GPU Memory Prediction for Multimodal Model Training

Jinwoo Jeong*
Korea University
Seoul, Republic of Korea
jwjeong@os.korea.ac.kr

Minchul Kang*
Korea University
Seoul, Republic of Korea
mckang@os.korea.ac.kr

Younghun Go
Korea University
Seoul, Republic of Korea
yhgo@os.korea.ac.kr

Changyong Shin
Korea University
Seoul, Republic of Korea
cyshin@os.korea.ac.kr

Hyunho Lee
Korea University
Seoul, Republic of Korea
hhlee@os.korea.ac.kr

Junho Yoon
KT Corporation
Seoul, Republic of Korea
jun_ho.yoon@kt.com

Gyeongsik Yang
Korea University
Seoul, Republic of Korea
g_yang@korea.ac.kr

Chuck Yoo
Korea University
Seoul, Republic of Korea
chuckyoo@os.korea.ac.kr

Abstract

As deep learning models in agentic AI systems grow in scale and complexity, GPU memory requirements increase and often exceed the available GPU memory capacity, so that out-of-memory (OoM) errors occur. It is well known that OoM interrupts the whole training itself and wastes substantial computational resources. Therefore, to prevent OoM, accurate prediction of GPU memory usage is essential. However, previous studies focus only on unimodal architectures and fail to generalize to multimodal models, even though the multimodal models are a common choice in agentic AI systems. To address this limitation, we propose a framework that predicts the peak GPU memory usage by analyzing the model architecture and training behavior of multimodal models. Specifically, the framework decomposes the multimodal model into its constituent layers and applies "factorization" to estimate the memory usage of each layer. Our evaluation shows that our framework achieves high prediction accuracy of ~8.7% average MAPE.

1 Introduction

Agentic AI systems increasingly rely on multimodal models to perform complex tasks. In particular, vision–language models, a representative type of multimodal architectures, serve as a core component of Agentic AI systems because they can integrate diverse modalities and adapt to tool interfaces [9]. For example, to enable an agent to plan actions with the external tools (e.g., databases, search engines, or code execution) in response to user queries, the models are trained or fine-tuned to generate API calls that understand such tools [4]. Recently, the increasing scale and complexity of the models have sharply raised GPU memory requirements during training. When the peak GPU memory usage consumed by the model exceeds the available capacity, OoM errors occur, interrupting training and often wasting substantial computational resources. Therefore, accurately predicting GPU memory usage in advance is critical to avoid OoM errors and ensure efficient, stable training.

To address this need, several studies have proposed methods for predicting GPU memory usage in unimodal architectures: 1) profiling-based prediction [3, 12, 13], which runs a few training iterations to predict peak memory usage—assuming that training consists of repeated, identical forward and backward propagation steps, and 2) formulation-based modeling [2, 6], which predicts the memory usage of individual layers in the model architecture to determine the GPU memory consumption.

However, recently multimodal models pose significant challenges for memory prediction. Such models comprise different types of modules (e.g., vision encoders and language decoders), each containing various layers (e.g., embedding and feed-forward networks). For example, the representative multimodal model LLaVA [7] consists of several hundred layers across multiple modules, making memory usage prediction highly challenging. Furthermore, the memory usage of a layer is determined by several factors, such as parameters and gradients. Thus, existing prediction methods have the following limitations. First, existing profiling-based methods require multiple pre-training runs, causing significant overhead that can take a long time [12]. Second, formulation-based methods work on specific layer types or particular model architectures so that they have to come up with a new formula each time when new modules are added to multimodal models. We have applied the method in [2] to a multimodal model, but we found that it does not work at all because the formula was designed for a specific unimodal

*The first two authors contributed equally to this work.

architecture and does not generalize to heterogeneous modules in multimodal models. To our knowledge, there is no literature that applies the profiling method to a multimodal model.

To address this limitation, we propose a framework that accurately predicts the peak GPU memory usage by analyzing the architecture and training behavior of multimodal models. Our evaluation on the LLaVA-1.5 (7B) model demonstrates that, even under diverse hyperparameter settings, our framework achieves MAPE as low as 8.7%.

