Testing-based GPU-Memory Consumption Estimation for Deep Learning

Haiyi Liu[1,a)] Shaoying Liu[2,†1,b)] Ai Liu[2]

Abstract: Deep learning(DL) has been successfully applied in many software systems and deployed to a variety of server. The training of DL needs a lot of GPU computing resources. However, it is difficult for developers to accurately calculate the GPU resources that the model may consume before running the model, which brings great inconvenience to the development of DL system. Especially nowadays, a lot of model training runs on cloud services. Therefore, it is very important to estimate the GPU-memory resources that any model may use in a certain computing framework. The existing work mainly focuses on the static analysis method to evaluate the GPU-memory consumption, which is highly coupled with the framework implementation, and lacks the research on the software testing and evaluation method of GPU-memory consumption (It does not depend on the framework implementation itself). In this paper, we propose a new method to estimate the memory consumption of DL framework. The method is based on software testing and static analysis estimation. Firstly, heuristic random search algorithm is used to explore the real GPU-memory consumption of different DL models at runtime. At the same time, the software static analysis method is used to evaluate the theoretical memory consumption of DL model, which can reduce the number of model testing (because the testing model requires computing resources). Finally, the known data is modeled to estimate the real memory consumption of different models in different computing frameworks. To evaluate the effectiveness of our proposed method, we apply it to the mainstream computing framework, i.e., TensorFlow, Pytorch. The results show that our method can achieve more accurate evaluation of GPU-memory consumption without knowing the operation mechanism of the framework.

Keywords: Deep Learning, Program static analysis, Automated testing, Bug detection

1. Introduction

In recent years, with the continuous improvement of computer performance and the continuous accumulation of various data, the research and engineering implementation of artificial intelligence algorithms have made great progress. Deep learning system is the most applied and realized system in artificial intelligence system. It is widely used in many scenes, such as image recognition, speech recognition, recommendation system and so on. Although the accuracy and breadth of artificial intelligence systems such as deep learning are improving year by year, the hardware cost and time cost of constructing neural network system are also increasing year by year. In 2020, the gpt-3[1] model released by open AI has 175 billion parameters, and the cost of training the network is as high as $12 million and the high cost of model training is a common phenomenon of neural network system. Facing such a high cost of model training, how to ensure that the constructed deep learning model will not report errors in the model training stage has become an important issue. Among the many errors that may occur in the construction of deep learning model, video memory overflow is a kind of error that greatly affects the model training. This error is caused by the fact that developers cannot accurately estimate the size of the video memory occupied by the model before the model runs, so they cannot find the upper and lower limits of the super parameters suitable for their own development environment. According to relevant research literature, among all program failures of deep learning jobs, out of memory(OOM) account for 9.1% (including GPU and CPU)[2], and often occur in training process[3], which makes all the previous efforts of ongoing model training wasted. This not only wastes GPU computing resources, but also affects the development progress of engineers. Therefore, the memory consumption of different deep learning models and various deep learning libraries becomes particularly important. In terms of deep learning model and memory consumption, many researchers have made great contributions and provided corresponding solutions from different angles. The main methods include memory exchange, memory sharing, recalculation, and compressed neural network, etc. these methods reduce the use of memory in the training process of deep learning model by analyzing the calculation graph model and using the technologies such as liveness analysis in static analysis or dynamic memory sharing and memory exchange. But their technology is usually used to make the built model input a larger batch size in the current hardware environment. Not to evaluate that the built model will cause memory overflow in a certain environment before model training.

In terms of deep learning framework and memory consump-

[1] Graduate School of Advanced Science and Engineering, Hiroshima University
[2] Graduate School of Advanced Science and Engineering, Hiroshima University
[†1] Presently with Graduate School of Advanced Science and Engineering, Hiroshima University
[a)] d200101@hiroshima-u.ac.jp
[b)] sliu@hiroshima-u.ac.jp

tion, Gao et al.[4] proposed the method of using static analysis and calculation diagram and resident buffer to predict the memory utilization before model training.

