ISBN: 978-981-94-4198-3
2025 the 15th International Workshop on Computer Science and Engineering (WCSE 2025)
doi: 10.18178/wcse.2025.06.004

Towards Efficient Multimodal AI: A Flexible Framework for Distributed Training

WeiYu Chen[1,2], Yen Han Chiang[1,2+] and Vincent S. Tseng[2]

[1] NCHC (National Center for High-Performance Computing)

[2] NYCU (National Yang Ming Chiao Tung University)

Abstract. The rapid advancement of artificial intelligence has led to an increasing demand for systems capable of processing and understanding multimodal information. However, developing such systems poses challenges in integrating data, architecture, and computational resources. To address these issues, this study proposes a flexible and efficient distributed multimodal training and model deployment framework designed to streamline the development process. The framework facilitates seamless interaction between key components, optimizing resource allocation and training efficiency. To evaluate the effectiveness of the proposed framework, we conduct experiments on a multimodal anomaly detection task, demonstrating its ability to handle complex data representations. Our study also explores distributed training strategies to enhance the scalability of large-scale multimodal models. Through performance analysis, we identify and mitigate key bottlenecks, leading to improved computational efficiency and reduced training time. Experimental results confirm that our framework not only supports efficient multimodal learning but also enhances system performance in large-scale AI applications.

Keywords: MultiModal Learning, Anomaly Detection, Performance Optimization

1. Introduction

Traditional information processing methods primarily relied on unimodal approaches, where data from a single source (such as text, images, or audio) was analyzed in isolation. However, real-world scenarios often require understanding and integrating multiple data types, as different modalities provide complementary information. For instance, in autonomous driving, sensor fusion enables a more comprehensive understanding of the environment. Multimodal learning addresses these limitations by leveraging heterogeneous data sources to improve accuracy, robustness, and generalizability. Despite its advantages, multimodal learning presents significant challenges, such as modality alignment, data fusion, and increased computational complexity.

The integration of multiple modalities offers several advantages. First, it allows for richer feature extraction, leading to improved performance in tasks like anomaly detection and medical diagnosis. For example, combining medical imaging with patient history enhances diagnostic accuracy. Second, multimodal learning facilitates deeper insights into complex phenomena, such as human behavior analysis, by incorporating textual, visual, and auditory cues. Finally, it fosters interdisciplinary research and innovation, enabling applications across domains like healthcare, security, and entertainment. However, these benefits come at the cost of increased system complexity, requiring sophisticated frameworks for data alignment, model fusion, and computational efficiency.

To address the challenges of multimodal learning, we propose a flexible and efficient distributed training framework that integrates multiple key components. Our system also supports various distributed training strategies, including data parallelism, model parallelism, and hybrid approaches such as 3D parallelism, with Zero Redundancy Optimizer (ZeRO) reducing memory redundancy and improving scalability.

To validate our proposed framework, we conduct experiments on a multimodal anomaly detection task using datasets such as MVTec-AD and ViSA. Our system distributes training tasks across multiple nodes, leveraging advanced parallelism techniques to optimize training efficiency. Performance analysis with NVIDIA Nsight tools helps identify and mitigate computational bottlenecks.

+ Corresponding author. Tel.: + 886-3-5776085; fax: +886-3-5776082.
E-mail address: .2203003@narlabs.org.tw

Our work makes several key contributions to collectively advance the field of multimodal AI by offering a robust, efficient, and scalable solution for training and deploying large-scale models.

- Introduce a novel multimodal training framework that integrates advanced distributed training techniques, lowering the entry barrier for multimodal AI research.
- Design a systematic algorithm to describe interactions among system components, simplifying the complexity of multimodal learning.
- Provide empirical validation of our framework's scalability and efficiency through anomaly detection experiments, demonstrating its ability to handle diverse multimodal data.

2. Related Works

2.1. Multimodal Fusion and Alignment

Multimodal fusion integrates data from different modalities (e.g., images, audio, text) to enhance model performance. Current methods employ deep learning architectures such as CNNs [1] and RNNs [2] to combine multimodal information effectively. Another approach maps embeddings from different modalities into a shared space for better integration [3]. Recently, attention mechanisms have been used to dynamically highlight relevant features, improving fusion results [4].

