Task Completion Time Prediction Scaled by Machine Learning Model Uncertainty

Shumpei Kawaguchi*, Yuichi Ohsita†, Masahisa Kawashima† and Hideyuki Shimonishi†
*Graduate School of Information Science and Technology, Osaka University, Osaka, Japan
Email: s-kawaguchi@ist.osaka-u.ac.jp
†Cybermedia Center, Osaka University, Osaka, Japan
Email: yuichi.ohsita.cmc@osaka-u.ac.jp, kawashima.masahisa.cmc@osaka-u.ac.jp, shimonishi.cmc@osaka-u.ac.jp

***Abstract*—We discuss "GPU Time Sharing," which involves sharing GPU servers locally owned in the private cloud to increase their usage and to reduce the costs of cloud services. In the GPU Time Sharing, users can reserve and use a GPU server for a certain period of time to complete a task, such as training the AI model or inference on a video. In the reservation, task completion time must be predicted conservatively to ensure that the task is completed in the reserved time, but at the same time, the reserved time should be as short as possible for efficient resource sharing. In this paper, we propose a task completion time prediction method called "Uncertainty Scaled Gradient Boosting Decision Tree" (USGBDT), which first predicts the completion time of the Deep Learning (DL) tasks using the Gradient Boosting Decision Tree, and then scales the predicted time based on the expected uncertainty of the machine learning models. Applying the proposed method to the GPU Time Sharing for video analysis tasks, we have confirmed that all tasks are completed in the predicted completion time and the GPU usage time over the reserved time is improved from 53.4% to 67.0%.**

***Index Terms*—Machine Learning, GPU Sharing, Resource Management**

I. Introduction

In recent years, the utilization of AI and machine learning technologies has expanded, and the rapid proliferation of new applications and services using GPUs, especially Generative AIs, LLM (Large Language Model), and foundation models, has dramatically increased the demand for GPUs.

Traditionally, AI models and machine learning models have often been used with GPUs provided by cloud services, resulting in increased financial costs for GPU resources for individual users and university laboratories. Furthermore, as the number of cloud service users increases, there are concerns about the growing power consumption due to the expansion of data centers and the increased network load caused by the large amount of data flowing through the network.

In contrast, corporations, such as IT service provider or large IT solution users, universities and national research institutions often possess high-performance GPU servers for their private cloud, but some of them are not fully utilized, i.e. remain connected to power and network despite only being used for a short amount of time. Therefore, in recent years, the concept of sharing these resources has received more attention. For example, securing computational resources by sharing owned servers has been provided by many universities in the United States [1]. In addition, a new computing paradigm has been proposed in which any computer connected to the Internet can be used as a computational resource in the future [2].

One major problem with sharing private GPU servers in this way is that, since they are deployed by each company for its own business, they do not lend them freely as in the cloud, and the company that owns them must be able to prioritize their use at any time. Therefore, we propose a new service called "GPU Time Sharing," which allows GPU servers in the private cloud to be used by other users in a gap time when the local users have declared they will not use. This is exactly the same as operating as a taxi when not using a private car. By lending out such GPU servers during their non-operational hours, it is possible to effectively utilize the low-utilization GPU servers in the private cloud. This approach can reduce the financial burden on individual laboratories while also achieving a reduction in social energy consumption.

In GPU Time Sharing, local users should be able to use their own GPU servers whenever they want, and thus external users should avoid occupying servers for long periods of time but use them on a task basis, such as a batch to train an AI model or inference on a video. Such fine-grained lending is important to ensure owner convenience, especially in small private clouds. However, if the user's task does not complete within the reserved GPU server time, the task might be forcibly terminated before completion, so that the usage rights of the server are returned to server's owner. To ensure that tasks are completed within the reserved time, it is necessary to predict the task completion time conservatively. However, such prediction leads to a shorter actual usage time of the GPU server relative to the reserved time. To achieve a higher task completion rate within the reserved time and higher GPU usage time, it is necessary to accurately predict the task completion time and select the appropriate GPU server before the user starts executing the task.

In conventional task scheduling, since many tasks are assigned to many GPU servers, a statistical multiplexing effect can be expected, and thus it is not a major problem even if the completion time prediction of individual tasks contains certain errors. However, in the GPU Time Sharing, since the tasks are scheduled in a fine grain, accuracy of the individual completion time prediction is quite important. To predict the completion time using a machine learning model, there is the possibility of underestimating the actual completion time. Therefore, scaling must be applied according to the uncertainty of the machine learning model's predictions to avoid underestimating.

