Large-Scale Data Processing and Machine Learning Analysis Model Based on Distributed Algorithm

Manfei Lo
BASIS International School PLH, Huizhou, China
manfei.lo40211-biph@basischina.com

Abstract. The model of large-scale data processing and ML(machine learning) analysis based on DA(distributed algorithm) is a powerful computing method, which aims at processing huge data sets and performing efficient ML analysis. In this paper, a cluster topology driver module based on gradient switching and aggregate communication is designed, and its core goal is to adapt the distributed system to various underlying network topologies. By designing decentralized gradient exchange algorithm and aggregate communication framework, the parallel transmission ability of multi-interface network can be fully exerted, thus improving the model synchronization efficiency of ML task. The experimental results show that the cluster topology driver module can provide better performance than the existing methods in terms of training convergence, cluster scalability and communication overhead. Large-scale data processing and ML analysis model based on DA is widely used in processing massive data and realizing complex analysis tasks.

Keywords: Machine Learning; Large-Scale Data; Distributed Algorithm.

1. Introduction

In the information age, the generation and accumulation of large-scale data has become a phenomenon, which comes from various fields, including social media, sensor technology, financial transactions, medical care and so on. These massive data contain infinite value, but they also bring unprecedented challenges. To extract useful information, insight and knowledge from these data, efficient data processing and analysis methods are needed [1]. Large-scale data processing and ML(machine learning) analysis model based on DA(distributed algorithm) came into being and became an important tool to solve this challenge.

DA is a computing method that can effectively process large-scale data. By dispersing data on multiple computing nodes and performing parallel computing, the efficiency and speed of data processing can be greatly improved [2-3]. This method is widely used in the fields of cloud computing, big data analysis and artificial intelligence, which is helpful to solve the calculation bottleneck problem in data processing and analysis. At the same time, ML, as a powerful tool, can help us to mine patterns from data, build prediction models and make automatic decisions [4]. However, when dealing with large-scale data, the traditional ML method faces the problem of insufficient computing and storage resources. The large-scale ML analysis model based on DA overcomes these problems by combining ML algorithm with distributed computing, which enables us to train complex models on huge data sets and make efficient predictions and decisions [5-6].

The purpose of this paper is to deeply discuss the principle, method and application of large-scale data processing and ML analysis model based on DA. A cluster topology driver module based on gradient switching and aggregate communication is designed, and its core goal is to adapt the distributed system to various underlying network topologies. By designing decentralized gradient exchange algorithm and aggregate communication framework, the parallel transmission ability of multi-interface network can be fully exerted, thus improving the model synchronization efficiency of ML task. Through the research and discussion in this paper, we hope that readers can better understand and apply the model of large-scale data processing and ML analysis based on DA, so as to mine meaningful information from the ever-emerging massive data and provide strong support for decision-making and innovation in various fields.

2. Research method

2.1 DA design

DA is a computing model, which is used to solve various problems in distributed systems. In a distributed system, multiple computers or nodes work together to complete tasks or solve problems. These nodes are usually distributed in different geographical locations and connected together through the network. The goal of DA is to coordinate the operations of these nodes to achieve a common goal or solve a problem [7]. DA is also used in distributed computing, allowing parallel execution of computing tasks on multiple nodes to improve performance and scalability. MapReduce and Apache Hadoop are some common distributed computing frameworks.

DA is widely used in cloud computing, big data processing, blockchain, distributed storage and distributed artificial intelligence. Designing and implementing efficient DA is one of the key challenges of distributed systems, because they need to consider many factors such as network delay, concurrency control, fault tolerance and performance [8-9]. The architecture of distributed job collaborative processing is shown in Figure 1:

