Technology Reports

Deep Learning | Container Virtualization | Parallel Processing

Deep Learning Platform Capable of Rapid Creation and Deployment of AI Models

Service Innovation Department Takero Ibuki†

AI development using deep learning is being conducted enthusiastically in recent years, but to realize an accurate AI, developers must run multiple learning trials, modifying learning parameters on a trial-and-error basis. This trial-and-error process is a major factor in increasing development time. There are also many frameworks and libraries for performing deep learning, and it is becoming a complex task just to manage and update these systems. NTT DOCOMO has developed a deep learning platform to resolve these issues, enabling highly accurate learning in a short period of time. This article gives details of this platform and describes the technology.

1. Introduction

Deep learning using deep neural networks has attracted much attention recently in Artificial Intelligence (AI), artificially performing tasks such as image recognition, translation, and speech recognition using computers. Neural network*1 system technologies have existed for some time, but due to issues with accuracy, they were not practical. However, major advances in General Purpose computing on Graphics Processing Unit (GPGPU)*2 computing resources, Compute Unified Device Architecture (CUDA)®*3, and the development of the NVIDIA CUDA Deep Neural Network library (CuDNN)*4, have made it possible to implement deeper neural networks, greatly increasing their accuracy. Many frameworks*5 for developing deep neural networks, such as Tensorflow®*6 and Caffe*7 have appeared, and AI development using deep learning is now being done in many research facilities and enterprises. NTT DOCOMO is also using deep learning technology in AI services such as

† Currently DOCOMO Innovations, Inc.

*1 Neural network: An entity that numerically models nerve cells within the human brain (neurons) and the connections between them. It is composed of an input layer, an output layer and intermediate layers and are able to approximate complex functions by varying the number of neurons and layers and the strength of connections between layers.

the image recognition Application Programming Interfaces (API)*8 being provided by docomo Developer Support. In development of this AI, multiple learning trials under different conditions needed to be done in order to obtain accurate results, and since these were done manually, it required much manpower and did not utilize servers efficiently. It was also difficult to build and update the multiple libraries*9 and framework environments used.

For these reasons, NTT DOCOMO has developed a deep learning platform using container virtualization technology*10 and an original scheduling system to process large volumes of conditions automatically. This has enabled highly accurate learning in short periods of time.

This article describes details of the technologies used in the deep learning platform developed by NTT DOCOMO, and the effects brought about by the platform.

2. Deep Learning and Related Issues

2.1 Overview of Deep Learning

Deep learning is a type of machine learning*11 for technologies using deep neural networks, which are neural networks having many intermediate layers.

Deep learning consists of the two phases: the learning phase, in which patterns and rules are learned from large amounts of data, and the inference phase in which the result of learning is used to infer which patterns apply to the new data. The neural network after learning is referred to as an AI model.

There are a wide range of AI models and services, such as image recognition using image data, or speech recognition and translation using speech data.

2.2 Tuning Parameters

For deep learning, it is not the case that by simply preparing the data, the best patterns and rules will be learned automatically. Learning conditions such as initial values and learning rates, which are called hyperparameters, must also be configured [1] [2]. Some typical hyperparameters are shown in Table 1.

Even with the same data, different hyperparameters

Table 1 Typical hyperparameters

Hyperparameter	Description
Minibatch size	Number of data items processed in a single trial
Learning rate	Breadth of weighting updates
Number of iterations	Number of trials
Weighting initial value	Initial value of neuron connection strength
Activation function	Neuron activation method
Optimization function	Method for deciding how learning progresses
Drop-out rate	The rate for disabling neurons in a given node

*2 GPGPU: The use of GPUs, which are generally used in computers for rendering and other types of image processing, for other types of applications. GPGPU excels at parallel distributed processing.
*3 CUDA®: A general-purpose parallel programming environment for GPUs provided by NVIDIA. A registered trademark of NVIDIA Corp.
*4 CuDNN: A deep learning CUDA library (See *8) published by NVIDIA. A registered trademark of NVIDIA Corp.
*5 Framework: Software that encompasses functionality and control structures generally required for software in a given domain. With a library, the developer calls individual functions, but with a framework, it handles the overall control and calls individual functions added by the developer.
*6 Tensorflow®: A deep learning programming framework (See *5). A registered trademark of Google Inc.
*7 Caffe: A deep learning programming framework (See *5).

can result in issues such as no improvement through training, or a problem called over-learning, in which the network loses generality and becomes biased to specific data only. Selecting the best hyperparameters is extremely important to building a deep learning model.

