Proceedings of the 3rd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2025.26376

GPU Computing Resource Allocation and Prediction Based on Machine Learning Algorithms

Liao Hu

University of Illinois at Chicago, Chicago, USA
lhu231@my.trine.edu

Abstract. This article proposes the BiGRU algorithm optimized by Transformer, providing a more reliable technical path for the fine-grained allocation of GPU computing resources in deep learning models. In the study, ExtraTrees, CatBoost, XGBoost, and LightGBM were selected as comparison models. The results showed that the proposed Transformer BiGRU model performed the best in deep learning GPU resource regression prediction. Among them, the MSE of this model is 2.208, significantly lower than ExtraTrees' 4.414, CatBoost's 6.71, XGBoost's 4.464, and LightGBM's 2.942; The RMSE is 1.486, which is lower than the 2.101, 2.59, 2.113, and 1.715 of other models; MAE is at its lowest level at 0.969, which is better than XGBoost's 1.442 and LightGBM's 1.296; MAPE is 26.938, which is lower than ExtraTrees' 34.478, CatBoost's 37.379, and XGBoost's 36.679; R ^{2} reaches 0.855, higher than LightGBM's 0.803 and XGBoost's 0.69. This indicates that the model has smaller prediction errors and better fitting effects on GPU resources. Its research value lies in laying an important foundation for improving the accuracy and scientificity of GPU computing resource allocation, and has a positive significance in promoting the efficient utilization of resources in the field of deep learning.

Keywords: Transformer, BiGRU algorithm, GPU computing resource prediction.

1. Introduction

With the rapid development of large-scale model training and industrial intelligence, the global demand for computing power is showing explosive growth [1]. The parameter scale of large models represented by GPT series, LLaMA, etc. has jumped from tens of billions to trillions, and the GPU time required for a single round of training can reach millions of hours. In industrial scenarios such as intelligent manufacturing and autonomous driving, the dynamic demand for computing power for real-time inference tasks presents high-frequency wave characteristics [2]. As the core carrier of deep learning computing power, the allocation efficiency of GPU resources directly restricts the process of technology implementation. Under the current static allocation mode, GPU utilization is generally below 30%, and in some scenarios, there is even a contradiction between "idle computing power" and "task queuing", which not only causes a great waste of hardware resources, but also prolongs the model training cycle by more than 30% and increases energy consumption costs by about 40% [3]. The mismatch between computing power supply and demand has become a core bottleneck restricting the large-scale application of AI technology, and there is an urgent need to achieve dynamic optimization of resource allocation through intelligent means.

Machine learning algorithms provide key technical support for solving the problem of GPU computing power allocation. Traditional rule-based allocation strategies are difficult to capture the nonlinear characteristics of computing power demand, while machine learning models can construct a mapping relationship between computing power demand and task attributes through deep mining of historical task data (such as model type, batch size, iteration times, hardware load, etc.). For example, the random forest algorithm can identify key factors that affect computing power demand through feature importance assessment, while the LSTM model can capture temporal dependencies during task execution and achieve short-term computing power fluctuation prediction [4]. The application of these algorithms shifts resource allocation from "experience driven" to "data-driven", which can increase GPU utilization to over 60% and shorten training cycles by 20% -50% in typical scenarios. Through real-time prediction of computing power demand curves of different tasks, the system can schedule resources in advance, avoid computing power redundancy or gaps, provide accurate decision-making basis for dynamic allocation, and reduce the probability of task blocking caused by resource contention [5].

Although existing machine learning algorithms have achieved certain results in computing power prediction, there is still a problem of insufficient prediction accuracy in the face of long-term dependencies and sudden demands in complex scenarios. The LSTM model exhibits significant memory decay on long sequence data, with prediction errors exceeding 25% in multi task concurrent scenarios; Traditional temporal models are difficult to effectively extract the global correlation between task features and computing power requirements [6]. To this end, this article proposes a Transformer optimized BiGRU algorithm, which enhances the ability to capture long-range dependencies between task attributes and computing power requirements by introducing the self attention mechanism of Transformer. At the same time, the bidirectional recursive structure of BiGRU is utilized to improve the efficiency of extracting local temporal features. This fusion architecture not only retains the sensitivity of temporal models to dynamic changes, but also has the ability to model global features, providing a more reliable technical path for achieving fine-grained allocation of GPU computing resources.