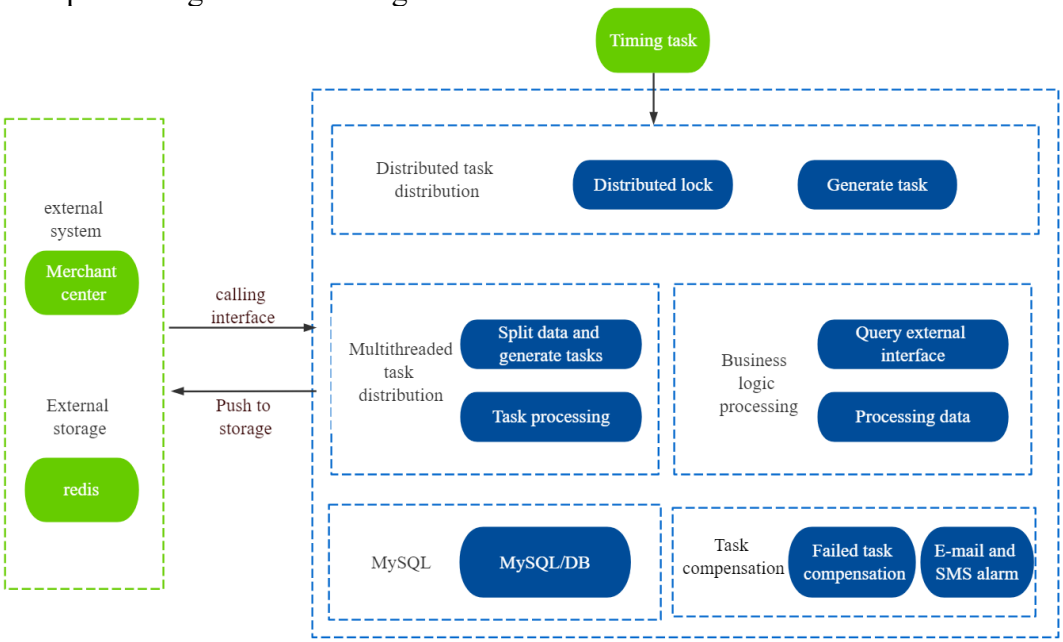

Figure 1 Distributed job cooperative processing architecture

Although multi-interface network can provide more available bandwidth for working nodes, the huge traffic generated during gradient switching is still the bottleneck restricting training efficiency. In the data center network, when the link and bandwidth resources are shared by multiple communication-intensive tasks, the data transmission and communication time will be significantly prolonged. The core of this algorithm is to use loose quantization and numerical discretization technology to map the original continuous floating-point gradient vector into discrete integer values, thus effectively reducing the number of bits needed for data expression and reducing the storage overhead of the system [10-11].

Gradient quantization method is a technology to reduce the calculation and transmission overhead of gradient in deep neural network. Its basic principle is to transform the gradient value from high-precision representation (usually floating point number) to low-precision representation, so as to reduce the calculation and communication costs and keep the model performance as much as possible. Gradient quantization methods usually involve determining the quantization parameters of gradients in advance, for example, using 8-bit integers to represent gradients. These parameters remain unchanged during model training.

Gradient quantization is a technique used to reduce the calculation and storage requirements of deep neural network models. The gradient quantization method usually includes the following steps:

Training deep neural network model: First, you need to train a deep neural network model, usually a deep learning model for specific tasks, such as CNN (Convolutional Neural Network) or RNN (Recurrent Neural Network). This model will be used to solve a certain task, such as image classification or natural language processing.

Gradient calculation: During the training period, the deep learning model calculates the gradient of each parameter through the back propagation algorithm, so as to update the weight of the model in the optimization process. These gradients are key information during training, but they are usually high-precision floating-point numbers.

Choose a gradient quantization method: Choose a gradient quantization method suitable for your task. Common gradient quantization methods include fixed-point representation, integer representation or other low-precision representations. These methods will map the original floating-point gradient value to a lower precision representation to reduce the calculation and storage overhead.

Gradient mapping: the calculated floating-point gradient is mapped and quantized into the representation of the selected precision. This usually involves mapping floating-point numbers to integers or other fixed-point representations.