There are many types of hyperparameter, and setting them manually by trial-and-error requires large amounts of human work. Manually running trials while changing hyperparameters requires logging into each server manually and checking the server state before each trial, which results in poor server utilization and increases the time required to obtain results.

2.3 Building Environments

Both the learning and inference phases of deep learning are composed of many matrix operations. For this reason, GPUs, which are capable of parallel processing and have more cores than a CPU, are able to accelerate deep learning. GPUs from NVIDIA are widely used. GPUs were originally designed for image processing, but they are also used for high-speed computing for deep learning and other applications, and there are programming environments and libraries such as CUDA, for general processing, and CuDNN for deep learning. Because using these programming environments and libraries requires learning specialized languages that extend the C programming language, which takes time, universities and enterprises have developed frameworks that make deep learning programming easier and these have been released as Open Source Software (OSS)*11. Typical frameworks include Tensorflow and Caffe. Advantages of these frameworks are that they can be used with Python*12, which is often used for data analysis, and there are API that make it easier to code neural network structures.

However, these libraries and frameworks must be kept up-to-date to ensure that they will run and produce the fastest processing speeds. Keeping up with all updates is labor intensive and can be quite complex due to dependency issues such as library versions that only support certain GPUs and frameworks only supporting certain versions of libraries.

3. Deep Learning Platform

3.1 Approaches to Resolving Issues

Below we describe methods for resolving the two issues described above: (1) inefficient human work in selecting hyperparameters, and (2) complexity in building environments; and the deep learning platform system that we have developed.

1) A System to Process Learning Jobs in Parallel, Automatically According to Server State

The following observations apply to issue (1) above.

- Parameter changes and learning processes are performed manually and successively.
- The state of each server is monitored manually.

Our approach to resolving these problems was to control a job queue and execute jobs automatically in parallel, according to server state. We developed a system able to manage servers centrally and run a sequence of learning processes in

*8 API: A set of rules that define programming procedures used to access commands and functions when software is developed for a particular platform (OS or middleware).

*9 Library: A collection of high-versatility programs in a reusable form.

*10 Container virtualization technology: A type of computer virtualization technology in which dedicated areas called containers are created on the host OS and the application software runs within these containers.

*11 Machine learning: A framework that enables a computer to learn useful judgment standards through statistical processing from sample data.

parallel.

2) Adoption of Container Virtualization Using Docker®*14

The following observations apply to issue (2) above.

- Updating libraries and frameworks consumes time.
- Dependency relationships between libraries and frameworks make updates difficult.

As a solution to these issues, we focused on a virtualization technology that enables us to build environments independently for each job within the server. In this case, we only needed libraries and frameworks for deep learning and not OS functionality as in conventional host virtualization technology*15, so we decided to use container virtualization technology, and specifically, Docker (**Figure 1**). Docker is software that uses Linux®*16 container virtualization technology, which isolates resources used by the OS so that multiple environments can be established. It is faster than conventional virtualization technologies such as hypervisor virtualization*17 and host virtualization because it does not require a new OS to be started. It also has the benefit that once a container image has been created, it can be moved to another environment easily.