2. Data sources

This dataset contains 517 rows and 12 columns of data, with a total of 12 features. Some of the content of the dataset is shown in Table 1. This dataset is related to the GPU computing power requirements of deep learning models. The data covers information such as model type, model parameter size (in millions), batch size, input dimension, number of layers, iteration count, accuracy, optimizer type, whether regularization is used, whether data augmentation is used, learning rate, and GPU requirements (in gflops). The model types include RNN, CNN, ResNet, Transformer, etc., with accuracies of FP16, FP32, and optimizers including Adam, AdamW, RMSprop, etc. These data can be used to construct machine learning regression models to predict the computing power requirements of GPUs, in order to support related research and applications such as GPU computing resource allocation for deep learning tasks.

Table 1. Selected partial dataset

Model type	Parameters million	Batch size	Input dimension	Num layers	Iterations	Precision	Optimizer	Learning rate	Gpu demand gflops
RNN	29.89	1472	409	13	124.83	FP16	Adam	0.002335	1.89762
CNN	17.98	201	619	39	150.33	FP32	AdamW	0.008076	1.88104
ResNet	33.72	854	630	15	336.83	FP32	AdamW	0.0006	9.89094
Transformer	67.93	722	660	29	231.84	FP32	RMSprop	0.008989	10
Transformer	16.65	1423	905	29	1608.08	FP16	Adam	0.006511	10

3. Method

3.1. Transformer

Transformer is a sequence modeling architecture based on self attention mechanism, and its core breakthrough lies in breaking away from the sequence dependency of models such as RNN and achieving parallel processing. Its structure consists of an encoder and a decoder: the encoder receives an input sequence and converts it into context aware feature vectors through N stacked encoding layers; The decoder generates the final output based on the encoder output and its own generated sequence [7]. The network structure of Transformer is shown in Figure 1.

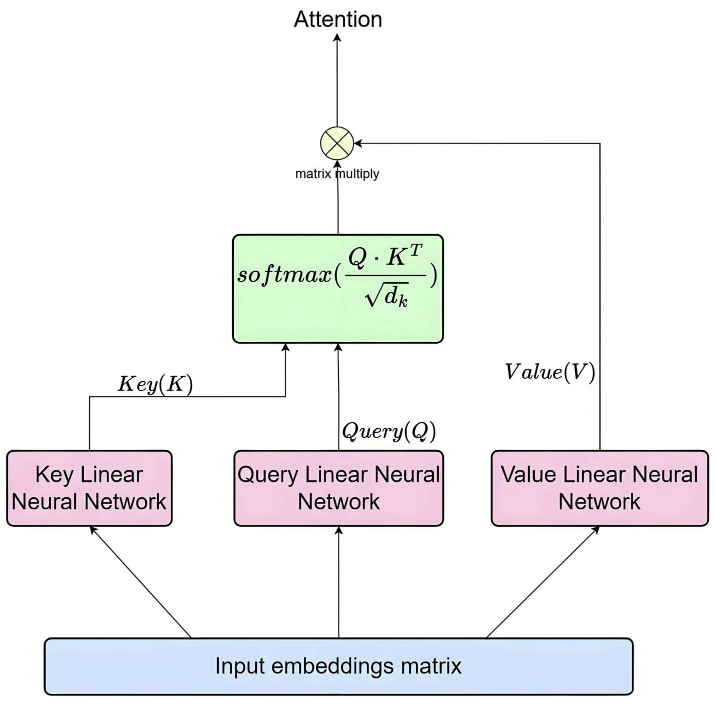
Attention
matrix multiply
$softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})$
$Key(K)$
$Query(Q)$
$Value(V)$
Key Linear Neural Network
Query Linear Neural Network
Value Linear Neural Network
Input embeddings matrix

Figure 1. The network structure of Transformer

Each encoding layer includes a multi head self attention mechanism and a feedforward neural network: multi head self attention maps the input to multiple sets of query (Q), key (K), and value (V) matrices, calculates the correlation weights of tokens at different positions, and then concatenates multiple sets of results to capture multidimensional dependencies; The feedforward network independently performs linear transformations and nonlinear activations on each token. The decoding layer additionally includes encoder decoder attention, allowing the generation process to focus on key information of the input sequence. To preserve sequence order, Transformer introduces positional encoding, injecting positional information into tokens at different positions through sine and cosine functions, solving the problem of self attention being insensitive to order. Compared to the serial mode of RNN word by word processing, Transformer can parallel compute the attention of the entire sequence, greatly improving training efficiency. This architecture, with its ability to capture long-range dependencies and parallelism, has become the foundation of pre trained models such as BERT and GPT, reshaping the technical paradigm of NLP and even multimodal tasks [8].