Also, since users are expected to perform various tasks, predicting the completion time of the tasks before executing them is quite difficult unless the user provides detailed information on the application, data, and tasks they intend to use on the provided GPU server. GPU Time Sharing is intended to be provided not only to researchers with deep knowledge of computers but also to general users without such knowledge. Providing detailed information about the characteristics of the source code of the applications they run and the specifications of the AI or machine learning being used would be extremely challenging.

In this paper, we propose a machine learning-based task completion time prediction method called Uncertainty Scaled Gradient Boosting Decision Tree (USGBDT); which considers the uncertainty of the predictions and only uses a small number of easily obtainable input parameters. In completion time prediction using machine learning, the magnitude of prediction error varies depending on the types of deep learning tasks and the training status of the using data. The proposed method adapts to the uncertainty based on these variations while maximizing the actual GPU usage time relative to the GPU server reserved time.

II. Related Work

Several prediction-based methods have been proposed to schedule GPU resources for deep learning tasks. One of the methods is the method called ”Horus” [3]. Horus is the interference-aware and prediction-based scheduling method in deep learning co-location systems. In [3], the relationship between the characteristics of popular deep learning models and GPU utilization is analyzed and characterized based on benchmark tests. These characteristics of the deep learning model include floating point operations (FLOPs), input data size, and the computational graph structure, such as the number of layers. A machine learning model is constructed based on the features of the computational graph such as FLOPs, Batch Size, Memory Parameters to predict GPU utilization. And they construct a scheduler to support co-location considering resource interference risk.

The method for predicting the execution time of deep learning tasks in large-scale GPU datacenters has also been proposed [4]. In this paper, a machine learning model using GBDT is constructed to predict GPU time and cluster status utilizing trace data from deep learning workloads executed in large-scale data center clusters. For the vast number of deep learning tasks running in the cluster, the resource manager is designed to predict execution time based on features such as job type, user characteristics, and cluster type, thereby achieving reductions in overall job completion time and waiting time.

In [5], it is demonstrated that the use of machine learning models to predict the execution time of the GPU application is more effective compared to analytical models. Unlike analytical models, machine learning models can reasonably

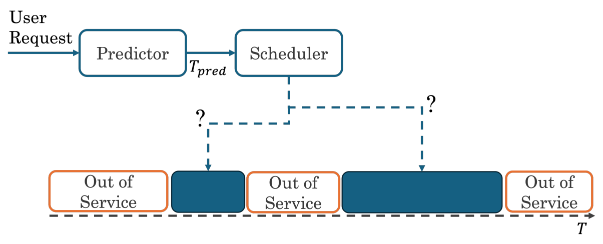

Fig. 1. Task Scheduling Method of the GPU Time Sharing

predict execution time without requiring detailed knowledge of application code, hardware characteristics, or modeling. Therefore, when a large dataset containing similar applications and hardware structures is available, the machine learning approach has been found to be effective in predicting the execution time of GPU applications on heterogeneous architectures utilizing GPUs.

In those previous studies, there have been no studies in which the uncertainty of the predictions made by machine learning models has been corrected during the prediction phase. In [3], resource interference is not considered during the prediction phase, but rather in the task scheduling algorithm to expect statistical multiplexing gains. In GPU Time Sharing, the usage time of GPU servers is limited, and prediction errors for individual deep learning tasks can significantly impact service quality. Therefore, this paper proposes that scaling should be applied according to the uncertainty during the prediction phase of the execution time of the deep learning task.

III. Completion Time Prediction Method

In this section, we provide an overview of GPU Time Sharing and explain the concept of Uncertainty Scaled GBDT.

A. GPU Time Sharing

In the GPU Time Sharing, users can use the GPU servers that are deployed in private clouds. The owner of the GPU server can decide the time periods during which the GPU server will be available for GPU Time Sharing. Users can use the GPU server that the task scheduler selects based on the predictor's prediction of the deep learning task completion time. Figure 1 represents the scheduling of tasks in GPU Time Sharing; the completion time of the task is predicted before running it, and the scheduler schedules the task using the predicted completion time. If the predicted completion time is shorter than the actual completion time, the task may not be completed within the reserved GPU time, leading to the task being interrupted midway. This would result in decreased user satisfaction. However, if the predicted completion time is significantly longer than the actual completion time, the actual GPU usage time will be shorter relative to the reserved time, leading to inefficiencies and wasted resources. Therefore, in the GPU Time Sharing, an accurate completion time prediction is extremely important.