Although the above methods have made effective solutions, there are still the following problems in the memory consumption evaluation of deep learning model:

(1) `Deep learning library` (e.g., TensorFlow, Pytoch)[5] generally contains two main functions, automatic differentiation, and GPU acceleration. Automatic differentiation is usually implemented by deep learning library, while GPU accelerated process is usually implemented by calling multiple NVIDIA components(e.g., CUDA, cudnn), it is difficult to achieve static analysis for the cooperative calls of multiple non-open-source components. Because the components called by the framework are in the closed source state, users cannot carry out common memory analysis methods such as context analysis. It also makes the deep learning library a black box for users.

(2) `Each framework of deep learning` is iterating rapidly in months, and new deep learning frameworks emerge one after another. The method of static analysis requires experts to analyze the framework. Therefore, the static analysis method undoubtedly increases the labor cost and time cost of evaluating the memory consumption of the deep learning model[6].

To solve the above problems, this paper proposes a method based on the combination of static analysis and dynamic test modeling analysis[7][8]. Firstly, using the method of static analysis, the calculation graph of neural network is statically analyzed to pre estimate the memory that may be consumed by the model. Then the pre estimated model is run in the deep learning framework to obtain the real value of the model under the framework. Finally, Polynomial regression**?**[9] is used to analyze the gap between the memory consumption estimated by static analysis and that of the real model, to deduce the possible memory consumption of the deep learning framework under different models.

2. Overview

In this chapter, we first define the formal description of the problem[10] and formalize the solution of the problem. For each function of the formalized scheme, the corresponding solution is given. Finally, a test-based $GPU-memory$ usage evaluation method is formed.

2.1 Formal Description of Problem

In order to more clearly describe the problem that we solve and the methods to be proposed. First, we write the formal specification of the deep learning framework and the formal specification of the deep learning model in the framework. Then we use two formal specifications to express the problems to be solved.

(1) Formal specification for deep learning framework.[11] Let's define set API as $\mathbf{I} = \{A_i\}_{i=1}^{n} = \{A_1, A_2, \ldots, A_n\}$, Where A_i is the existing API in the neural network framework, and n is the number of $\mathbf{I}$. At the same time, The set of hyperparameters to be set for each $\mathbf{I}$ is defined as $\mathbf{HP_{A_i}} = \{p_{A_i}^{j}\}_{j=1}^{n} = \{p_{A_i}^{1}, p_{A_i}^{2}, \ldots, p_{A_i}^{n}\}$ Where $p_{A_i}^{j}$ is the specific hyperparameter to be set in each $\mathbf{I}$. A_i is the element in set API and j is the number of all Hyperparameters of the $\mathbf{I}$. For example, $\sum_{i=1}^{n} |HP_{A_i}|$ can represent the types of all settable hyperparameters in the framework.

(2) Formal specification for deep learning model in framework[12] Next, we describe the form of the model in the framework based on the definition of the deep learning framework. Given a set of $\mathbf{I}$, We do a finite Cartesian product $\overbrace{I \times I \times \cdots \times I}^{K}$ denoted as I^K and $I^K = \{< A_1, A_2, \ldots, A_n > | A_i \in I, 1 \leq i \leq K\}$.Then, the model in the deep learning framework can be defined as $model \in I^* = \bigcup_{K=1}^{\infty} I^K$.The overview is shown in Figure 1.

2.2 Method Overview

Because in the DL framework, the model usually runs in the form of calculation graph, so we mark the calculation graph[13] set as G. Let $CG : model \rightarrow G$ represents the mapping between the model and the calculation graph, for a given input $m \in model$, there will be a corresponding calculation garaph $g \in G$.