Multimodal alignment is crucial for precise multimodal representations. Techniques such as CLIP [5] learn joint embedding spaces for images and text. Advanced models like Flamingo [6], BLIP-2 [7], and MAGIC [8] leverage pre-trained vision-language models for strong zero-shot performance. AudioCLIP [9] extends CLIP by incorporating audio, while ImageBind [10] aligns six modalities using image-paired data. Additionally, models such as LLaVa [11] and Mini-GPT4 [12] enable vision-based instruction following, DetGPT [13] enhances object detection reasoning, and SpeechGPT [14] integrates speech understanding in language models.

2.2. LVLM for Anomaly Detection

Recent advancements in Large Vision-Language Models (LVLMs) have significantly impacted anomaly detection across various domains. AnomalyGPT [15] addresses industrial anomaly detection by generating simulated anomalous images paired with textual descriptions, eliminating manual threshold settings and supporting multi-turn dialogues. VisionGPT [16] enhances real-time anomaly detection in visual navigation by identifying obstacles in camera frames and providing audio descriptions, utilizing open-vocabulary object detection models and specialized prompts. SOWA [17] improves performance by adapting hierarchical frozen window self-attention mechanisms to process multi-level features in visual-language models. VL4AD [18] integrates vision-language encoders into existing detectors, leveraging broad pre-training and a novel scoring function for data- and training-free outlier supervision via textual prompts.

2.3. System Acceleration Strategies

Efficient training of Large Language Models (LLMs) requires advanced system acceleration strategies to manage substantial computational demands. Distributed training partitions tasks across multiple computing nodes, enhancing efficiency and handling larger datasets [26]. This approach includes data parallelism, where the model is replicated across nodes processing different data subsets, and model parallelism, which divides the model among nodes to train models exceeding a single device's memory capacity [27]. Pipeline parallelism segments the model into stages, allowing concurrent processing of data batches, while 3D parallelism combines data, model, and pipeline strategies for optimal resource utilization [28]. Mixed-precision training employs both 16-bit and 32-bit floating-point representations, reducing memory usage and accelerating computation without compromising accuracy [29]. Adaptive optimizers like Adam and LAMB dynamically adjust learning rates, facilitating faster convergence [30]. Collectively, these strategies aim to enhance training efficiency, reduce computational costs, and maximize hardware resource utilization without sacrificing model performance [31].

3. System Architecture

3.1. System Component

Fig. 1 illustrates the framework of this study to depict the software framework as below components:

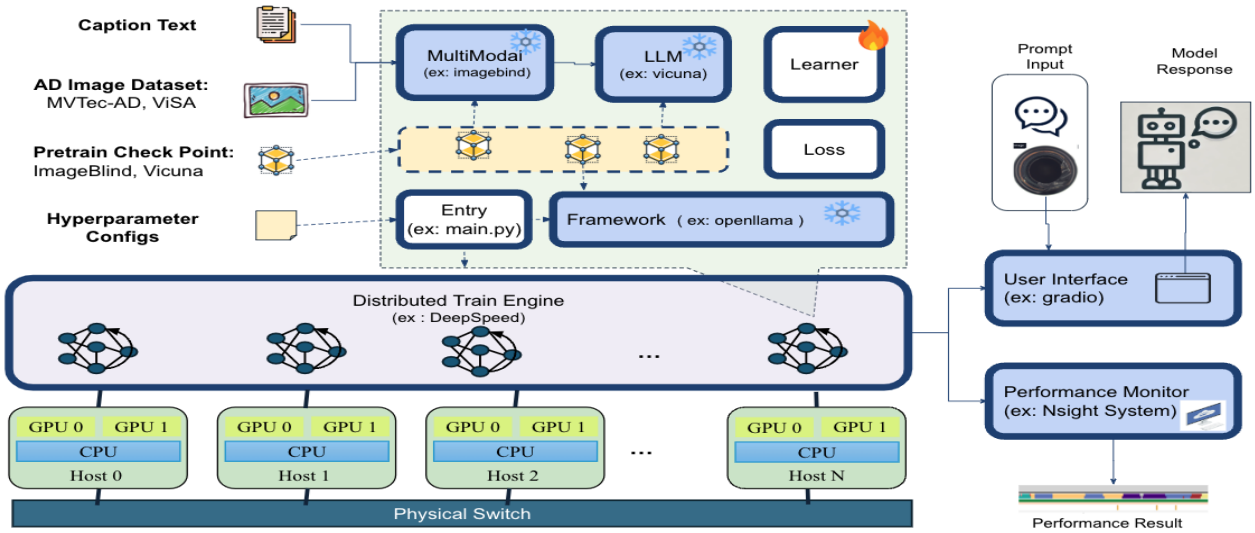

Fig. 1: System Framework.