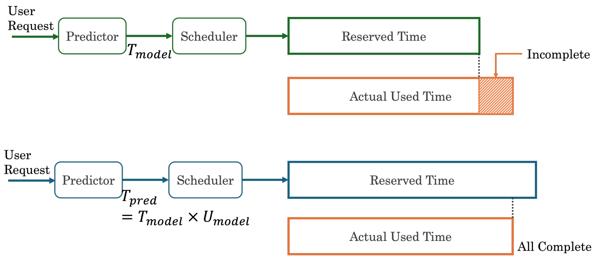

Fig. 2. Scheduling Method using T_{model} and T_{pred}

B. Overview of Uncertainty Scaled GBDT (USGBDT)

Task Scheduler in GPU Time Sharing should satisfy the following requirements.

(I) Ensure that the GPU reserved time does not end before the user's GPU task is completed.
(II) Minimize the difference between the completion time of the user's GPU task and the end of the GPU reserved time.

If requirement (I) is not satisfied, it is possible that the user's task will remain incomplete, which significantly degrades the quality of service. The above requirements (I) and (II) can be achieved by satisfying the conditions (a) and (b) using Δ defined by Eq. (1).

$$\Delta = \frac{T_{true} - T_{pred}}{T_{true}}, \tag{1}$$

(a) $\Delta \leq 0$ is ensured for all data.
(b) Δ is maximized under condition (a)

Figure 2 illustrates the scheduling using the predicted completion time. If condition (a) is not satisfied, incomplete tasks will occur, as shown in Fig. 2. In the proposed method, we first construct a machine learning model using GBDT to predict the completion time of deep learning tasks. However, the predicted completion time from the constructed machine learning model (T_{model}) includes underprediction errors, and using this time directly as the completion time prediction can often fail to satisfy condition (a). Therefore, we scaled the predicted completion time by machine learning model using a coefficient to obtain a completion time prediction using task scheduling. It is a significant challenge to determine this coefficient. If it is too small, condition (a) cannot be satisfied; if it is too large, condition (b) cannot be satisfied. Therefore, in this paper, we propose to vary the coefficient according to the prediction error of the machine learning model. In this method, we obtain the prediction result T_pred by Eq. (2).

$$T_{pred} = U_{model} \times T_{model} \tag{2}$$

In Eq. (2), T_{model} is the completion time predicted by a machine learning model, and T_{true} is the actual completion time. U_{model} is a coefficient that represents the uncertainty in the machine learning model, determined by the parameters input by the user. The selection of this coefficient will be discussed in Section V-E.

TABLE I
CANDIDATE INPUT PARAMETERS

Parameter	Description
G_{core}	Number of the GPU core
G_{boost}	Boost clock frequency of the GPU
G_{FLOPS}	GPU FLOPS
M_{frames}	Number of frames in the video data
M_{height}	Height of the video data
M_{width}	Width of the video data
M_{pixels}	Pixels of the video data
M_{size}	File size of the video data
A_{params}	Number of parameters of the Video Analysis AI
A_{FLOPs}	Video Analysis AI FLOPs
A_{mAP}	Video Analysis AI mAP

IV. Selection of Input Parameters for Predicting Video Analysis Task Completion Time

Task completion prediction requires the information on the application. However, providing detailed information about the characteristics of the applications would be extremely challenging. Therefore, We simplify the parameters input into the predictor and minimize the information users need to provide beforehand to make GPU Time Sharing as user-friendly as possible.

In this paper, we evaluate USGBDT by predicting the completion time of video analysis tasks. In this section, we discuss the selection of input parameters for USGBDT, focusing on the video analysis tasks.

We first list the features that are considered necessary to predict the completion time of video analysis tasks, and use those parameters as inputs to construct the machine learning model. Table I lists the candidate parameters considered as input for the video analysis task completion time predictor.

We select features from the candidates listed in Table I, incrementally increase the number of input parameters, and choose the best combination of parameters to construct the predictor. The best predictor was chosen based on the smallest MAPE (Mean Absolute Percentage Error) calculated between the completion time predicted by the machine learning model T_{model} and the actual completion time T_{true} using test data. In this paper, since the evaluation is conducted using the environment described in V-A, the selection of input parameters was also performed using the environment presented in V-A. Figure 3 shows the MAPE when the predictor is constructed by varying the number of input parameters.