3.2. BiGRU

The core foundation of BiGRU is GRU, which stands for Gated Recurrent Unit. The network structure of BiGRU is shown in Figure 2. GRU is designed to solve the problem of gradient vanishing or exploding in traditional recurrent neural networks when processing long sequences. Its key lies in flexibly controlling the transmission and forgetting of information through gating mechanisms. Among them, the reset gate is used to determine whether to ignore historical information. When the reset gate value is low, the model will pay more attention to the current input; The update gate determines how much historical information is retained and how much new information is included, similar to the memory update mechanism [9]. This design enables GRU to effectively capture long-term dependencies in sequences, while simplifying the structure of recurrent neural networks and reducing the number of parameters. BiGRU introduces a bidirectional mechanism based on GRU, consisting of two GRUs with opposite directions. A GRU processes data in the original sequence from left to right, capturing information from the past to the present; The other is processed in reverse order from right to left, capturing information from the future to the present. In each processing step, two GRUs will generate corresponding outputs separately, and then combine the results of the two through concatenation or fusion to form a comprehensive feature containing bidirectional context. This bidirectional design allows the model to simultaneously utilize all the information before and after a certain position in the sequence, breaking through the limitation of one-way models that can only rely on historical information [10].

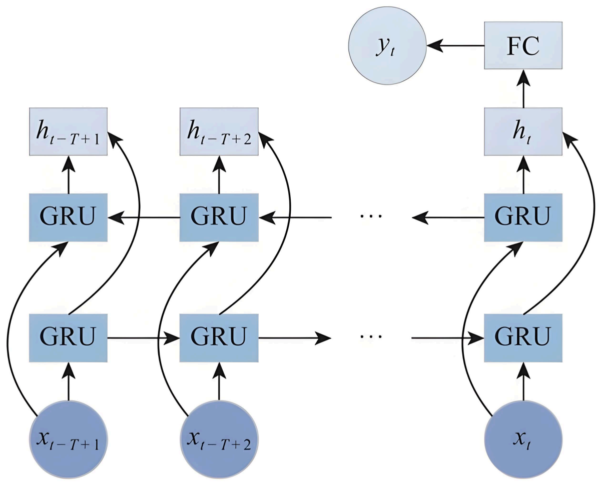

Figure 2. The network structure of BiGRU

The advantage of BiGRU lies in balancing the comprehensiveness of information capture with computational efficiency. Compared to unidirectional GRU, it can more accurately understand contextual information, and compared to more complex bidirectional LSTM structures, it has fewer parameters and faster training speed, making it more practical in resource limited scenarios.

3.3. Transformer-BiGRU

The principle of optimizing BiGRU with Transformer is mainly based on compensating for the inherent shortcomings of BiGRU in long sequence modeling and parallel computing, while retaining its local temporal perception advantages. As a bidirectional gated recurrent unit, BiGRU captures sequence dependencies through forward and backward hidden layers. However, the dependency loop structure requires computation to be performed sequentially in time steps, resulting in poor parallelism and low training efficiency. Additionally, information decay is prone to occur in long sequences, making it difficult to model long-range distance correlations. Transformer breaks this limitation through its self attention mechanism: its core self attention module can directly calculate the associated weights of any two positions in the sequence, and process all temporal positions in parallel through matrix operations, effectively capturing global dependencies and solving the problem of long-distance information loss; The multi head attention mechanism further maps features to multiple subspaces, learns association patterns from different scales, and enriches the dimensionality of feature representation.

In optimization practice, this article adopts a hybrid architecture: BiGRU is used to process local short-term temporal features, and the output is used as the input of the Transformer. The global dependencies are modeled through the self attention layer of the Transformer, while residual connections and layer normalization are combined to enhance training stability. This not only preserves BiGRU's sensitive capture of local context, but also utilizes the global modeling capability and parallel efficiency of the Transformer to improve overall sequence modeling performance.

4. Result

In terms of experimental parameters, the dataset is divided into training and testing sets in a 7:3 ratio, and the input feature dimension is determined by the number of features in the dataset (output dimension is 1). The model includes a sequence input layer (with the number of channels equal to the number of features), a position embedding layer (with a maximum position encoding of 512), 2 self attention layers (4 heads, 32 key channels per head), a forward GRU layer (6 units), a reverse GRU layer (10 units, input after flipping the data by FlipLayer), as well as ReLU activation layer, 0.01 dropout layer, fully connected layer (output dimension 1), and regression layer. The training uses Adam optimizer with a maximum round size of 200, batch size of 256, initial learning rate of 0.01 (reduced to 0.001 after 80 rounds), L2 regularization coefficient of 0.001, and gradient clipping threshold of 10,. In terms of software and hardware configuration, the software relies on MATLAB and its deep learning toolbox. The hardware is equipped with NVIDIA GPU that supports CUDA to accelerate training, and the CPU is a multi-core processor. The specific parameter settings are shown in Table 2.