2 Background & Motivation

Multimodal models. Multimodal models serve as a core component in Agent AI systems to plan the actions with various agent tools for the given request. It uses heterogeneous modules and layers to handle the tasks of different natures such as image captioning and image-text reasoning. This heterogeneity makes variations in the types and configurations of layers within each module. A representative model is LLaVA [7] that uses the pre-trained CLIP ViT-L/14 [10] as a vision encoder module to extract image features, and a Vicuna-based [1] model as a language decoder module to generate text. These two are connected by a projection layer that aligns the vision module's output with the language module's input.

Training of LLaVA proceeds in two stages: 1) pre-training, 2) fine-tuning. In the pre-training stage, only the projection layer is updated, while the vision module and language module remain frozen. In the subsequent fine-tuning stage, the projection layer and parts of the language module are updated, while the vision module remains frozen [7]. Note that this training behavior may vary depending on multimodal models and fine-tuning techniques (e.g., LoRA [5]). Such training behavior inevitably changes the GPU memory usage, in contrast to unimodal models. So it is very challenging to predict GPU memory usage for multimodal models.

GPU memory usage. In training DL models, the peak GPU memory usage is affected by multiple factors. The main factors are:

- **Model parameters (M_{param})**: weights and biases for each layer that must remain resident in GPU memory throughout forward and backward passes, and their memory footprint scales with model size.
- **Optimizer states (M_{opt})**: additional parameter states maintained by the optimizer to facilitate parameter updates, such as momentum terms and variance estimates in Adam.
- **Gradients (M_{grad})**: partial derivatives of the loss for each parameter, computed during backpropagation. They are stored temporarily until the optimizer step is performed.
- **Activations (M_{act})**: intermediate outputs of each layer in the forward pass. They must be stored in memory until their corresponding backward computations are complete.

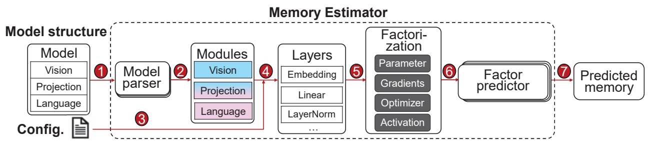


Figure 1. Workflow of the proposed framework.

Previous studies based on formula [2, 6] predict memory usage from these factors, but their methods focused on unimodal architectures. Multimodal models differ in having heterogeneous modules with distinct training behaviors. This motivates our framework that predicts GPU memory usage by explicitly modeling the architecture and training behavior of multimodal models.

3 Design

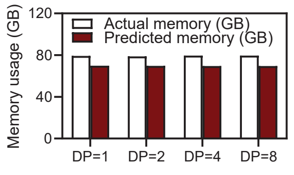
Fig. 1 shows the workflow of the proposed framework. It begins with the *Model parser* (①) that analyzes the model architecture and extracts key "modules" (②) based on modality. A configuration file provides training hyperparameters such as batch size (③). Each parsed module is then further decomposed into fine-grained "layers", such as `nn.Linear` (④), to facilitate the next step called "factorization". Specifically, we identify the layers within the model's modules using the PyTorch API. The framework factorizes memory usage for each layer into four factors: model parameters, gradients, optimizer states, and activations (⑤). Note that each layer can have different factors. For example, an embedding layer in a frozen vision module has neither gradients nor optimizer states, whereas a feed-forward layer in a language module requires both in addition to its parameters. Thus, this factorization reflects the structure and training behavior of multimodal models.

We then predict the peak memory usage of each layer by applying a per-factor analytical equation, called *factor predictor* (⑥). In this paper, we focus on presenting the overall framework and do not include detailed derivations of each factor. For each layer, the factor predictor computes four factors (e.g., parameters and gradients), and it repeats the calculation for all layers across all modules.

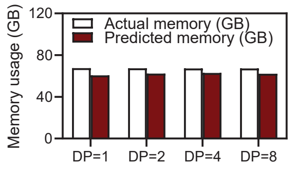
Formally, we predict the peak memory usage as follows:

$$M_{\text{peak}} = \sum_{\text{module}} \sum_{\text{layer}} \left(M_{\text{param}} + M_{\text{opt}} + M_{\text{grad}} + M_{\text{act}} \right) \quad (1)$$

Here, M_{act} in a multimodal model is obtained by computing activations for modalities whose parameters are being updated, whereas in a unimodal model it is computed for the single modality being trained. Finally, the framework generates the predicted peak memory usage (⑦), which is used to prevent OoM errors.