- Input :
  - Multi-modal supports various types of data, including "Caption Text" for textual information, "AD Image Dataset" for anomaly detection dataset, Pretrain model for fine tuning, and HyperParameter for system configuration.
- Software :
  - Entry: (main.py): main.py initializes the environment, parses arguments, imports configurations, and starts the training process with model and optimizer setup.
  - Framework: (OpenLLaMA) : OpenLLaMA integrates and processes diverse multi-modal data, offering a flexible architecture for rapid multi-modal application development.
  - Multi-modal: (ImageBind) : ImageBind efficiently integrates and processes multi-modal data, with tools for feature extraction and model optimization in image processing.
  - LLM: (Vicuna) : Vicuna, based on LLaMA2, offers language understanding and generation for multi-language text processing and applications like customer service.
  - Learner: Loss function : The learner algorithm iterates to minimize prediction errors; we use Focal Loss to address class imbalance in training.
- Hardware Communication:
  - The hardware architecture in this study employs multiple nodes, each containing several GPUs, as computational nodes. These nodes exchange data through high-speed switches for communication between nodes.
- Distributed Train Engine: (DeepSpeed):
  - This study leverages DeepSpeed's ZeRO stage to optimize memory efficiency by distributing model weights, optimizer states, and gradients across all nodes and GPUs. During backpropagation, each GPU stores only a portion of the gradients, which are later aggregated using All-Reduce operations to ensure proper parameter updates. This approach enhances memory utilization, allowing larger models to be trained on the same hardware resources.
- Output
  - User Interface: (Gradio) : Gradio enables easy, interactive UI for machine learning models, supporting real-time input/output for images, text, and audio.
  - Monitoring: (NVIDIA Nsight) : This tool is used for node-level monitoring, providing in-depth performance analysis of the system and applications. It offers a detailed timeline view, showing the operational status and interdependencies of computation units, helping identify key factors that impact performance.

3.2. Method

This study employs Algorithm to outline the detailed workflow, illustrating the step-by-step process and methodology followed throughout the experiment. The algorithm provides a clear representation of the operations involved, ensuring that the execution flow is well-defined and easily understandable for further

analysis and implementation.

Algorithm 1	**Descriptions:**
• *Input: Configs & Data_Img, Data_Text* • *Pretrain : P_{llm}, P_{mm}, P_{model}* • *Output: Trained_Multi_Modal_Model* • *Component: TrainEngine(), Entry(), Framework(), MultiModal(), LLM(), LossLearner()* • *Algorithm:* ***Entry():*** *1. conf ← parse(Configs)* *2. Agent ← initialize(TrainEngine(conf))* *3. For each GPU per node:* *4. call TrainEngine():* ***TrainEngine ()*** *1. initialize({ LLM()←P_{llm}, MultiModal()←P_{mm}, Framework()←P_{model}})* *2. Training Loop for each round:* *3. For each (Batch_Img, Batch_Text) in data_loader(Data_Img , Data_Text):* *4. outputs ← Framework (LLM(MultiModal(Batch_Img , Batch_Text)))* *5. loss ← LossLearner(outputs, targets)* *6. back_propagate()* *7. weights, optimizer ← update()* *8. save(Trained_Multi_Modal_Model)*	The algorithm processes configuration files (Configs), image data (Data_Img), and text data (Data_Text) to train a multimodal model. It utilizes pre-trained models: a language model (Pllm), a multimodal model (Pmm), and a framework model (Pmodel) to enhance performance and stability. The output is a Trained_Multi_Modal_Model capable of understanding image and text data.Key components include TrainEngine, Entry, Framework, MultiModal, LLM, and LossLearner. The process starts at Entry(), implemented in main.py, where Configs are parsed, and a training agent (Agent) initializes TrainEngine. This function is executed across multiple GPU nodes. Inside TrainEngine(), core models (LLM, MultiModal, Framework) are initialized from their pre-trained versions (Pllm, Pmm, Pmodel). The training loop iterates over data batches, processing images and text through MultiModal and LLM, then passing results to Framework. The LossLearner computes losses, followed by backpropagation to update weights and optimize parameters. The loop continues until performance stabilizes, and the final trained model is saved (Trained_Multi_Modal_Model).