Table 2. The specific parameter settings

Aameter Type	Specific Parameters
Dataset split	70% training, 30% testing
Input/Output dims	Input dim determined by dataset, output dim 1
Position encoding	Max position encoding 512
Self-attention layers	2 layers, 4 heads, 32 key channels per head
GRU layers	6 forward units, 10 backward units
Other layers	ReLU activation, dropout rate 0.01
Optimizer	Adam
Training epochs	Max 200 epochs
Batch size	256
Learning rate	Initial 0.01, 0.001 after 80 epochs
Regularization	L2 coefficient 0.001
Gradient clipping	Threshold 10

In terms of comparative models, this article selects ExtraTrees, CatBoost, XGBoost, and LightGBM. In terms of model evaluation parameters, this article selects MSE, RMSE, MAE, MAPE, and R2.

Firstly, output the comparative results of ExtraTrees, CatBoost, XGBoost, LightGBM, and the proposed Transformer BiGRU on deep learning GPU computing power prediction tasks. The bar chart comparing the indicators of ExtraTrees, CatBoost, XGBoost, LightGBM, and Transformer BiGRU is shown in Figure 3, and the specific comparison results of the indicators are shown in Table 3.

Table 3. The specific comparison results of the indicators

Model	MSE	RMSE	MAE	MAPE	R^2
ExtraTrees	4.414	2.101	1.654	34.478	0.68
CatBoost	6.71	2.59	2.142	37.379	0.568
XGBoost	4.464	2.113	1.442	36.679	0.69
LightGBM	2.942	1.715	1.296	432.656	0.803
Transformer-BiGRU	2.208	1.486	0.969	26.938	0.855

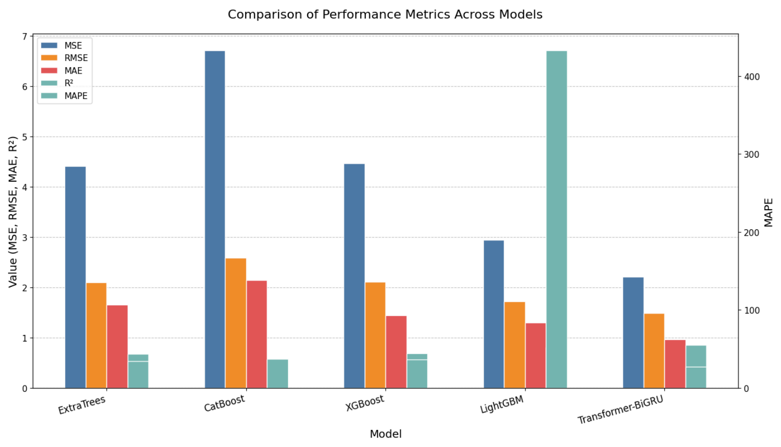

Figure 3. The bar chart comparing the indicators of ExtraTrees, CatBoost, XGBoost, LightGBM, and Transformer BiGRU

From various evaluation indicators, the Transformer BiGRU model proposed in this article performs the best in deep learning GPU resource regression prediction. Its MSE is 2.208, significantly lower than ExtraTrees' 4.414, CatBoost's 6.71, XGBoost's 4.464, and LightGBM's 2.942; The RMSE is 1.486, which is lower than the 2.101, 2.59, 2.113, and 1.715 of other models; MAE is at its lowest level at 0.969, which is better than XGBoost's 1.442 and LightGBM's 1.296; MAPE is 26.938, which is lower than ExtraTrees' 34.478, CatBoost's 37.379, and XGBoost's 36.679; The R ² reached 0.855, higher than LightGBM's 0.803 and XGBoost's 0.69, indicating that the model has smaller prediction errors for GPU resources and better fitting effects.

Output the predicted actual value curve of the Transformer BiGRU test set, as shown in Figure 4. From the predicted actual value curve, it can be seen that Transformer BiGRU's predicted values for deep learning GPU resources are very close to the actual values, which proves the predictive performance of the model.