3.2 System Overview

A system overview of the deep learning platform we have developed is shown below (**Figure 2**). The platform is composed of a Master server, Worker

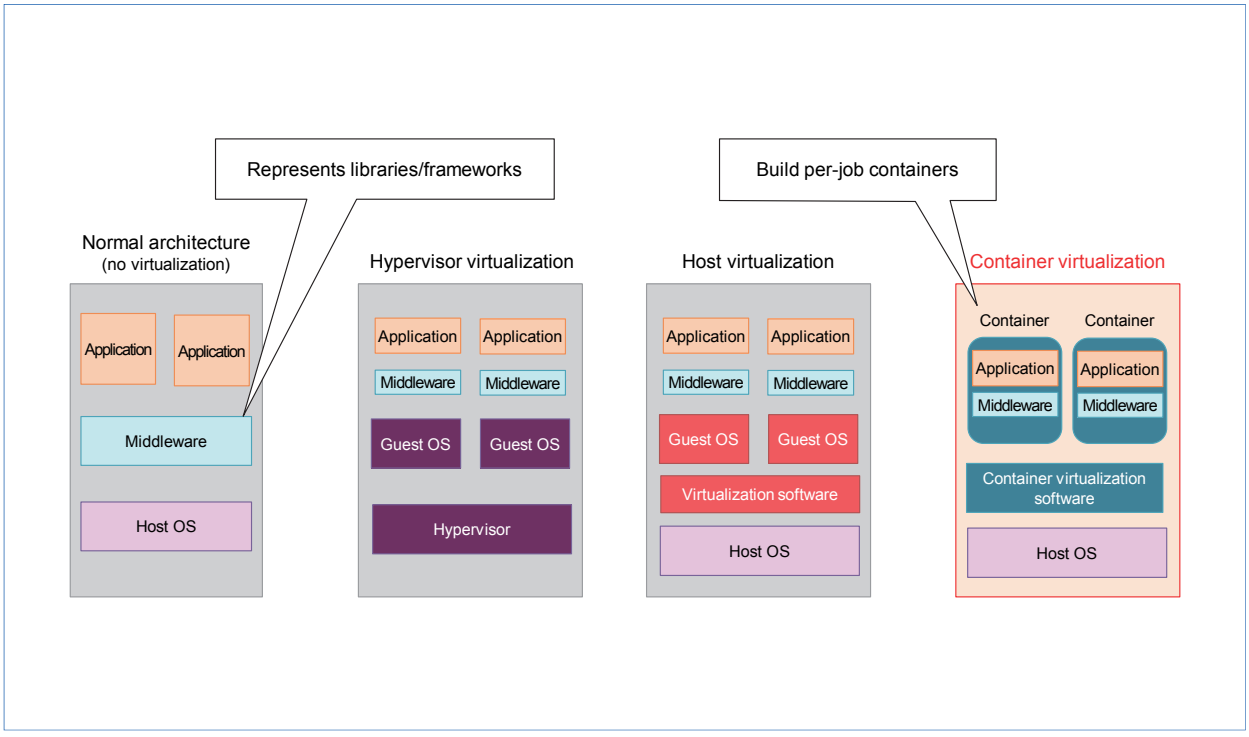

Figure 1 Differences between container virtualization and other virtualization technologies

*12 OSS: Software for which the source code is published free-of-charge, and anyone can re-use or modify it. However, the original author retains the copyright, and even if a derived work is created or it is redistributed, expression of the original author's copyright must be maintained.

*13 Python: A general purpose programming language.

*14 Docker®: Container-type virtualization software. A registered trademark of Docker Inc.

*15 Host virtualization technology: A type of computer virtualization technology. Software that acts as a base for the OS is installed on the server, and virtual machines run on that software.

*16 Linux®: An open-source Unix-type OS that can be freely redistributed under GNU Public License (GPL). A registered trademark or trademark of Linus Torvalds in the United States and other countries.