Fig. 2: Workflow Representation Based on Algorithm

4. Experiments

4.1. Environment Setup

The experimental setup in this study utilizes a distributed computing cluster comprising four identical nodes, each featuring a 24-core Intel® Xeon® Gold 6336Y CPU (2.40 GHz) and two NVIDIA A30 GPUs with 24 GB of memory per GPU. These nodes are interconnected via a Mellanox NVLink switch, offering a bandwidth of 200 GB/sec to ensure high-speed data transfer. Within this cluster, one node serves as the Master Node, while the remaining three function as Worker Nodes, forming a distributed training architecture. The Master Node oversees task coordination, including distributing workloads, aggregating results, and synchronizing model gradients and parameters. It also actively participates in forward propagation, backpropagation, and gradient calculations. The Worker Nodes handle computational tasks and return their results to the Master Node for integration. To optimize memory usage and enhance training efficiency, the system employs DeepSpeed with ZeRO Stage-2, which distributes model parameters and optimizer states across all nodes and GPUs. The cluster operates on PyTorch 1.13.1 and DeepSpeed 0.9.3 frameworks, with password-free SSH enabling seamless communication between nodes. Training tasks are initiated by the Master Node, which assigns parallel computations to Worker Nodes. Preloading all required data and scripts onto each node ensures consistency across the cluster. Resource utilization during training is monitored using NVIDIA Nsight. This scalable system was used to evaluate the performance of various vision models in anomaly detection tasks. Models were trained on the VISA dataset under this distributed setup and subsequently tested on the MVTecAD dataset to assess their generalization capabilities across related datasets.

4.2. Experiment Results

The experiments were conducted using the VISA dataset for training and MVTecAD for testing to evaluate anomaly detection performance. VISA contains categorized object images, while MVTecAD includes labeled anomalies for industrial applications. The study compared four models: ResNet, Fully Convolutional Network (FCN), Multi-Layer Perceptron (MLP), and MLP with Batch Normalization (BatchNorm). Results showed that FCN (90.94%) and MLP (91.09%) outperformed ResNet (90.49%), while

BatchNorm had minimal impact (90.93%). Category-wise, ResNet and FCN excelled in “bottle” and “hazelnut” (>96%), whereas MLP struggled with “screw” and “transistor” (<80%). FCN emerged as the most consistent model across categories.

	Precision (visa->MVtecAD)			
	Resnet	FCN	MLP	BN
bottle	96.39	95.18	95.18	95.18
cable	84.67	88.00	85.33	84.67
capsule	78.79	79.55	80.30	80.30
carpet	100.00	99.15	100.00	100.00
grid	97.44	97.44	96.15	98.72
hazelnut	96.36	97.27	95.45	95.45
leather	98.39	98.39	98.39	98.39
metal_nut	99.13	98.26	100.00	99.13
pill	85.63	86.83	86.83	86.83
screw	75.63	76.25	76.88	77.50
tile	98.29	97.44	97.44	98.29
toothbrush	95.24	97.62	100.00	97.62
transistor	79.00	79.00	80.00	78.00
wood	93.67	93.67	93.67	93.67
zipper	78.81	80.13	80.79	80.13
avg	90.49	90.94	**91.09**	90.93

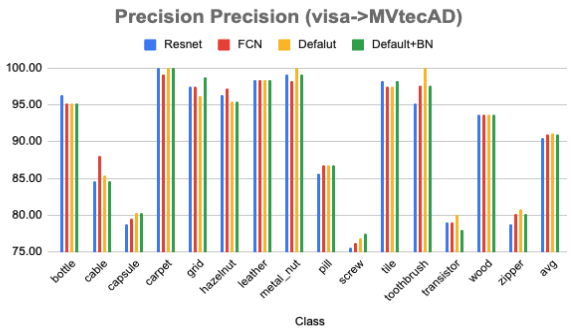

Fig. 3: Experiment Results

In this experiment, as shown in Fig. 4, a detailed performance analysis of the DeepSpeed training process was conducted using the NVIDIA Nsight tool. During the training of the multimodal model, a total of 600 seconds was captured. The performance analysis view displays the threads in DeepSpeed, where different colored blocks represent various types of threads and their durations. The DeepSpeed process primarily performs computational operations, such as forward and backward propagation. Black blocks indicate these computational operations, while purple, green, and yellow blocks represent communication and synchronization-heavy operations.

Notably, from 59 seconds to 476 seconds, a significant portion of the time is spent in the “waitpid” state, where the process is waiting for the completion of other processes. This delay is caused by communication latency or resource contention among multiple worker nodes. The performance analysis view further reveals the operation status distribution of threads across different time intervals, from 114 seconds to 536 seconds. When zooming in on this section, the red blocks indicate data waiting times, and the green blocks represent data transfer. Data transmission and synchronization occur between nodes, which corresponds to the waiting time observed in DeepSpeed.

