension ensures consistent and reliable saving and loading of heterogeneous model components.

Meta Device Initialization. OmniScale supports model initialization on the meta device for large models without allocating physical memory during the definition phase. After the model is instantiated on the meta device, we perform parameter sharding and parallel loading by converting parameters into the DTensor format, significantly accelerating the initialization and loading processes for large models.

Experiments

Experimental Setup

Environments. We evaluate the training performance and scalability of OmniScale on large-scale productive GPU clusters across configurations ranging from 8 to 128 GPUs, enabling a comprehensive study of OmniScale's behavior under both moderate and large-scale distributed settings.

Models and Datasets. We evaluate a diverse set of model architectures, including dense models such as Qwen2-VL 7B and 72B (Wang et al. 2024b), and a mixture-of-experts (MoE) omni-modal LLM based on Qwen3-MoE (Team 2025). To assess the multimodal capabilities of OmniScale, we use the following domain-specific datasets: FineWeb-100T (Penedo et al. 2024) for text understanding, ShareGPT4V (Chen et al. 2024) for image understanding, LLaVA-Video (Zhang et al. 2024a) for video understanding, Voice Assistant (Xie and Wu 2024) for audio understanding,

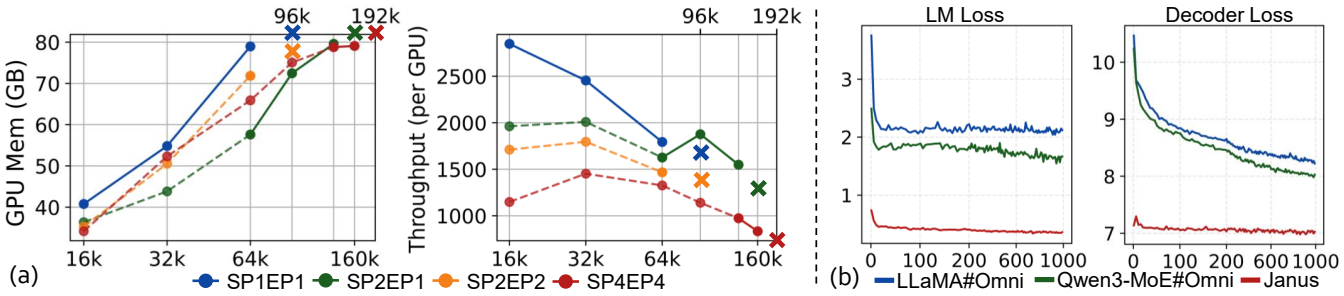

Figure 6: (a) 3D Parallelism (FSDP+SP+EP) on a 30B Qwen3MoE-based omni-modal LLM with 128 GPUs. The maximum sequence length varies from 16K to 192K tokens, with supported sequence parallel sizes ranging from 1 to 4 and expert parallel sizes ranging from 1 to 4. (b) Convergence Study on three Distinct omni-modal LLMs.

and ImageNet (Deng et al. 2009) for image generation tasks. Each model is adapted using its native instruction template and augmented with special tokens to delineate modality boundaries (*e.g.* `<image_start>` and `<image_end>`).

Workloads and Metrics. To assess scalability across different model sizes and GPU counts, we progressively scale the input context length from 8K to 256K tokens. We evaluate the performance and scalability of OmniScale by measuring training throughput (tokens per second per GPU) and model FLOPs utilization (MFU) (Chowdhery et al. 2023), which together provide a comprehensive view of system efficiency under varying workloads and parallel strategies. Additionally, we analyze loss convergence behavior across three structurally distinct omni-modal LLMs to validate training stability and overall effectiveness. For all experiments, we freeze the modality-specific encoders and decoders, while fully fine-tuning the remaining components, including the foundation model and multimodal projectors.