Communication and calculation: on distributed deep learning or edge devices, gradient quantization is usually accompanied by gradient transmission and calculation operations. These operations need to transfer the quantized gradient to other devices and then calculate it.

Inverse quantization: At the receiving end, the quantized gradient is inversely mapped back to the original floating-point representation for gradient update. This step ensures that too much information will not be lost during model training.

Model update: use the gradient of inverse quantization to update the weight of the model. This is usually the same as the standard deep learning training process, but the quantization error of gradient needs to be handled carefully.

Cyclic iteration: Generally, gradient quantization is an iterative process. The above steps can be repeated many times to gradually reduce the quantization error to ensure that the performance of the model will not be greatly affected.

It should be noted that gradient quantization can significantly reduce the calculation and storage requirements of the model, but it will also introduce some information loss, so it is necessary to carefully weigh the trade-off between performance and resource consumption. Different tasks and hardware platforms may require different gradient quantization methods.

Because the gradient quantization algorithm needs different discretization and data mapping methods in the fully connected and convolution layers, the hierarchical structure of neural network will affect the effect of quantization in the model training process. In order to classify and control the hierarchical structure, this section puts forward the concept of "joint layer", which is used to reorganize the whole neural network structure to give full play to the advantages of gradient quantization [12]. The purpose of introducing joint layer is to dynamically adjust the granularity of gradient quantization and pipeline parallel operation, so as to further improve the calculation and communication efficiency in the back propagation stage during training. The definition of joint layer is a set of network hierarchical institutions with similar characteristics, which contains the commonness of computing load and communication characteristics, so the whole neural network model can be reorganized into a set of several joint layers.

2.2 Distributed ML analysis model

Cluster topology driver is a key component for managing and optimizing resource allocation and communication in computing clusters. Designing an effective cluster topology driver needs to consider several principles to ensure the performance, scalability and reliability of the cluster. Here are some design principles:

Performance optimization: The design of the drive should optimize performance to ensure efficient communication between nodes in the cluster. This includes reducing latency, maximizing bandwidth utilization and increasing data transmission speed.

Load balancing: The driver should be able to balance the load of each node in the cluster, so as to ensure reasonable resource allocation, and some nodes will not be overloaded while others are idle.

Fault tolerance: the driver should have fault tolerance mechanism, which can deal with problems such as node failure or communication interruption. This can be achieved by backing up nodes, automatic failover, or data replication.

Scalability: Drives should be able to easily expand to support more nodes or resources without introducing a lot of performance overhead or complexity.

Resource allocation strategy: The driver needs to define a clear resource allocation strategy to ensure that various workloads can be properly allocated. This may include CPU, memory, storage and other resources.

Communication optimization: The driver should optimize the communication mode to reduce the network overhead. This can be achieved by local data transmission, data compression, caching and other technologies.

Dynamic adaptability: Drives should be able to dynamically adapt to changes in the cluster, including the joining and leaving of nodes and changes in workload. This can be achieved through monitoring and automatic adjustment.

Security: The drive should have security measures to ensure that the communication and resource allocation in the cluster are not vulnerable to malicious attacks or unauthorized access.

Ease of use: User-friendliness should be considered in the design of the drive, so as to simplify the tasks of cluster management and configuration, and make it easy for administrators to perform necessary operations.

Monitoring and logging: Drives should provide detailed monitoring and logging functions to help administrators track and solve problems and conduct performance analysis.

These principles can help to design an efficient, reliable and scalable cluster topology driver to meet the needs of different cluster applications. When designing and implementing the driver, it is necessary to carefully consider the specific purpose and scale of the cluster, so as to make appropriate adjustments and customization according to the actual situation.

Cluster topology driver module is an important component for managing and controlling the connection and communication between nodes in the cluster. Its architecture design needs to consider many factors such as cluster size, performance requirements, scalability, fault tolerance and so on. The gradient switching algorithm and aggregate communication framework proposed in this paper are abstracted into a special cluster topology driver module. The core function of the module in the system is to adapt to the topology of the underlying network, and gradient slicing and numerical quantization techniques are used to accelerate the iterative process of model training.