In this paper, we use XGBoost [6] and LightGBM [7] to construct a machine learning model for prediction. We refer to the machine learning model built with XGBoost as the XGB model, and the model built with LightGBM as the LGB model. It was confirmed that MAPE was minimized when the number of input parameters was 4 or 5 in the XGB model and 4, 5 or 6 in the LGB model. Therefore, the completion time predictor proposed in this paper adopts the model with 4 input parameters. The combination of parameters are listed in Table II for the XGB model and in Table III for the LGB model. M_{frames} and M_{height} can be easily extracted from the input video data. In this study, we use some versions of YOLO for video analysis, and Y_{FLOPs} and Y_{params} are properties of YOLO that are publicly available and can be

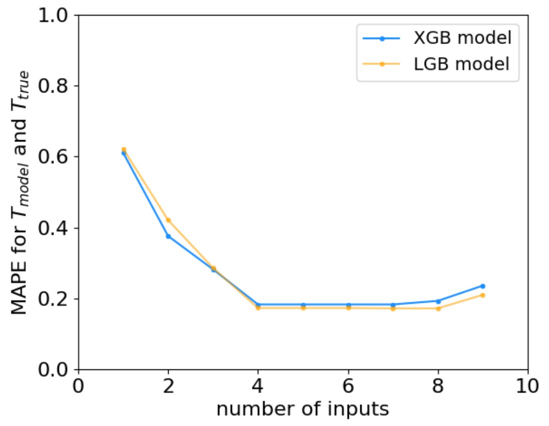

Fig. 3. The Relationship Between the Number of Input Parameters and the Model Score in the Constructed Predictors

TABLE II
INPUT PARAMETERS USED IN THE XGB MODEL

Inputs parameter	Description
G_{FLOPS}	GPU FLOPS
M_{frames}	Number of frames in the video data
M_{height}	Height of the video data
Y_{FLOPs}	YOLO FLOPs

TABLE III
INPUT PARAMETERS USED IN THE LGB MODEL

Inputs parameter	Description
G_{FLOPS}	GPU FLOPS
M_{frames}	Number of frames in the video data
M_{height}	Height of the video data
Y_{params}	Number of parameters of the YOLO

easily obtained. Therefore, it was concluded that the proposed prediction method allows the prediction of the completion time using only parameters that the users can easily provide.

Note that G_{core}, G_{boost}, and G_{FLOPS} are GPU related parameters, which only need to be known by the service provider to select GPU servers, and users do not need to care about them. In this paper, we only tested two types of GPU servers; thus, only G_{FLOPS} is sufficient to discriminate them, but more GPU related parameters might be used for the prediction to integrate various GPU servers into GPU Time Sharing in our future work.

V. EVALUATION AND DISCUSSION

A. Evaluation Environment

In this paper, we verify the effectiveness of USGBDT specifically for object detection tasks among various deep learning tasks. So, we used four versions of YOLO in Tables IV and V to evaluate the USGBDT. We use YOLOX [10] and YOLOv5 [11] to obtain the inference completion time data for training machine learning based prediction model (IV). And for test data, we use YOLOv8 [12] and YOLOv10 [13] (V). We refer to the information from [10]–[13] to obtain YOLO specifications and implement them accordingly.

We used two types of servers with two types of GPUs, as shown in Table VI, as resources for edge computing. Table VII shows the specifications of GPU.

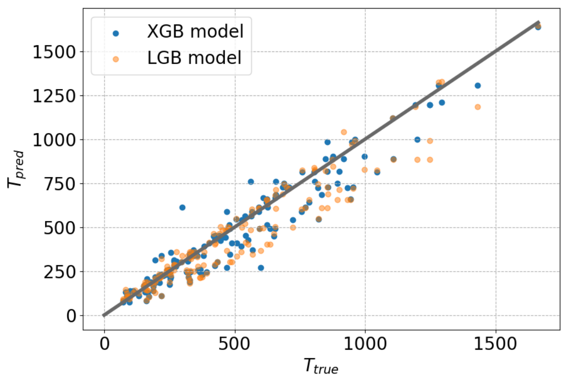

Fig. 4. Distribution of Prediction Errors by the GBDT model

We construct a machine learning based prediction model using XGBoost and LightGBM. We refer to [8], [9] for implementation. Additionally, the models with Uncertainty Scaled applied to each machine learning model are referred to as the USXGB model and the USLGB model.

The videos used for inference with YOLO were created based on the videos obtained from the 7th AI city challenge dataset [14]. Since the dataset obtained from [14] lacked variety in video duration, we created video data with diverse duration by purging and merging together different segments of the videos. Table VIII shows videos used to obtain training data, and Table IX shows videos used to obtain test data.