Training Recipes under Different Scenarios

Figures 4–6 (a) present a comprehensive comparison of the efficiency of various parallelism strategies—fully sharded data parallel (FSDP), sequence parallelism (SP), and expert parallelism (EP)—across diverse model scales and hardware configurations. Figures 4-5 show results of training Qwen2-VL on 8 and 128 GPUs, respectively. The findings reveal that increasing the degree of sequence parallelism enables the models to handle significantly longer context lengths. In particular, OmniScale can support up to 192K tokens for training the 7B model with an MFU of 61.5% and 96K tokens for training the 72B model with an MFU of 54.82%. Figure 6 further extends this analysis to a 3D parallelism configuration (FSDP + SP + EP) using a 30B parameter a mixture-of-experts (MoE) omni-modal LLM based on Qwen3-MoE, on 128 GPUs. In this setting, combinations involving moderate levels of SP and EP achieve a balanced trade-off, supporting extended sequence lengths of up to 160K tokens, while maintaining competitive throughput.

The experimental results underscore the efficiency of OmniScale in handling long-sequence training and scaling mixture-of-experts (MoE) models. Specifically, OmniScale demonstrates strong performance by minimizing overhead in short-context scenarios, while effectively leveraging sequence and expert parallelism to maintain scalability and feasibility in extended-context and MoE settings.

Convergence Study on Omni-Modal LLMs

Figure 6 (b) presents the convergence behavior of three omni-modal LLMs on multimodal understanding and generation tasks. The understanding tasks span four modalities, *i.e.*, text, image, video, and audio, while the generation tasks involve text and image synthesis. Janus (Wu et al. 2025) is trained solely on image understanding and image generation. LLaMA#Omni and Qwen3-Moe#Omni are two custom models that share similar architectures: Qwen2.5-Omni (Xu et al. 2025) ViT serves as the image and video encoder, Qwen2.5-Omni Whisper as the audio encoder, and MoVQGAN (Zheng et al. 2022) as the image decoder. LLaMA#Omni uses LLaMA (Touvron et al. 2023) as the foundation, while Qwen3-Moe#Omni adopts Qwen3-MoE (Team 2025). Janus adopts SigLip (Zhai et al. 2023) as the image encoder, LLaMA as the foundation model, and LlamaGen (Sun et al. 2024a) as the image decoder. "LM Loss" refers to the cross-entropy loss over text tokens, and "Decoder Loss" refers to the cross-entropy loss over image tokens. As shown in Figure 6 (b), all models exhibit stable convergence across both understanding and generation tasks, demonstrating that OmniScale enables efficient and robust training for large omni-modal LLMs.

Conclusion

We introduce OmniScale, a model-centric distributed training framework designed to scale any-modality models efficiently. By integrating multiple parallelism methods and system optimizations into a composable recipe-based design, OmniScale enables seamless and efficient distributed training strategies across omni-modal LLMs. We further demonstrate the system-level optimizations, modular APIs, and real-world training applications on large-scale vision-language-audio models. Experimental analysis shows that OmniScale not only achieves high throughput and scalability but also provides developer-friendly abstractions for fast prototyping and production-scale deployment. In future work, we plan to extend OmniScale to support non-intrusive pipeline parallelism, enabling further decoupling of model definition and parallel execution. Additionally, we aim to enhance sequence parallelism with modality-aware data balancing strategies to better support multimodal training.

References

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35: 23716–23736.

Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; Jakubczak, S.; Jones, T.; Ke, L.; Levine, S.; Li-Bell, A.; Mothukuri, M.; Nair, S.; Pertsch, K.; Shi, L. X.; Tanner, J.; Vuong, Q.; Walling, A.; Wang, H.; and Zhilinsky, U. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chang, L.-W.; Bao, W.; Hou, Q.; Jiang, C.; Zheng, N.; Zhong, Y.; Zhang, X.; Song, Z.; Yao, C.; Jiang, Z.; Lin, H.; Jin, X.; and Liu, X. 2024. FLUX: Fast Software-based Communication Overlap On GPUs Through Kernel Fusion. arXiv:2406.06858.

Chen, L.; Li, J.; Dong, X.; Zhang, P.; He, C.; Wang, J.; Zhao, F.; and Lin, D. 2024. Sharegpt4v: Improving large multimodal models with better captions. In *European Conference on Computer Vision*, 370–387. Springer.

Chen, T.; Xu, B.; Zhang, C.; and Guestrin, C. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Dao, T. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems*, 35: 16344–16359.

DeepSeek-AI. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437.