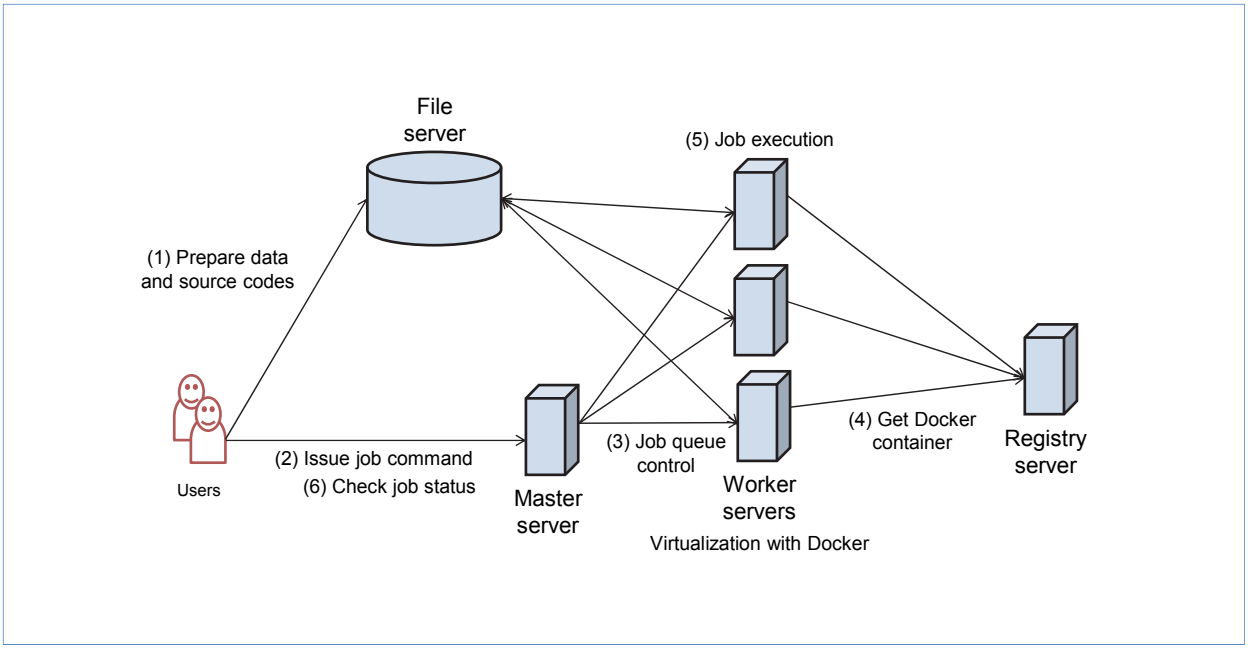

Figure 2 Deep learning platform system architecture

servers, a File server, and a Registry server.

- The Master server manages all deep learning jobs, and all input from users is processed through the Master server.
- Worker servers receive instructions from the Master server and perform learning tasks. Learning tasks are performed in Docker containers in each of the Worker servers. Note that Docker alone does not include functionality to use the GPU on a Worker server, so the Nvidia-docker*18 OSS is used.
- The File server is used to store deep learning data and source code.
- The Registry server is used to store the Docker Registry, which centrally manages the Docker containers. Docker containers with many combinations of frameworks and libraries can be created and stored in the Registry server so that using the platform, users can select any of the deep learning framework and library combinations they need.

Users perform two tasks when using the system: storing deep learning data and source code, and registering a job in the Master server. Job control and execution is performed automatically.

3.3 Job Management

The following functions were developed for the Master server, which manages jobs submitted by users.

(a) Receiving and registering jobs (Fig. 2 (2))
(b) Controlling the job queue (Fig. 2 (3))
(c) Building run-time environments for jobs and executing jobs (Fig. 2 (3)–(5))

*17 Hypervisor virtualization: A type of computer virtualization technology. Virtual machines run installed directly on the server.
*18 Nvidia-docker: Docker extension software.

(d) Monitoring job state (Fig. 2 (6))

We studied two ways to implement the job registration and job control functions: using a message queue*19, and using a Relational DataBase (RDB)*20. Using a message queue yields greater throughput and lower latency than a RDB, but monitoring of job states and processing retries is more difficult to implement. We adopted the RDB method in this case, because job monitoring was important, and since actual job execution is done by Worker servers, throughput and latency of job registration and queue control was less important.