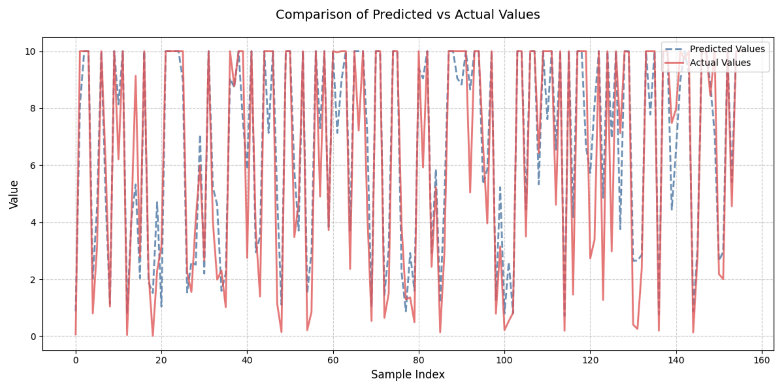

Figure 4. The predicted actual value curve of the Transformer BiGRU test set

The Transformer optimized BiGRU algorithm proposed in this article lays a solid technical foundation for the precise allocation of GPU computing resources in deep learning models. Through comparative experiments with ExtraTrees, CatBoost, XGBoost, and LightGBM models, it can be seen that the Transformer BiGRU model exhibits the best performance in deep learning GPU resource regression prediction tasks. Specifically, its mean square error (MSE) is 2.208, significantly lower than ExtraTrees' 4.414, CatBoost's 6.71, XGBoost's 4.464, and LightGBM's 2.942; The root mean square error (RMSE) is 1.486, which is better than the 2.101, 2.59, 2.113, and 1.715 of other models; The Mean Absolute Error (MAE) is at its lowest level at 0.969, ahead of XGBoost's 1.442 and LightGBM's 1.296; The mean absolute percentage error (MAPE) is 26.938, which is less than 34.478 for ExtraTrees, 37.379 for CatBoost, and 36.679 for XGBoost; The coefficient of determination (R^2) reaches 0.855, higher than LightGBM's 0.803 and XGBoost's 0.69. These data fully demonstrate that the model has a smaller prediction error for GPU resources and a better fitting effect with actual data.

The significance of this research achievement lies in the fact that the Transformer BiGRU model not only provides an efficient solution for accurate prediction of GPU computing power resources in deep learning scenarios, but also lays a key foundation for dynamic scheduling and refined allocation of GPU resources by improving the accuracy of resource prediction. It helps to reduce the waste of computing power resources, improve resource utilization efficiency, and provide more solid computing power support for the large-scale application of deep learning technology in various industries.

References

[1] Li, Ming, et al. "Deep learning and machine learning with gpgpu and cuda: Unlocking the power of parallel computing." arxiv preprint arxiv: 2410.05686 (2024).
[2] Wu, Gene, et al. "GPGPU performance and power estimation using machine learning." 2015 IEEE 21st international symposium on high performance computer architecture (HPCA). IEEE, 2015.
[3] Lin, Yiyu, et al. "GPU-Optimized Image Processing and Generation Based on Deep Learning and Computer Vision." Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023 5.1 (2024): 39-49.
[4] Dhanasekaran, S., et al. "Utilizing cloud computing for distributed training of deep learning models." 2024 Second International Conference on Data Science and Information System (ICDSIS). IEEE, 2024.
[5] Shank, Daniel B., et al. "AI composer bias: Listeners like music less when they think it was composed by an AI." Journal of Experimental Psychology: Applied 29.3 (2023): 676.
[6] Yang, Tiancheng, and Shah Nazir. "A comprehensive overview of AI-enabled music classification and its influence in games." Soft Computing 26.16 (2022): 7679-7693.
[7] Tchemeube, Renaud Bougueng, et al. "Evaluating human-AI interaction via usability, user experience and acceptance measures for MMM-c: A creative AI system for music composition." arXiv preprint arXiv: 2504.14071 (2025).
[8] Qiusi, Mao. "Research on the improvement method of music education level under the background of AI technology." Mobile information systems 2022.1 (2022): 7616619.
[9] El Ardeliya, Vina, Joshua Taylor, and Julia Wolfson. "Exploration of artificial intelligence in creative fields: Generative art, music, and design." International Journal of Cyber and IT Service Management 4.1 (2024): 40-46.
[10] Zhao, Hanbing, et al. "AI-driven music composition: Melody generation using Recurrent Neural Networks and Variational Autoencoders." Alexandria Engineering Journal 120 (2025): 258-270.