Cluster topology driver modules are usually distributed to support large-scale clustering. Different nodes can run different sub-modules and work together to maintain the whole topology information. Topology information contains sensitive data, so it is necessary to ensure the confidentiality and integrity of the data. Use appropriate encryption and authentication mechanisms to protect data. Topology management interface allows administrators to query and modify topology information, such as manually adding or deleting nodes and changing node properties. The cluster topology driver module needs to support a communication protocol for information exchange between nodes. Common protocols include HTTP, RPC, WebSocket, etc. The specific choice depends on the requirements and performance requirements. The cluster topology driver module should generate detailed logs for troubleshooting and performance optimization. At the same time, it is necessary to implement monitoring mechanism and alarm system in order to find and deal with problems in time.

In addition, the cluster topology driver module realizes the acceleration of matrix operation and the collaborative control of distributed processes through the mixed coding of Python and C++ in Open MPI architecture and PyTorch distributed communication library. In order to be compatible with heterogeneous clusters composed of CPU and GPU nodes, the system will use CUDA and NCCL toolkits to accelerate the computing and communication processes respectively only when the nodes are equipped with GPU.

Distributed ML is a ML method, which uses multiple computing resources (usually multiple computers or servers) to process large-scale data sets and complex models. Deploy the trained model to the production environment for real-time or batch inference. This can involve servicing the model so that external applications can query the model to make predictions. As shown in Figure 2, it is a distributed ML analysis model.