Next we describe specifically how these functions were implemented.

The user prepares the data and source code required for the deep learning task (Fig. 2 (1)). Then, a job command in a pre-determined format is registered on the Master server (Fig. 2 (2)), and a queue entry in the form of an RDB record is created and added in the Master server. Job commands and options were developed for this platform so that the deep learning framework can be selected, and the location of data and source code on the File server can be specified.

For job queue control, it was desirable to only run a single job on one GPU due to GPU memory restrictions. As such, job control was implemented by having the Master server periodically check the status of GPU usage on each Worker server (Fig. 2 (3)), and run the next job on the queue if it is idle (Fig. 2 (3)). For jobs that are ready to run, the allocated Worker server retrieves the specified Docker container from the Registry server (Fig. 2 (4)) and runs the job (Fig. 2 (5)).

The last function is job state monitoring, which the user can do by referring to the RDB table (Fig. 2 (6)). Note that, as mentioned earlier, (b) and (c) are controlled and performed automatically.

4. Results

There are three main results achieved through introduction of this platform.

1) Reduced AI Model Training Time

Previously, when developing AI models, each run and set of learning conditions were run together on a server, as shown in **Figure 3** (a). This resulted in low server utilization and much waste. This platform enables job scheduling to be automated and run in parallel, so that significant reductions in training time are achieved (Fig. 3 (b)).

2) Improved Learning Accuracy

With an increase in efficiency of AI model training, training can be attempted under more conditions, which has achieved great increases in training accuracy (**Figure 4**).

3) Accelerated Environment Building

Use of Docker has made it easy to introduce new deep learning frameworks or updates. In concrete terms, the several hours it previously took to build an environment has been reduced by a factor of ten.

5. Future Prospects

Three items present themselves as future prospects.

*19 Message queue: A mechanism used when exchanging data and cooperating between application software, in which data to be transmitted is temporarily stored without waiting for the other application to process it, and the sending application continues on with its next task.

*20 RDB: A data base management scheme in which data is managed in multiple tables and complex data relationships are handled by defining relationships among tables.