Fig. 4: Performance Monitor

5. Conclusions

Multimodal frameworks have demonstrated remarkable performance across various domains. However, due to their high computational demands and complex architectures, they pose a significant entry barrier for general researchers. This study adopts a multimodal framework and utilizes anomaly detection as a case study to demonstrate its feasibility. In the future, we plan to integrate additional applications, such as real-world image retrieval, to further validate the framework’s effectiveness.

6. Acknowledgements

We thank to National Center for High-performance Computing (NCHC) for providing computational and storage resources.

7. References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, May 2015.

[2] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[3] T. Mikolov et al., “Efficient estimation of word representations in vector space,” in Proc. ICLR, 2013.

[4] A. Vaswani et al., “Attention is all you need,” in Proc. NeurIPS, 2017.

[5] A. Radford et al., “Learning transferable visual models from natural language supervision,” in Proc. ICML, 2021.

[6] J. Alayrac et al., “Flamingo: a visual language model for few-shot learning,” in Proc. NeurIPS, 2022.

[7] J. Li et al., “BLIP-2: Bootstrapped language-image pre-training with frozen image encoders and large language models,” in Proc. ICML, 2023.

[8] X. Chen et al., “MAGIC: Multimodal augmented generation with image-conditioned language models,” in Proc. CVPR, 2023.

[9] H. Ilharco et al., “AudioCLIP: Extending CLIP to audio with multimodal data,” in Proc. ICASSP, 2022.

[10] P. Girdhar et al., “ImageBind: One embedding space to bind them all,” in Proc. CVPR, 2023.

[11] H. Liu et al., “Visual instruction tuning,” in Proc. NeurIPS, 2023.

[12] Z. Zhu et al., “Mini-GPT4: Enhancing vision-language understanding with advanced LLMs,” in Proc. ACL, 2023.

[13] Y. Li et al., “DetGPT: Object detection meets multimodal foundation models,” in Proc. ECCV, 2024.

[14] X. Wang et al., “SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities,” in Proc. Interspeech, 2023.

[15] Z. Gu et al., “AnomalyGPT: Detecting Industrial Anomalies Using Large Vision-Language Models,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 3, pp. 1932–1940, 2024.

[16] H. Wang et al., “VisionGPT: LLM-Assisted Real-Time Anomaly Detection for Safe Visual Navigation,” arXiv preprint arXiv:2403.12415, 2024.

[17] Z. Hu and Z. Zhang, “SOWA: Adapting Hierarchical Frozen Window Self-Attention to Visual-Language Models for Better Anomaly Detection,” arXiv preprint arXiv:2407.03634, 2024.

[18] J. Dean et al., “Large Scale Distributed Deep Networks,” in Advances in Neural Information Processing Systems, vol. 25, 2012.

[19] P. Goyal et al., “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour,” arXiv preprint arXiv:1706.02677, 2017.

[20] S. Shazeer et al., “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” in Proceedings of the International Conference on Learning Representations, 2017.

[21] M. Shoeybi et al., “Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism,” arXiv preprint arXiv:1909.08053, 2019.

[22] N. Narayanan et al., “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, 2021.

[23] S. Micikevicius et al., “Mixed Precision Training,” in Proceedings of the International Conference on Learning Representations, 2018.

[24] Y. You et al., “Large Batch Optimization for Deep Learning: Training BERT in 76 minutes,” in Proceedings of the International Conference on Learning Representations, 2020.

[25] D. D. Johnson, “Accelerating Deep Learning with Dynamic Data Pruning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018.

[26] J. Dean et al., “Large Scale Distributed Deep Networks,” in Advances in Neural Information Processing Systems, vol. 25, 2012.

[27] P. Goyal et al., “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour,” arXiv preprint arXiv:1706.02677, 2017.

[28] S. Shazeer et al., “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer,” in Proceedings of the International Conference on Learning Representations, 2017.

[29] S. Micikevicius et al., “Mixed Precision Training,” in Proceedings of the International Conference on Learning Representations, 2018.

[30] Y. You et al., “Large Batch Optimization for Deep Learning: Training BERT in 76 minutes,” in Proceedings of the International Conference on Learning Representations, 2020.

[31] D. D. Johnson, “Accelerating Deep Learning with Dynamic Data Pruning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018.