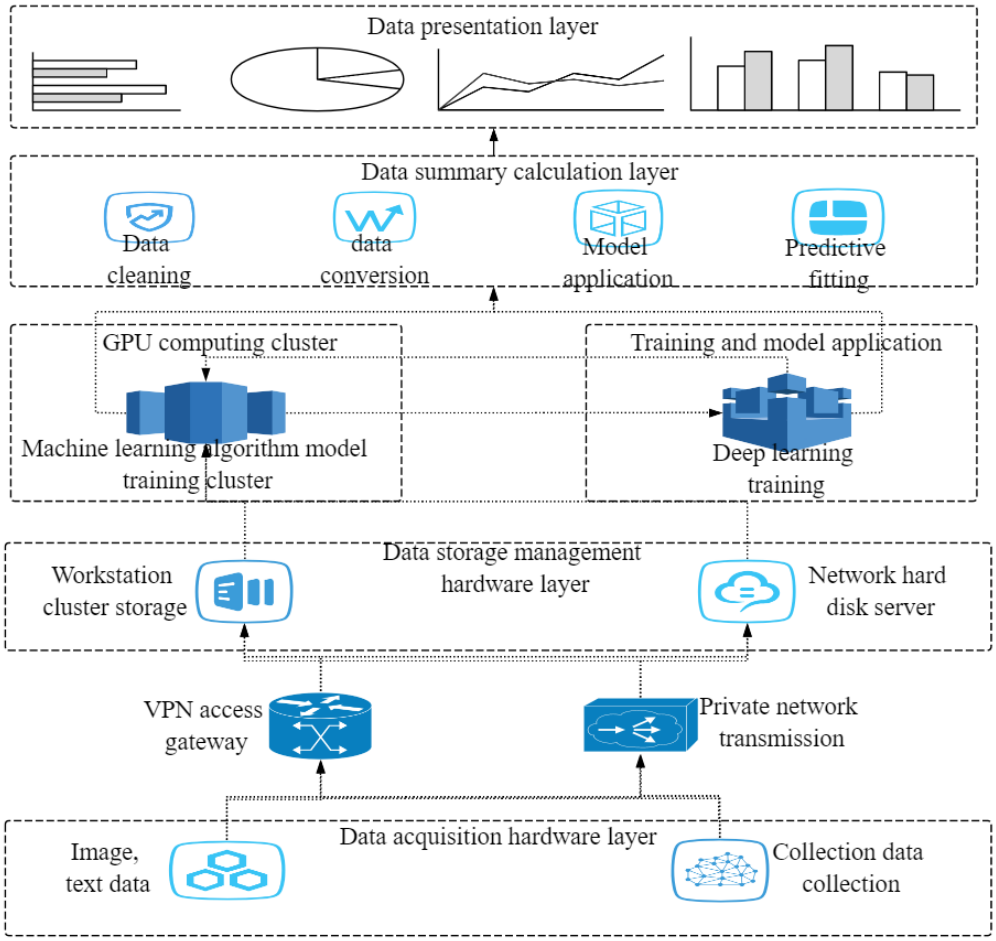

Figure 2 Distributed ML analysis model

First, you need to prepare data sets for training and testing. Data usually needs to be divided into small pieces for processing on multiple computing nodes. Data may also need preprocessing and feature engineering. Select the ML model suitable for the problem. In distributed environment, parallelizable models, such as random forest, gradient lifting tree and neural network, are usually used. The selected model is decomposed into several sub-models, which can be trained independently on different computing nodes. Each computing node is responsible for processing a part of data.

In a distributed environment, data is usually divided into multiple partitions and distributed to different computing nodes. This helps to process data in parallel. Sub-models are trained in parallel on each computing node, and their respective data partitions are used. This can significantly speed up the training process. The results of the sub-models trained on each computing node are integrated into the final model. This can be done by various techniques, such as voting, averaging, stacking, etc. The final model is evaluated with test data set to ensure its performance and generalization ability. Deploy the trained model to the production environment for real-time or batch inference. This can involve servicing the model so that external applications can query the model to make predictions.

3. Result analysis

The experiment relies on Alibaba Cloud's elastic computing service to build a cluster environment. All GPU nodes adopt Alibaba Cloud elastic GPU services and are virtualized based on GN5 and GN6 instances. The experiment uses two-dimensional image classification as the benchmark test. The

training task is based on six kinds of neural network models (AlexNet, VGG19, Inception-V3, ResNet18, ResNeXt101, ResNeXt152) and one kind of image data set ImageNet. In this experiment, the global batch size is set to 256, and the local batch size is adjusted according to the number of working nodes.

In the training process, the proportion of gradient quantization in the whole training process can be controlled, so as to compare the convergence performance of the model under different settings. Figure 3 shows the training quality of the gradient quantization algorithm.