B. Uncertainty of the GBDT Model

In this section, we evaluate the prediction errors of the XGB model and the LGB model. The x-axis of Figure 4 represents the true completion time, and the y-axis represents the values predicted by the machine learning model. The straight line in Fig. 4 represents the line $y = x$, and any points plotted below this line indicate that the predicted values by the machine learning model are underpredicted. In this evaluation, we found that the XGB model resulted in underestimation in 50.6% of the whole data, while the LGB model exhibited underestimation in 53.8% of the whole data.

Here, to calculate the utilization efficiency of the GPU server in GPU Time Sharing, we present a metric defined by Eq. 3.

$$G_{util} = \begin{cases} \dfrac{T_{true}}{T_{pred}} & (\text{if} \ \ T_{true} \leq T_{pred}) \\ 0 \quad (\text{if} \ \ T_{true} > T_{pred}) \end{cases} \tag{3}$$

TABLE IV
TRAINED YOLO

YOLO Models	params (M)	FLOPS (G)	mAP
YOLOv5x	86.7	205.7	50.7
YOLOv5l	46.5	109.1	49.0
YOLOv5m	21.2	49.0	45.4
YOLOv5s	7.2	16.5	37.4
YOLOX-x	99.1	281.9	51.1
YOLOX-l	54.2	155.6	49.7
YOLOX-m	25.3	73.8	46.9
YOLOX-s	9.0	26.8	40.5

TABLE V
UNTRAINED YOLO

YOLO Models	params (M)	FLOPs (G)	mAP
YOLOv8x	68.2	257.8	53.9
YOLOv8l	43.7	165.2	52.9
YOLOv8m	25.9	78.9	50.2
YOLOv8s	11.2	28.6	44.9
YOLOv10-X	29.5	160.4	54.4
YOLOv10-L	24.4	120.3	53.2
YOLOv10-M	15.4	59.1	51.1
YOLOv10-S	7.2	21.6	46.3

TABLE VI
GPU SERVER USED AS EDGE COMPUTING RESOURCES

	Server1	Server2
CPU	Intel Core i7-13700F	Intel Core i5-13400F
GPU	Nvidia RTX4070	Nvidia GTX1650
RAM	32GB	16GB

TABLE VII
GPU SPECIFICATIONS

GPUs	CUDA Core	Boost Clock (GHz)	FLOPS
NVIDIA GeForce GTX1650	896	1.59	2.849
NVIDIA GeForce RTX4070	5888	248	29.204

In GPU Time Sharing, if a task does not finish within the reserved time due to underprediction of the completion time, the task will be forcibly terminated. Therefore, the GPU utilization for data that was underpredicted is considered 0% in Eq. 3. When scaling is not applied according to the uncertainty of the machine learning model, the prediction made by the model, T_{model}, is used as the final predicted result, T_{pred}. Therefore, in Eq. 3, $T_{pred} = T_{model}$. Calculating GPU utilization from the equation results in 45.0% for the XGB model and 42.9% for the LGB model. This indicates that using predictions from the machine learning models for scheduling in GPU Time Sharing fails to meet both requirements (I) and (II) in Section III.B. Therefore, it is obvious that there is a need to scale the uncertainty of the machine learning models.

C. Dependency on Video Data

Whereas the variation of YOLO versions is countable, users would use their own video data for their inference tasks, and thus variations of input video data are enormous. Therefore, the XGB/LGB models should accurately predict the completion time of the inference for various video data that are completely different from those used for model training.

Figure 5 shows the distribution of the prediction error between T_{model} and T_{true} for both the XGB model and the LGB model. This figure shows that it is clear that many points are plotted near the line $y = x$. This indicates that the predictions made by the machine learning model can achieve high accuracy in unknown video data. Furthermore, data analysis revealed that the percent error is almost within the range of -0.2 to 0.2. This indicates that machine learning models can predict task completion time for unknown video data within a 20% error range. This error range is much smaller than that of the unknown YOLO versions, thus we can

TABLE VIII
TRAINED VIDEO

movies	length	width	height	size (MB)	frames
c001	28:46	1920	1080	823.7	51728
c003	32:17	1920	1080	794.1	58052
c007	05:02	1920	1080	84.7	9051
c008	11:07	1920	1080	75.4	19990
c047	20:01	1920	1080	439.8	35994
c107	10:00	1920	1080	134.2	17982
c001s	28:46	213	120	39.3	51728
c003s	32:17	213	120	36.8	58052
c007s	05:02	426	240	9.9	9051
c008s	11:07	160	120	2.6	19990
c047s	20:01	320	240	14.2	35994
c107s	10:00	960	720	53.6	17982