Meanwhile, let GU be the set of interger means the size of the $GPU-memory$ consumed by the model, for each $m \in model$, a corresponding $GPU-memory$ usage can be obtained by running the model or static analysis for the graph[14]. Let $GV : model \rightarrow GU$ be a function of GU for the model. Through the investigation of previous studies, it can be seen that the static analysis of the calculation graph can roughly estimate the usage of the $GPU-memory$ of the model at run time. Therefore, let $SGV : graph \rightarrow GU$ be a function of GU for the model. There will be a certain gap between the GU obtained by static analysis of the model and the GU obtained by running the model. This gap can be defined as $Gap(model) = GV(model) - SGV(CG(model))$. It is critical to define the relationship between $Gap(model)$ and $model$. Not only can it be used to get a more accurate model $GPU-memory$ usage, but it can also be used to evaluate the execution efficiency of the DL framework[15].

In order to find the specific mathematical form of $Gap(model)$, we propose a data fitting method based on orthogonal array test strategy(OATS). First, a certain scale of deep learning model is generated through the orthogonal array test strategy, which is used as a test case to test the GU value (Test Oracle) of the DL framework at runtime. Next, use the regression algorithm to find the relationship between test case and test oracle, that is, to obtain $GV(model)$. However, if we want to know $Gap(model)$, we still need to know the specific value of SGV. This paper adopts the method of static analysis of the computational graph, and evaluates the specific value of $SGV(CG(model))$ [2]through the analysis of the tensor scale.

3. preliminaries

3.1 Orthogonal array testing strategy

Orthogonal array testing strategy(OATS) is a technology applied to software integration testing. The shape of the test case table depends on the number of factors and levels in the test.[7]

Definition 1 An Orthogonal can be defined as $OA(p, l, n, d)$,

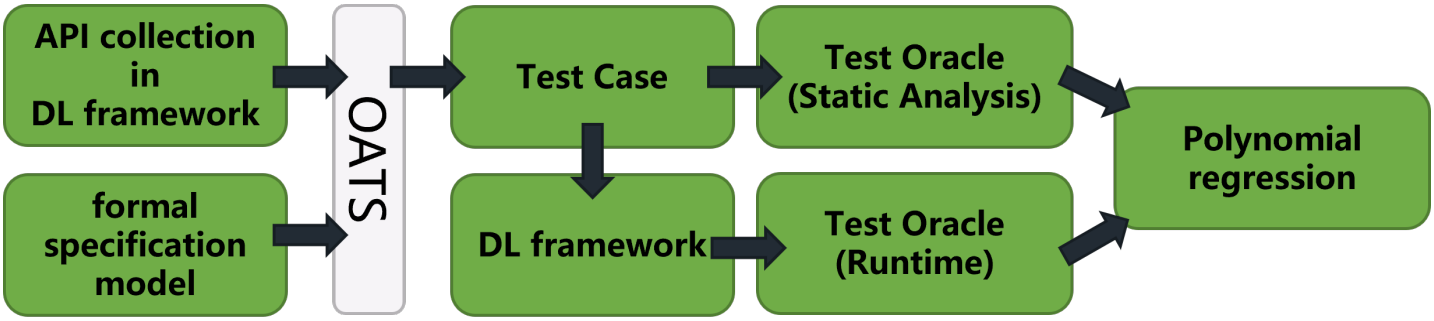

Fig. 1 Overview

where:

(1) p is the number of rows in the array.In the test, it represents how many test cases there are.

(2) l indicates how many parameters (factors) need to be tested. In this article, factors represent hyperparameters in neural networks.

(3) n represents the value range of each parameter.

(4) d is the strength of the array. An orthogonal array has strength d if in any $p \times d$ sub-matrix (that is, select any d columns), each of the $n \times d$ possible d-tuples (rows) appears the same number of times.

3.2 Polynomial Regression

Definition 2 we have a polynomial equation of degree n represented as:

$Y = \delta_0 + \delta_1 x_i + \delta_2 x_i^2 + \ldots + \delta_n x_i^n + \epsilon (i = 1, 2, \ldots, m)$

can be expressed in matrix form in terms of a design matrix X, a response vector $\vec{y}$,a parameter vector $\vec{\delta}$ and a vector$\vec{\epsilon}$of random errors. The i-th row of X and $\vec{y}$ will contain the x and y value for the i-th data sample,Then the model can be written as a system of linear equations:

$\vec{y} = X\vec{\delta} + \vec{\epsilon}$

The vector of estimated polynomial regression coefficients is:

$\vec{\delta} = (X^T X)^{-1} X^T \vec{y}$

assuming $m < n$ which is required for the matrix to be invertible, then since X is a Vandermonde matrix, the invertibility condition is guaranteed to hold if all the x_i values are distinct. This is the unique least-squares solution.