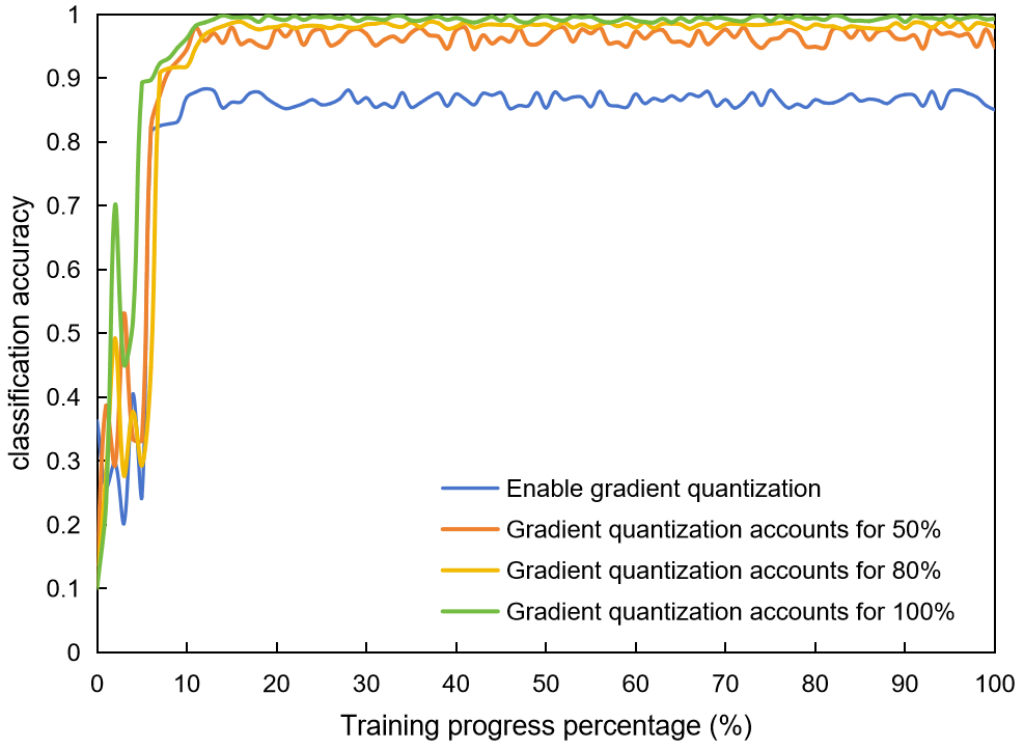

Figure 3 Training quality of gradient quantization algorithm

The proportion of gradient quantization will change from 0 (gradient quantization is not enabled at all) to 100% (gradient quantization is always enabled). It can be observed that compared with the original training configuration, the convergence speed of the model in the early training stage is obviously improved by enabling gradient quantization. The results show that the gradient quantization algorithm can effectively accelerate the convergence speed of training without reducing the accuracy and quality of the model.

According to different training tasks, the experiment also analyzes the calculation cost and proportion of gradient quantization algorithm in a single iteration. As shown in figure 4.

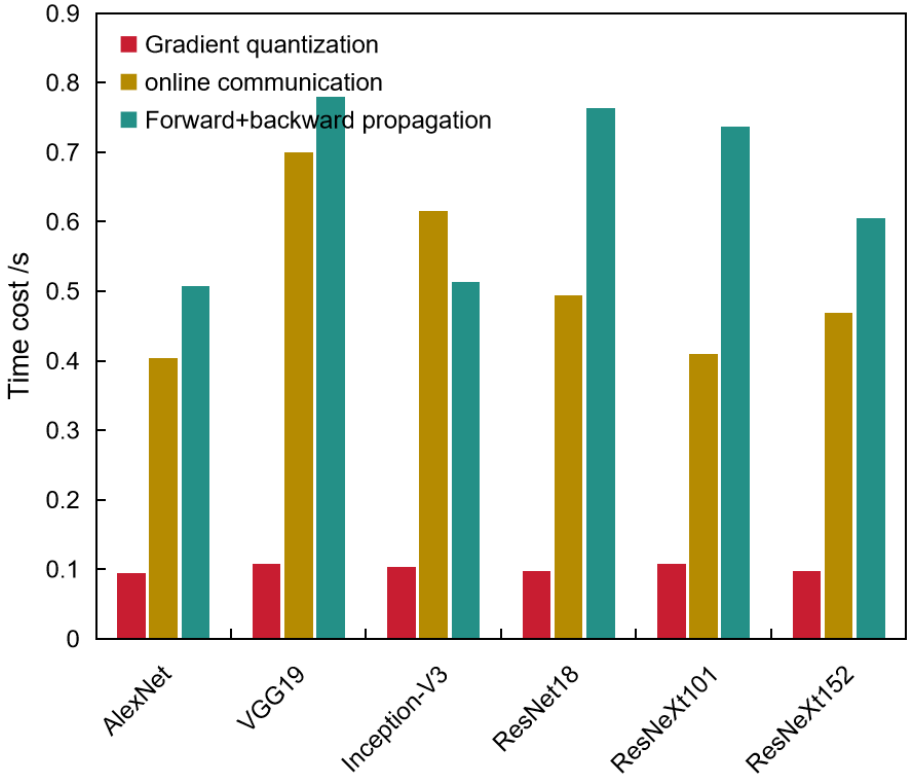

Figure 4 Comparison of time expenditure

It can be seen that the main calculation cost in a single iteration comes from the tensor calculation of forward and backward propagation, while the communication cost comes from the network transmission of gradient tensor. This phenomenon shows that the introduction of gradient quantization will not bring obvious computational overhead to distributed training system. In addition, the gradient quantization algorithm can also be realized by special hardware such as FPGA, so as to give full play to its advantages of data compression and calculation acceleration, which are worthy of further study in the future.