Deng, C.; Zhu, D.; Li, K.; Gou, C.; Li, F.; Wang, Z.; Zhong, S.; Yu, W.; Nie, X.; Song, Z.; Shi, G.; and Fan, H. 2025a. Emerging Properties in Unified Multimodal Pretraining. *arXiv preprint arXiv:2505.14683*.

Deng, C.; Zhu, D.; Li, K.; Gou, C.; Li, F.; Wang, Z.; Zhong, S.; Yu, W.; Nie, X.; Song, Z.; et al. 2025b. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dong, L.; Xu, S.; and Xu, B. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5884–5888. IEEE.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Esser, P.; Rombach, R.; and Ommer, B. 2020. Taming Transformers for High-Resolution Image Synthesis. arXiv:2012.09841.

Feng, W.; Chen, Y.; Wang, S.; Peng, Y.; Lin, H.; and Yu, M. 2025. Optimus: Accelerating {Large-Scale}{Multi-Modal}{LLM} Training by Bubble Exploitation. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, 161–177.

Gao, Y.; Guo, H.; Hoang, T.; Huang, W.; Jiang, L.; Kong, F.; Li, H.; Li, J.; Li, L.; Li, X.; et al. 2025. Seedance 1.0: Exploring the Boundaries of Video Generation Models. *arXiv preprint arXiv:2506.09113*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Guo, J.; Li, J.; Li, D.; Tiong, A. M. H.; Li, B.; Tao, D.; and Hoi, S. 2023. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10867–10877.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hsu, P.-L.; Dai, Y.; Kothapalli, V.; Song, Q.; Tang, S.; Zhu, S.; Shimizu, S.; Sahni, S.; Ning, H.; and Chen, Y. 2025. Liger Kernel: Efficient Triton Kernels for LLM Training. arXiv:2410.10989.

Huang, J.; Zhang, Z.; Zheng, S.; Qin, F.; and Wang, Y. 2024a. DISTMM: accelerating distributed multimodal model training. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation*, 1157–1171.

Huang, R.; Wang, C.; Yang, J.; Lu, G.; Yuan, Y.; Han, J.; Hou, L.; Zhang, W.; Hong, L.; Zhao, H.; et al. 2025a. Illume+: Illuminating unified mllm with dual visual tokenization and diffusion refinement. *arXiv preprint arXiv:2504.01934*.

Huang, W.; Jia, B.; Zhai, Z.; Cao, S.; Ye, Z.; Zhao, F.; Xu, Z.; Hu, Y.; and Lin, S. 2025b. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Huang, Y.; Xie, L.; Wang, X.; Yuan, Z.; Cun, X.; Ge, Y.; Zhou, J.; Dong, C.; Huang, R.; Zhang, R.; et al. 2024b. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8362–8371.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jacobs, S. A.; Tanaka, M.; Zhang, C.; Zhang, M.; Song, S. L.; Rajbhandari, S.; and He, Y. 2023. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*.

Ji, J.; Zhou, J.; Lou, H.; Chen, B.; Hong, D.; Wang, X.; Chen, W.; Wang, K.; Pan, R.; Li, J.; et al. 2024. Align anything: Training all-modality models to follow instructions with language feedback. *arXiv preprint arXiv:2412.15838*.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; Vuong, Q.; Kollar, T.; Burchfiel, B.; Tedrake, R.; Sadigh, D.; Levine, S.; Liang, P.; and Finn, C. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246.

Li, S.; Liu, H.; Bian, Z.; Fang, J.; Huang, H.; Liu, Y.; Wang, B.; and You, Y. 2023. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, 766–775.

Li, S.; Zhao, Y.; Varma, R.; Salpekar, O.; Noordhuis, P.; Li, T.; Paszke, A.; Smith, J.; Vaughan, B.; Damania, P.; et al. 2020. PyTorch distributed: experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12): 3005–3018.

Li, T.; Tian, Y.; Li, H.; Deng, M.; and He, K. 2024. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37: 56424–56445.

Li, Z.; Cheng, T.; Chen, S.; Sun, P.; Shen, H.; Ran, L.; Chen, X.; Liu, W.; and Wang, X. 2025. ControlAR: Controllable Image Generation with Autoregressive Models. In *The Thirteenth International Conference on Learning Representations*.