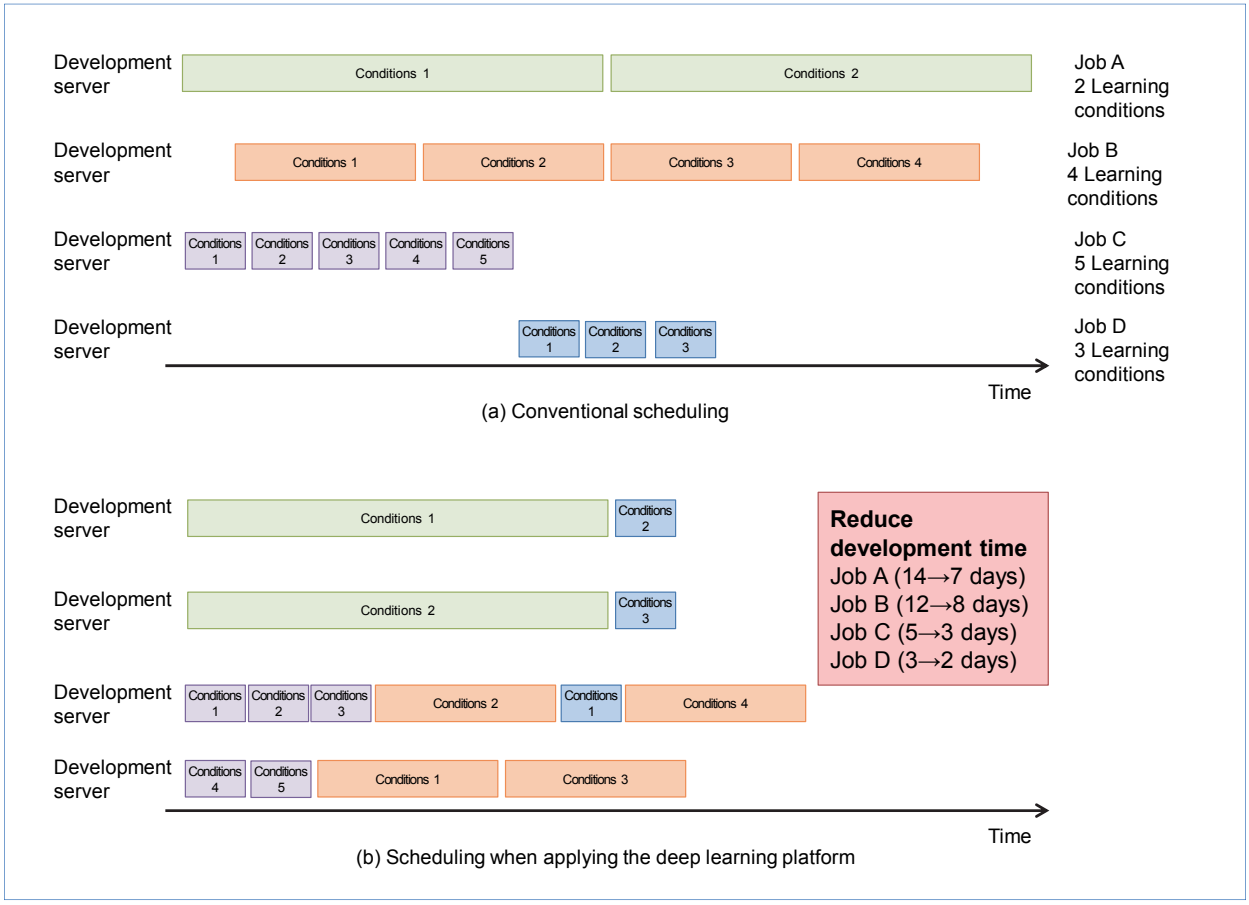

Figure 3 Effects of applying the deep learning platform

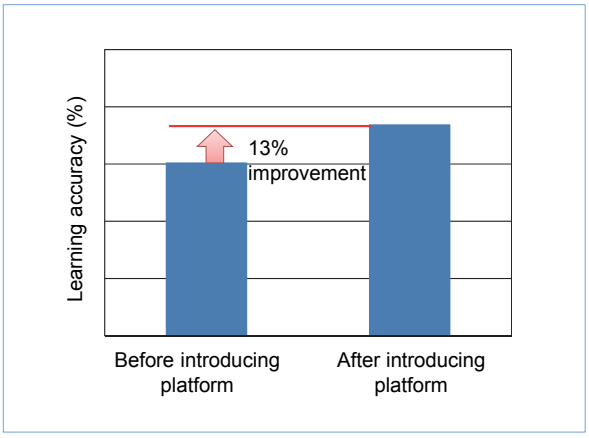

Figure 4 Improvement in accuracy through use of the deep learning platform

1) Support multiple generations of GPU, multiple cloud environments

Generations of GPU advance almost every year, and the size of model that they can handle also changes, so jobs must be distributed to Worker servers with consideration of the GPU generation. Implementation must also be consolidated to handle multiple cloud environments.

2) Upgrade the Scheduler

The current job scheduler can only apply a simple priority to jobs, so this will be improved to allow application of more complex priorities.

3) Optimize Parameters

Current training only allows multiple conditions to be run in parallel, so the more conditions are submitted, the more the servers will be monopolized. By establishing algorithms for selecting optimal combinations of parameters that can efficiently reduce computing time, it will be possible to develop accurate AI models in less time.

6. Conclusion

In this article, we have described technical details of a platform that facilitates deep learning processes for developing accurate AI models in short periods of time. We intend to use this platform to accelerate deep learning initiatives, to advance the platform itself, and to improve efficiencies.

REFERENCES

[1] T. Okatani: "Deep Learning," Kodansha, 2015.
[2] Y. Bengio: "Practical recommendations for gradient-based training of deep architectures," Neural Networks, Springer, pp.437–478, 2012.