4. Approach

In this section, we first introduce what OATS is and how to use OATS to generate test cases that can test the DL framework, and then analyze the feasibility of test cases generated based on OATS and the ability of the generated test cases and test oracle to be applied to regression analysis feasibility. Finally, the regression model and static analysis model used in this method are introduced.

4.1 Test case generation based OATS

The purpose of testing is to find out how much $GPU-memory$ is consumed by different models running under a certain framework. But there are many hyperparameters e.g., batch size[16].in the deep learning algorithm, and not all hyperparameter changes will have a huge impact on the $GPU-memory$. The static analysis of the deep learning calculation graph can filter out the APIs that have a greater impact on the memory consumption.

Observatios and motivations about API screening: GPU's advantage lies in parallel computing. In the process of training deep learning models, there are a large number of operators that need to be calculated in parallel. For example, feature mapping in forward propagation, gradient mapping in back propagation, etc. Therefore, we have screened APIs related to convolution operation, pool operation, and Batch Normalization that will generate a large number of parallel calculations. In each API, the input scale and output scale of each layer of neurons can be set, and different parameters correspond to different memory usage. condition. In addition, the depth in deep learning is also a major factor in consuming memory. Therefore, in the test, models with different depths and different structures will be tested orthogonally[17]. Corresponding to the memory consumed by different models. Because the memory consumption of the underlying framework will not decrease with the increase of the influencing parameters in the model, that is, the direction of data change is known, and only the rate of change is unknown. Therefore, the data obtained by the orthogonal test is sufficient for multivariate polynomia regression analysis.

Let's take the classic visual geometry group network as a case study.Suppose that through the static analysis[18] of the neural network model, three representative hyperparameters are selected, namely Batch-size, Depth and Number of convolutional layers[19]. These three hyperparameters constitute the factors in the orthogonal array. As shown in Table 1.In an orthogonal array, the range of values for each factor is called levels. Table 1 shows an orthogonal array with a factor of 3 and levels of 4. If a comprehensive experimental method is used for testing, up to 3^4 tests are required.And the number of tests increases exponentially with the value of levels. However, using orthogonal experiments to generate orthogonal arrays requires only 4^2 tests. In other words, when the levels become very large, a comprehensive test test is impossible and unnecessary. Therefore, this article uses the OATS method to generate test cases[20].

4.2 Polynomial regression

In 4.1, we got a lot of pairs of test case and test Ora-

Table 1 Orthogonal test example of VGG network

Test Number	Batch-size	Depth	Number of convolutional layers
Case1	4	11	8
Case2	8	13	10
Case3	16	16	13
Case4	32	19	16

cle, where test case is marked as *model* and test Oracle is marked as GU.Because the model is generated by transforming parameters. therefore, *model* can be denoted as $model(hyperparameter_1, hyperparameter_2, \ldots, hyperparameter_n)$ and The hyperparameters are derived from the static analysis calculation graph.

Next, we use polynomial regression to find the relationship between hyperparameters and GU, which is equivalent to finding a way to solve $GV(model)$. Furthermore, $SGV(model)$ is known. We have also found a way to solve $Gap(model)$.

5. Conclusion and Future work

In this article, we propose a test-based method to evaluate the memory usage of the DL framework. This method is different from the previous static analysis method. The possible errors of the static analysis method can be corrected through testing, and the possible GPU-memory usage of the deep learning model can be better evaluated before the deep learning model is running.

At present, it is only a theoretical framework. In the future, this method will be used to automatically generate a large number of test cases to test the mainstream DL framework, so as to prove the effectiveness of this method.

Acknowledgments The research was supported by ROIS NII Open Collaborative Research 2021-(21FS02).