Liang, W.; Liu, T.; Wright, L.; Constable, W.; Gu, A.; Huang, C.-C.; Zhang, I.; Feng, W.; Huang, H.; Wang, J.; et al. 2025. TorchTitan: One-stop PyTorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*.

Lin, B.; Zhu, B.; Ye, Y.; Ning, M.; Jin, P.; and Yuan, L. 2023. Video-LLaVA: Learning United Visual Representation by Alignment Before Projection. *arXiv preprint arXiv:2311.10122*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 34892–34916. Curran Associates, Inc.

Liu, H.; Yan, W.; Zaharia, M.; and Abbeel, P. 2025. World Model on Million-Length Video And Language With Blockwise RingAttention. In *The Thirteenth International Conference on Learning Representations*.

Narayanan, D.; Harlap, A.; Phanishayee, A.; Seshadri, V.; Devanur, N. R.; Ganger, G. R.; Gibbons, P. B.; and Zaharia, M. 2019. PipeDream: Generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM symposium on operating systems principles*, 1–15.

Narayanan, D.; Shoeybi, M.; Casper, J.; LeGresley, P.; Patwary, M.; Korthikanti, V.; Vainbrand, D.; Kashinkunti, P.; Bernauer, J.; Catanzaro, B.; et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 1–15.

Pang, Y.; Jin, P.; Yang, S.; Lin, B.; Zhu, B.; Tang, Z.; Chen, L.; Tay, F. E.; Lim, S.-N.; Yang, H.; et al. 2024. Next patch prediction for autoregressive visual generation. *arXiv preprint arXiv:2412.15321*.

Penedo, G.; Kydlíček, H.; allal, L. B.; Lozhkov, A.; Mitchell, M.; Raffel, C.; Werra, L. V.; and Wolf, T. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Rajbhandari, S.; Rasley, J.; Ruwase, O.; and He, Y. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–16. IEEE.

Razavi, A.; Van den Oord, A.; and Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

Ren, S.; Ma, S.; Sun, X.; and Wei, F. 2025. Next block prediction: Video generation via semi-autoregressive modeling. *arXiv preprint arXiv:2502.07737*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Shah, J.; Bikshandi, G.; Zhang, Y.; Thakkar, V.; Ramani, P.; and Dao, T. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37: 68658–68685.

Shen, G.; Wang, Z.; Delalleau, O.; Zeng, J.; Dong, Y.; Egert, D.; Sun, S.; Zhang, J. J.; Jain, S.; Taghibakhshi, A.; et al. 2024. NeMo-Aligner: Scalable Toolkit for Efficient Model Alignment. In *First Conference on Language Modeling*.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Su, Z.; Xia, P.; Guo, H.; Liu, Z.; Ma, Y.; Qu, X.; Liu, J.; Li, Y.; Zeng, K.; Yang, Z.; et al. 2025. Thinking with Images for Multimodal Reasoning: Foundations, Methods, and Future Frontiers. *arXiv preprint arXiv:2506.23918*.

Sun, P.; Jiang, Y.; Chen, S.; Zhang, S.; Peng, B.; Luo, P.; and Yuan, Z. 2024a. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*.

Sun, Q.; Cui, Y.; Zhang, X.; Zhang, F.; Yu, Q.; Wang, Y.; Rao, Y.; Liu, J.; Huang, T.; and Wang, X. 2024b. Generative multimodal models are in-context learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14398–14409.

Sun, Q.; Yu, Q.; Cui, Y.; Zhang, F.; Zhang, X.; Wang, Y.; Gao, H.; Liu, J.; Huang, T.; and Wang, X. 2024c. Emu: Generative Pretraining in Multimodality. In *The Twelfth International Conference on Learning Representations*.

Team, C. 2024. Chameleon: Mixed-Modal Early-Fusion Foundation Models. *arXiv preprint arXiv:2405.09818*.

Team, Q. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Tian, K.; Jiang, Y.; Yuan, Z.; Peng, B.; and Wang, L. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37: 84839–84865.

Tian, X.; Li, W.; Xu, B.; Yuan, Y.; Wang, Y.; and Shen, H. 2025. MIGE: A Unified Framework for Multimodal Instruction-Based Image Generation and Editing. arXiv:2502.21291.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

veScale Team. 2024. veScale: Towards PyTorch-Native Auto-Parallel Framework. https://github.com/volcengine/veScale.