4. Conclusion

In this paper, a cluster topology driver module based on gradient exchange and aggregate communication is designed, which can be deployed in various distributed ML scenarios and can effectively improve the efficiency of the system in performing model training tasks. The research shows that DA has excellent performance and scalability in large-scale data processing and ML. By adding computing nodes, more data can be processed effectively and the training speed and accuracy of the model can be improved. Although DA has made remarkable progress in large-scale data processing and ML, there are still challenges. This includes communication overhead, data consistency, load balancing and other issues. Future research directions include improving the efficiency of DA, improving fault tolerance, and applying distributed ML in a wider range of applications.

References

[1] Lu, Y. , Gu, H. , Yu, X. , & Chakrabarty, K. (2021). Lotus: a new topology for large-scale distributed machine learning. ACM Journal on Emerging Technologies in Computing Systems(1), 17.

[2] Chang, K. , Jiang, W. , Zhan, J. , Gong, Z. , & Pan, W. (2021). Archnet: a data hiding design for distributed machine learning systems. Journal of systems architecture(114), 114.

[3] Zhang, Y. , Duchi, J. C. , & Wainwright, M. J. (2013). Divide and conquer kernel ridge regression: a distributed algorithm with minimax optimal rates. Journal of Machine Learning Research, 30(1), 592-617.

[4] Olfa, H. L. , Ichrak, M. , & Thomas, D. (2021). Machine learning to design an auto-tuning system for the best compressed format detection for parallel sparse computations. Parallel Processing Letters(4), 31.

[5] Forsati, R. , & Meybodi, M. R. (2010). Effective page recommendation algorithms based on distributed learning automata and weighted association rules. Expert Systems with Applications, 37(2), 1316-1330.

[6] Li, X. , Yan, Z. , & Liu, Z. (2019). Combination and application of machine learning and computational mechanics. Chinese Science Bulletin, 64(7), 635-648.

[7] Huang, H. , Xu, H. , Cai, Y. , Khalid, R. S. , & Yu, H. (2018). Distributed machine learning on smart-gateway network toward real-time smart-grid energy management with behavior cognition. ACM Transactions on Design Automation of Electronic Systems(5), 23.

[8] Yilmaz, A. , Kucuker, A. , & Bayrak, G. (2022). Automated classification of power quality disturbances in a sofc&pv-based distributed generator using a hybrid machine learning method with high noise immunity. International journal of hydrogen energy(45), 47.

[9] Al Karim, M. , Currie, J. , & Lie, T. T. (2018). A machine learning based optimized energy dispatching scheme for restoring a hybrid microgrid. Electric Power Systems Research, 155(78), 206-215.

[10] Ezzat, A. , Elnaghi, B. E. , & Abdelsalam, A. A. (2021). Microgrids islanding detection using fourier transform and machine learning algorithm. Electric Power Systems Research, 196(4), 107224.

[11] Benslimane, A. , Hussain, F. , Hussain, R. , & Anpalagan, A. (2020). A new block-based reinforcement learning approach for distributed resource allocation in clustered iot networks. IEEE Transactions on Vehicular Technology, (99), 1-1.

[12] Raza, M. , Hussain, F. K. , Hussain, O. K. , Zhao, M. , & Rehman, Z. U. (2019). A comparative analysis of machine learning models for quality pillar assessment of saas services by multi-class text classification of users' reviews. Future generation computer systems, 101(10), 341-371.