Real-time Scheduling for Deep Learning Tasks on a Multiprocessor Environment

2210011 Ishii Hibiki

In recent years, machine learning such as deep learning has been spreading around various fields. However, it is difficult for embedded systems to execute machine learning. That is because most embedded systems do not have sufficient resources to execute machine learning with high computational load. Therefore, methods to have machine learning processed by edge-servers instead of embedded systems are being considered. In the edge-server, the order of execution must be determined according to the requests from each embedded system. The server must satisfy the predetermined timing constraints (deadlines) of the requests. This is real-time scheduling. In the inference in deep learning, increasing the number of layers improves accuracy but increases computation time. Increasing computation time means that scheduling multiple tasks while satisfying the predetermined timing constraints is difficult. On the other hand, if the computation time is shortened using fewer layers, the accuracy will decrease. Because of this trade-off problem, scheduling deep learning tasks is difficult. In 2020, a method was proposed for scheduling aperiodic deep learning tasks on a single-processor environment. However, a method for scheduling periodic tasks on a multiprocessor environment has not been proposed. Therefore, the objective of this study is to develop an algorithm for real-time scheduling of a set of periodic deep learning tasks on a multiprocessor environment.

Both the previous study and this study's method utilize the neural network model of deep learning with one input layer and multiple output layers (Multi-Exit neural network model). And both studies apply a task model in which the neural network is divided into execution units called stages. This task model is regarded as Imprecise Computation. Imprecise Computation divides the deep learning task into stages that must be executed (Mandatory stages) and stages that are executed if time is available (Optional stages). The proposed algorithm consists of two modules. The first module (Selection module) is a module that selects Optional stages to satisfy the multiprocessor scheduling policy. The second module (Scheduling module) performs scheduling in a multiprocessor environment. In the selection module, I propose an algorithm to select Optional stages that can obtain high accuracy while satisfying the scheduling policy by applying the knapsack problem algorithm. The knapsack problem is a combinatorial optimization problem where the combination of items with weight and value is determined so that the total value is as large as possible, and the total weight is less than or equal to the knapsack's capacity. In this study's method, I define the weight of an item as the processor utilization of the Optional stage, the value of an item as the accuracy gained by the Optional stage, and the knapsack's capacity as the maximum processor utilization (the number of processors) that can be assigned to task's execution. By the above definition, I apply the solution algorithms of the knapsack problem. Typical algorithms for the knapsack problem include approximate solution model (greedy method) and exact solution model (dynamic programming method). I designed two types of solutions for selecting Optional stages: an approximate solution model and an exact solution model. In the scheduling module, I use the existing P-Fair scheduling algorithm. P-Fair scheduling algorithm is the real-time scheduling algorithm for periodic tasks on a multiprocessor environment. This algorithm is a method that decomposes a task into multiple smaller tasks (subtasks) and schedules them in units of subtasks.

As a comparison target for the proposed method, I extend the method of the previous study for a single-processor environment and aperiodic tasks scheduling to the method for a multiprocessor environment and periodic tasks scheduling. The previous study's algorithm consists of three parts. The first part is the EDF scheduling part of the Mandatory stages, the second part is the selection part of Optional stages, and the third part is the EDF scheduling part of the selected Optional stages. EDF scheduling algorithm is the real-time scheduling algorithm for periodic tasks on a single-processor environment. This algorithm schedules a task with the closest deadline preferentially when events such as arrivals of execution requests and deadlines occur. The comparison target divides the (original) deep learning task set by the number of processors, assigns the divided subset to each processor, and

performs the selection part of Optional stages and the EDF scheduling part of Mandatory stages and Optional stages on each processor. In the selection part of the previous study and the comparison target, the combinatorial optimization problem is considered to determine the combination with the minimum execution time among the combinations of Optional stages that can achieve the maximum total accuracy. The combinatorial optimization problem is then solved using a dynamic programming algorithm and the combination of Optional stages is calculated.

Both the proposed and comparison target methods were designed in C programming language in Visual Studio 2022. In this study, the methods were evaluated as a simulation. In the simulation procedure, a set of virtual deep learning tasks was randomly generated and input to the proposed method and the comparison target. I examined the selection results of the Optional stages and measured the processing time of the Selection module. I finally obtained the average value (average accuracy) from the output accuracy of all tasks in a task set. Based on the above procedure, 100 virtual deep learning task sets were randomly generated, and I obtained 100 average accuracies and 100 processing times through the simulation. The average value (final average accuracy and final average processing time) for 100 patterns is the result of the simulation. I examined the relationship between the number of tasks and simulation results (final average accuracy and final average processing time).

First, from the relationship between the number of tasks and final average accuracies, the proposed methods with approximate solution model and exact solution model obtained higher accuracy than comparison target even as the number of tasks increased. Therefore, the results showed that the proposed method obtained a combination of Optional stages with higher accuracy than the comparison target while satisfying the scheduling policy. Therefore, it was found that the proposed method executes deep learning tasks in a way that ensures higher accuracy while scheduling them within their deadlines. Next, from the relationship between the number of tasks and final average processing time, the results showed that the proposed method (approximate solution model) executes in a shorter time than the comparison target.

From the above evaluation, I found that the proposed method (approximate solution model) is the best method for scheduling a set of periodic deep learning tasks on a multiprocessor environment. The future work is to improve the algorithm so that the Selection module of proposed method (exact solution model) can be processed faster. Another work is to develop an algorithm that enables dynamic Optional stage selection and scheduling when task parameters change or when a new task is input. I developed a new real-time scheduling algorithm for a set of periodic deep learning tasks on a multiprocessor environment through this study. I also presented the performance of the developed algorithm. In the future, the developed algorithm is expected to be applied to an edge-server for real-time inference by embedded systems.