TABLE IX
UNTRAINED VIDEO

movies	length	width	height	size (MB)	frames
c005	30:29	1920	1080	507.1	54815
c014	10:00	1920	1080	96.2	17982
c048	20:01	1920	1080	346	35994
c006s	30:59	213	120	36.7	55714
c015s	10:00	320	240	9.4	17982

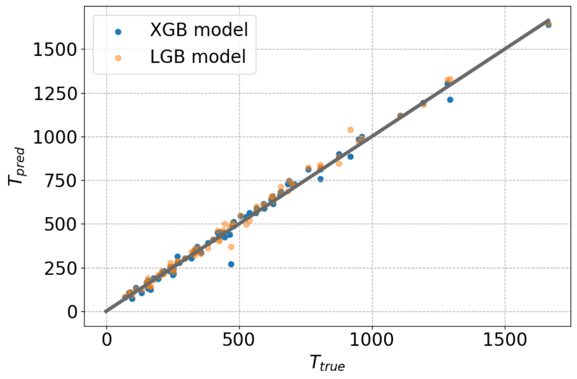

Fig. 5. Distribution of Prediction Error (Trained YOLO and Untrained Video)

conclude that the predictor is capable of handling various input video data accurately. Furthermore, since the value of U_{model} that satisfies condition (a) in III.B is expected to be small, it is anticipated that condition (b) can be satisfied sufficiently. It may be possible to further improve the accuracy by training the model with more input data, which would be our future work.

D. Dependency on YOLO Versions

There are a number of variations YOLO versions as introduced above, and we can assume that many versions of YOLO are learnt into the machine learning model and many users would use these YOLO versions. We can also assume that new YOLO versions will be proposed in the future, or variations of existing YOLO versions are prepared, thus users would also use YOLO versions that are not used for machine learning model training. Note that we assume that some parameters, i.e. YOLO FLOPs or number of parameters of the YOLO, as discussed in the previous section, should be known in advance even for the untrained YOLO versions.

Figure 6 shows the distribution of the prediction error between T_{model} and T_{true} for both the XGB model and the LGB model.

This figure shows that many points are plotted below the line $y = x$. Data analysis revealed that an underprediction occurred in 73.8% of the data for the XGB model and 81.3% for the LGB model. This indicates that the predictions made by the machine learning models have a significant uncertainty when applied to unknown YOLO versions. Also, further analysis showed that the percent error for the LGB model falls within the range of -0.5 to 0.5, while the percent error for the XGB model falls within the range of -1.2 to 0.5. This suggests that the XGB model has the potential to significantly overprediction the completion time. By comparing Figs. 5 and 6, the predictor can produce larger errors for untrained YOLO models. This indicates that there are challenges in generalizability to different YOLO models. Furthermore, the value of U_{model} that satisfies condition (a) in III.B is expected to be higher compared to that for trained YOLO. Therefore, if the U_{model} value that satisfies condition (a) for unknown YOLO is uniformly applied to all data, it is anticipated that condition (b) may not be satisfied adequately. It is significant to adjust the value of U_{model} based on the training status of the YOLO version to scale the uncertainty of the machine learning model accordingly.

Table X shows the value U_{model} required to ensure $\Delta \leq 0$, which means that the predicted task completion time will never be shorter than the actual completion time. To meet this stringent requirement, a large U_{model} value is needed, as expected. As discussed above, the predictor can produce larger errors for untrained YOLO versions, thus larger U_{model} values are required. Comparing the XGB and LGB models, the LGB model has higher accuracy and requires smaller U_{model} values.

E. Analysis on U_{model} Values

Figure 7 shows the values of U_{model} when the allowable value of Δ is varied. In this figure, "T" represents Trained and "UT" represents Untrained. When using common U_{model} to calculate T_{pred}, $\Delta \leq 0$ must be satisfied for all data to achieve 0% task incomplete rate. Higher U_{model} values, which means reserving the completion time larger than the machine

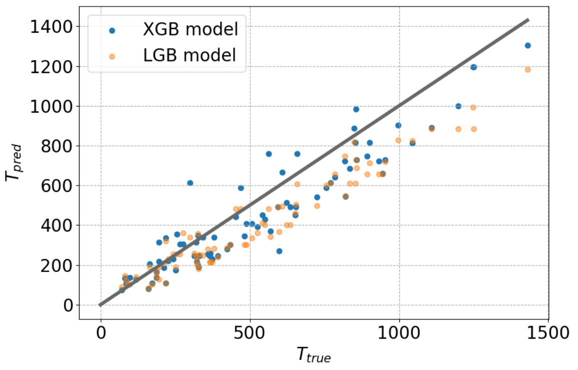

Fig. 6. Distribution of Prediction Error (Untrained YOLO and Untrained Video)