(a) SeqLen=1024, MBS=16.



(b) SeqLen=2048, MBS=8.

Figure 2. GPU memory usage prediction results.

4 Evaluation

To demonstrate that our framework maintains robust predictive accuracy under varying hyperparameter configurations, we evaluate the LLaVA-1.5 (7B) model in two hyperparameter settings. In the first setting, we set the sequence length (SeqLen) to 1,024 and the micro-batch size (MBS) to 16, varying the data parallelism (DP) from 1 to 8. In the second setting, we use SeqLen of 2048 and MBS of 8, again varying DP from 1 to 8. All experiments run on a single GPU node with eight NVIDIA H100 80GB GPUs (NVLink), using PyTorch 24.07 [8] and ZeRO-2 from DeepSpeed [11].

Fig. 2a shows the prediction accuracy for the first setting. We observe an average MAPE of 13% across various DP degrees. Fig. 2b shows the prediction accuracy for the second setting. The results show that the average MAPE is 8.7%, demonstrating that our framework provides consistent and reliable prediction accuracy across various training setups.

5 Future work

In the future, we plan to extend our memory usage prediction to include other optimization techniques such as parameter-efficient fine-tuning and kernel fusion. We also plan to extend our memory prediction to inference workloads of agentic AI systems that manage memory with key-value caching and multi-turn orchestration.

6 Conclusion

In this paper, we propose a framework to predict GPU memory usage for training multimodal models. The framework parses the model to extract modules. Each module is further decomposed into individual layers, and memory usage is predicted based on four factors. The factor predictor applies per-factor analytical equations to each layer, and the predictions are aggregated to compute the peak GPU memory usage. The proposed framework achieves high prediction accuracy, with MAPE values of ~8.7% in two different settings, demonstrating reliable and consistent performance across various hyperparameters.

Acknowledgments

This work was supported by KT (Korea Telecom)-Korea University AICT R&D Center. Corresponding authors are Gyeongsik Yang and Chuck Yoo.

References

[1] Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)* 2, 3 (2023), 6.

[2] Kazuki Fujii, Kohei Watanabe, and Rio Yokota. 2024. Accelerating large language model training with 4d parallelism and memory consumption estimator. *arXiv preprint arXiv:2411.06465* (2024).

[3] Yanjie Gao, Yu Liu, Hongyu Zhang, Zhengxian Li, Yonghao Zhu, Haoxiang Lin, and Mao Yang. 2020. Estimating GPU memory consumption of deep learning models. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1342–1352.

[4] Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. 2024. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage. *arXiv preprint arXiv:2412.15606* (2024).

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[6] Taeho Kim, Yanming Wang, Vatshank Chaturvedi, Lokesh Gupta, Seyeon Kim, Yongin Kwon, and Sangtae Ha. 2024. LLMem: Estimating GPU Memory Usage for Fine-Tuning Pre-Trained LLMs. arXiv:2404.10933 [cs.AI] https://arxiv.org/abs/2404.10933

[7] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems* 36 (2023), 34892–34916.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[9] Kangan Qian, Sicong Jiang, Yang Zhong, Ziang Luo, Zilin Huang, Tianze Zhu, Kun Jiang, Mengmeng Yang, Zheng Fu, Jinyu Miao, et al. 2025. Agentthink: A unified framework for tool-augmented chain-of-thought reasoning in vision-language models for autonomous driving. *arXiv preprint arXiv:2505.15298* (2025).

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[11] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3505–3506.

[12] Changyong Shin, Gyeongsik Yang, Yeonho Yoo, Jeunghwan Lee, and Chuck Yoo. 2022. Xonar: Profiling-based job orderer for distributed deep learning. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 112–114.

[13] Gyeongsik Yang, Changyong Shin, Jeunghwan Lee, Yeonho Yoo, and Chuck Yoo. 2022. Prediction of the resource consumption of distributed deep learning systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 2 (2022), 1–25.