Wan, B.; Han, M.; Sheng, Y.; Peng, Y.; Lin, H.; Zhang, M.; Lai, Z.; Yu, M.; Zhang, J.; Song, Z.; et al. 2024. ByteCheckpoint: A Unified Checkpointing System for Large Foundation Model Development. *arXiv preprint arXiv:2407.20143*.

Wang, C.; Lu, G.; Yang, J.; Huang, R.; Han, J.; Hou, L.; Zhang, W.; and Xu, H. 2024a. Illume: Illuminating your llms to see, draw, and self-enhance. *arXiv preprint arXiv:2412.06673*.

Wang, P.; Bai, S.; Tan, S.; Wang, S.; Fan, Z.; Bai, J.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Fan, Y.; Dang, K.; Du, M.; Ren, X.; Men, R.; Liu, D.; Zhou, C.; Zhou, J.; and Lin, J. 2024b. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191*.

Wang, Y.; Chen, W.; Han, X.; Lin, X.; Zhao, H.; Liu, Y.; Zhai, B.; Yuan, J.; You, Q.; and Yang, H. 2024c. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. *arXiv preprint arXiv:2401.06805*.

Wu, C.; Chen, X.; Wu, Z.; Ma, Y.; Liu, X.; Pan, Z.; Liu, W.; Xie, Z.; Yu, X.; Ruan, C.; et al. 2025. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 12966–12977.

Xie, Z.; and Wu, C. 2024. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*.

Xu, J.; Guo, Z.; He, J.; Hu, H.; He, T.; Bai, S.; Chen, K.; Wang, J.; Fan, Y.; Dang, K.; et al. 2025. Qwen2.5-Omni Technical Report. *arXiv preprint arXiv:2503.20215*.

Yang, L.; Tian, Y.; Li, B.; Zhang, X.; Shen, K.; Tong, Y.; and Wang, M. 2025. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*.

Yin, Z.; Wang, J.; Cao, J.; Shi, Z.; Liu, D.; Li, M.; Huang, X.; Wang, Z.; Sheng, L.; BAI, L.; Shao, J.; and Ouyang, W. 2023. LAMM: Language-Assisted Multi-Modal Instruction-Tuning Dataset, Framework, and Benchmark. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 26650–26685. Curran Associates, Inc.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, 11975–11986.

Zhang, S.; Zheng, N.; Lin, H.; Jiang, Z.; Bao, W.; Jiang, C.; Hou, Q.; Cui, W.; Zheng, S.; Chang, L.-W.; Chen, Q.; and Liu, X. 2025. Comet: Fine-grained Computation-communication Overlapping for Mixture-of-Experts. arXiv:2502.19811.

Zhang, T.; Zhang, Y.; Vineet, V.; Joshi, N.; and Wang, X. 2023. Controllable text-to-image generation with gpt-4. *arXiv preprint arXiv:2305.18583*.

Zhang, Y.; Wu, J.; Li, W.; Li, B.; Ma, Z.; Liu, Z.; and Li, C. 2024a. Video Instruction Tuning With Synthetic Data. arXiv:2410.02713.

Zhang, Z.; Zhong, Y.; Ming, R.; Hu, H.; Sun, J.; Ge, Z.; Zhu, Y.; and Jin, X. 2024b. Disttrain: Addressing model and data heterogeneity with disaggregated training for multimodal large language models. *arXiv preprint arXiv:2408.04275*.

Zhao, Y.; Gu, A.; Varma, R.; Luo, L.; Huang, C.-C.; Xu, M.; Wright, L.; Shojanazeri, H.; Ott, M.; Shleifer, S.; et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

Zheng, C.; Vuong, T.-L.; Cai, J.; and Phung, D. 2022. Movq: Modulating quantized vectors for high-fidelity image generation. *Advances in Neural Information Processing Systems*, 35: 23412–23425.

Zhou, C.; Yu, L.; Babu, A.; Tirumala, K.; Yasunaga, M.; Shamis, L.; Kahn, J.; Ma, X.; Zettlemoyer, L.; and Levy, O. 2024. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*.