TABLE X
U_{model} VALUES WITH EACH COMBINATION

	U_{model}
USXGB model (T YOLO and UT Video)	1.75
USXGB model (UT YOLO and UT Video)	2.23
USLGB model (T YOLO and UT Video)	1.28
USLGB model (UT YOLO and UT Video)	2.02

learning model predicts, decrease the rate of task incompletion. However, as shown in Eq. (2), since U_{model} is uniformly applied to all predicted values, if U_{model} is too large, the prediction accuracy decreases. We select the smallest U_{model} that satisfies $\Delta \leq 0$ for all data, the value was 2.23 for the XGB Model and 2.02 for the LGB Model.

Figure 8 shows the relationship between task incomplete rate and U_{model} values. The figure indicates the trade off between these values. When we set U_{model} to 1, the task incomplete rate can be as large as 0.2 to 0.8. We confirmed that when U_{model} is set to 2.23 for the XGB Model and 2.02 for the LGB Model, the task incompletion rate becomes 0%.

F. Differentiation of U_{model} values

The analysis on the previous subsection indicates that, if the YOLO version of a user task is indicated and whether the YOLO has been learnt by the machine learning model or not is known in advance, and actually this is true, we can select appropriate U_{model} value to efficiently predict the task completion time T_{pred}. Therefore, in this paper, we also propose a scheme to use differentiated U_{model} values for each task.

We evaluated the predicted completion times (T_{pred}) against the actual completion times (T_{true}) using MAPE and U_{model}, which satisfies $\Delta \leq 0$. Table X shows the values of U_{model} for each combination.

We note that before applying USGBDT, the GPU utilization calculated using Eq. 3 was 45.0% for the XGB model and 42.9% for the LGB model. However, after applying USGBDT, Table XI shows that GPU utilization improved to 52.7% for the XGB model and 67.0% for the LGB model while achieving 0% task incomplete rate caused by underprediction. It is also evident that when U_{model} is determined for each combination, the MAPE is smaller, and the ratio of GPU utilization is higher. Especially, in the completion time prediction based LGB model, while achieving a 100% task completion rate, the

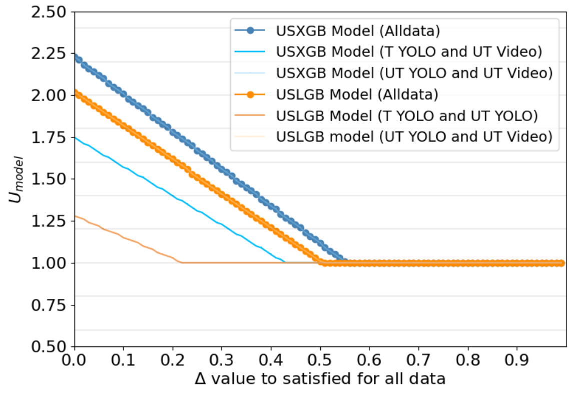

Fig. 7. The Relationship Between U_{model} and Δ in Eq. 1

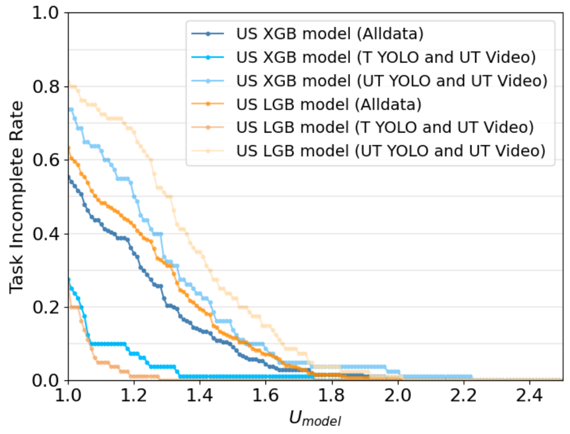

Fig. 8. The coefficient U_{model} of the predicted completion time in Eq. 2 and the rate of tasks not completed within the allotted time.