References

[1] Floridi, L. and Chiriatti, M.: GPT-3: Its nature, scope, limits, and consequences, *Minds and Machines*, Vol. 30, No. 4, pp. 681–694 (2020).
[2] Zhang, R., Xiao, W., Zhang, H., Liu, Y., Lin, H. and Yang, M.: An empirical study on program failures of deep learning jobs, *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, pp. 1159–1170 (2020).
[3] Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A. and Tonella, P.: Taxonomy of real faults in deep learning systems, *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 1110–1121 (2020).
[4] Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H. and Yang, M.: Estimating gpu memory consumption of deep learning models, *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1342–1352 (2020).
[5] Guo, Q., Xie, X., Li, Y., Zhang, X., Liu, Y., Li, X. and Shen, C.: Audee: Automated testing for deep learning frameworks, *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, pp. 486–498 (2020).
[6] Pham, H. V., Lutellier, T., Qi, W. and Tan, L.: CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries, *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, pp. 1027–1038 (2019).
[7] LAZIĆ, L.: Use of orthogonal arrays and design of experiments via Taguchi methods in software testing, *18th International Conference on APPLIED MATHEMATICS (AMATH 2013), Budapest, Hungary* (2013).
[8] Wu, H.: Application of orthogonal experimental design for the automatic software testing, *Applied mechanics and materials*, Vol. 347, Trans Tech Publ, pp. 812–818 (2013).
[9] Vesely, M.: Computer Curve Fitting of Polynomials, Technical report, ILLINOIS UNIV URBANA COORDINATED SCIENCE LAB (1972).
[10] Liu, S. and Nakajima, S.: Automatic test case and test oracle generation based on functional scenarios in formal specifications for conformance testing, *IEEE Transactions on Software Engineering* (2020).
[11] Seshia, S. A., Desai, A., Dreossi, T., Fremont, D. J., Ghosh, S., Kim, E., Shivakumar, S., Vazquez-Chanlatte, M. and Yue, X.: Formal specification for deep neural networks, *International Symposium on Automated Technology for Verification and Analysis*, Springer, pp. 20–34 (2018).
[12] Dreossi, T., Ghosh, S., Sangiovanni-Vincentelli, A. and Seshia, S. A.: A formalization of robustness for deep neural networks, *arXiv preprint arXiv:1903.10033* (2019).
[13] Chen, T., Xu, B., Zhang, C. and Guestrin, C.: Training deep nets with sublinear memory cost, *arXiv preprint arXiv:1604.06174* (2016).
[14] Wang, L., Ye, J., Zhao, Y., Wu, W., Li, A., Song, S. L., Xu, Z. and Kraska, T.: Superneurons: Dynamic GPU memory management for training deep neural networks, *Proceedings of the 23rd ACM SIGPLAN symposium on principles and practice of parallel programming*, pp. 41–53 (2018).
[15] Peng, X., Shi, X., Dai, H., Jin, H., Ma, W., Xiong, Q., Yang, F. and Qian, X.: Capuchin: Tensor-based gpu memory management for deep learning, *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 891–905 (2020).
[16] Papini, M., Pirotta, M. and Restelli, M.: Adaptive batch size for safe policy gradients, *The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)* (2017).
[17] Liu, S. and Nakajima, S.: A" Vibration" method for automatically generating test cases based on formal specifications, *2011 18th Asia-Pacific Software Engineering Conference*, IEEE, pp. 73–80 (2011).
[18] Radiuk, P. M. et al.: Impact of training set batch size on the performance of convolutional neural networks for diverse datasets, *Information Technology and Management Science*, Vol. 20, No. 1, pp. 20–24 (2017).
[19] Mittal, S. and Vaishay, S.: A survey of techniques for optimizing deep learning on GPUs, *Journal of Systems Architecture*, Vol. 99, p. 101635 (2019).
[20] Di, B., Sun, J., Li, D., Chen, H. and Quan, Z.: GMOD: a dynamic GPU memory overflow detector, *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, pp. 1–13 (2018).