TABLE XI
MAPE FOR T_{pred} AND T_{true}

	MAPE	GPU Utilization
XGB model (without scaling)	15.5%	45.0%
USXGB model (Common U_{model})	114.0%	46.7%
USXGB model (Diferentiated U_{model})	89.8%	52.7%
LGB model (without scaling)	15.5%	42.9%
USLGB model (Common U_{model})	87.2%	53.4%
USLGB model (Diferentiated U_{model})	49.0%	**67.0%**

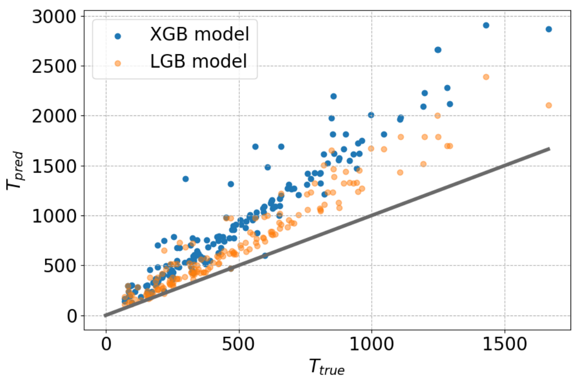

Fig. 9. Distribution of Prediction Error (Using USGBDT and Differentiated U_{model} Values)

ratio of GPU usage time to GPU reservation time is 67.0%. This result demonstrates that USGBDT using differentiated U_{model} improves task completion rate and GPU utilization efficiency in GPU Time Sharing.

Figure 9 shows that neither the XGB model nor the LGB model results in underprediction. The use of USGBDT has been shown to be an effective prediction method that can suppress incomplete tasks and improve GPU server utilization efficiency in GPU Time Sharing. However, the figure shows that the completion time for data with longer duration is overestimated. This is likely due to the application of a constant U_{model} across all data. By considering the scaling of uncertainty more deeply, it is believed that this could lead to the development of a more robust and efficient prediction method for GPU Time Sharing. However, this research will be left for future work.

VI. Conclusion

In this paper, we proposed a completion time prediction method, called USGBDT for deep learning tasks, intended for use in GPU Time Sharing. The proposed method minimizes underprediction by scaling the uncertainty in predictions made by the machine learning model, enabling predictions that improve task completion rates and GPU utilization in GPU Time Sharing.

In the evaluation of the proposed method, a completion time prediction for object detection using YOLO by USGBDT based on the LGB model achieved a MAPE of 49.2%. This indicates that by using the USGBDT LGB model-based completion time prediction for GPU server reservations, a 100% task completion rate can be achieved, while the ratio of actual GPU server usage time to GPU server reservation time can be maintained at 67.0%. Compared to before using USGBDT, we achieved an improvement in task completion rate by 46.2% and increased GPU utilization by 24.1% in GPU Time Sharing.

In GPU Time Sharing, it is expected that object detection using YOLO, as well as various other applications, will be used. Additionally, the system infrastructure for GPU Time Sharing is expected to use various GPUs. Therefore, in our future work we will construct a more generalized completion time model that can accommodate common GPU applications using various GPUs.

References

[1] "Nautilus by national research platform," https:// nationalresearchplatform.org/nautilus/. Accessed: June 10, 2024.
[2] X. Cheng, M. Xu, R. Pan, D. Yu, C. Wang, X. Xiao, and W. Lyu, "Meta computing," IEEE Network, pp.1–7, 2023.
[3] G. Yeung, D. Borowiec, R. Yang, A. Friday, R. Harper, and P. Garraghan, "Horus: Interference-aware and prediction-based scheduling in deep learning systems," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 1, pp. 88–100, 2021.
[4] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–15, 2021.
[5] M. Amaris Gonzalez, M. Dyab, D. Trystram, R. Camargo, and A. Goldman, "A comparison of gpu execution time prediction using machine learning and analytical modeling," 11 2016.
[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp.785–794, 2016.
[7] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," Advances in Neural Information Processing Systems, vol.30, Curran Associates, Inc., 2017.
[8] "XGBoost Documentation", https://xgboost.readthedocs.io/en/stable/index.html. Accessed: May 15, 2024.
[9] "LightGBM's documentation", https://lightgbm.readthedocs.io/en/latest/ index.html, Accessed: May 17, 2024.
[10] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021.
[11] "Ultralytics yolov5," https://github.com/ultralytics/ yolov5. Accessed: May 10, 2024.
[12] "Ultralytics yolov8," https://github.com/ultralytics/ultralytics?tab=readme-ov-file. Accessed: May 10, 2024.
[13] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," 2024.
[14] M. Naphade, et al., "The 7th AI City Challenge," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2023.