ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **27**, Number 1, 2024, 106–119

Neural Network-based Pattern Recognition in the Framework of Edge Computing

Jing NING

School of Information Engineering, Eastern Liaoning University, Dandong 118000, Liaoning, China
E-mails: ningjing@elnu.edu.cn
Corresponding author

Abstract. Neural network (NN) model has been widely used in *pattern recognition* (PR), speech recognition, image processing and other fields, but its application in *edge computing* (EC) environment faces performance and energy consumption problems. This article first introduced the basic structure and training process of NN, including backpropagation algorithms. Then, this article presented a NN modeling approach based on EC, including NN model compression, distributed NN model and knowledge distillation approach. Finally, this article implemented a PR model for the MNIST (Mixed National Institute of Standards and Technology database) dataset and analyzed the experimental results. The experimental outcomes indicated that the presented approach can significantly enhance the performance of the NN model in the EC environment, while ensuring a high recognition accuracy. The NN modeling approach based on EC can reduce the amount of computation and storage of the NN, thus improving the operating efficiency of the NN in the EC environment by 6%-12%. The NN modeling approach based on EC can optimize the performance and efficiency of the NN model in the EC environment, and provide new ideas and approaches for the application of NN in the EC environment.

Key-words: Edge computing; Mixed National Institute of Standards and Technology Database; neural networks; pattern recognition.

1. Introduction

Edge computing is to provide cloud services and IT environment services for application developers and service providers at the edge of the network; the goal is to provide computing, storage, and network bandwidth close to data input or users. EC is a computing mode that brings computing resources close to data sources, aiming to address the issues of massive data transmission, processing delay, etc. in traditional cloud computing architecture. It is gradually becoming a hot area of concern. NN, as an important machine learning technology, are widely

used in fields such as image recognition and speech recognition. However, the application of NN in the field of EC is greatly limited due to its large amount of computation and high computational complexity. Therefore, this article mainly studies the optimization of NN modeling approach based on EC and the realization of PR model, and discusses the main research content of this article on this basis: the optimization of NN modeling approach based on EC is mainly to solve the problem of low efficiency of traditional NN models in EC environment, so as to improve the performance of NN. Firstly, the existing NN modeling approaches are analyzed and summarized. Through experimental verification, the optimization approach for NN modeling is discussed. This approach can not only quickly construct efficient network models, but also effectively reduce network training time. Secondly, on the basis of the above approaches, a NN PR system based on EC is constructed.

NN, as the fundamental model of AI, are currently a hot research topic, and many scholars have conducted research on NN. Iris Cong proposed a quantum circuit-based algorithm inspired by convolutional NN [1]. David Bau believed that deep NN are adept at finding hierarchical representations to solve complex tasks on large datasets, and proposed an analytical framework for network anatomy [2]. Kun Yao believed that NN have the potential to solve the problem of traditional force fields being unable to simulate chemical reactivity, and have low universality without re-fitting. He also proposed that NN molecular dynamics has the potential to reduce resource barriers in simulation chemistry [3]. Nikolaus Kriegeskorte believed that deep NN models can help people understand brain computing [4]. Changliu Liu proposed that deep NN is widely used for nonlinear function approximation, and its application range is from computer vision to control [5]. John T. Hancock proposed dividing the techniques for using classified data in NN into three categories. Three different patterns have been discovered in technology, which identify technology as deterministic, algorithmic, or automated patterns [6]. Abul Bashar believed that *deep learning* (DL) NN have the most advanced accuracy and introduced a survey of DL NN architectures used in various applications, which accurately classify through automatic feature extraction [7]. These scholars have explored the applications of NN from multiple perspectives, their role in DL, and their potential for improving human life.

At present, EC has been widely used in various fields, and many scholars have studied it. Jiasi Chen believed that EC is a feasible approach to meet the high computing and low latency requirements of DL of edge device by placing the fine grid of computing nodes near the terminal devices [8]. Fang Liu proposed the energy efficiency and DL optimization of EC system, and also studied the open problem of analyzing and designing EC system [9]. Gopika Premsankar believed that EC is necessary to satisfy the delay needs of applications relating virtual reality and augmented reality [10]. Olga Krestinskaya believed that the data processing capability of EC devices can reduce the multiple costs of cloud computing solutions [11]. Yuan Ai proposed to transfer the function of centralized cloud computing to the edge device of the network [12]. Preitl, Zsuzsa presented some main aspects regarding multi-parametric quadratic programming (mp-QP) problems. *Model Predictive Control* (MPC) is considered as a particular mp-QP problem, and this powerful tool is applied for control and simulation through a case study [13]. Precup also discussed the situation prior to using the proposed approach. The results of this analysis demonstrate the efficiency of his approach based on complex systems optimization, modeling, and control targeting real-world practical applications, and a numerical outcome of the approach is given. This allows students to gain a better understanding of the theoretical aspects acquired during the lectures in comparison with the situation prior to using the proposed approach [14]. Zamfirache introduced a novel reference tracking control approach implemented using a combi-

nation of the Actor-Critic *Reinforcement Learning* (RL) framework and the *Grey Wolf Optimizer* (GWO) algorithm. The classical *neural network* (NN)-based implementation of the Critic, optimized with the *Gradient Descent* (GD) algorithm, is replaced with the GWO algorithm, aiming to eliminate the main drawbacks of the GD algorithm, ie, slow convergence and the tendency to get stuck in local optimal values [15]. The research on EC has been very mature, but the research involving NN is less.

Since the development of pattern recognition in the 1920s, it is a common view that there is no single model and single technology to solve all pattern recognition problems. What we have is just a tool bag, what we need to do is combine statistical and syntactic recognition with specific problems, combine statistical or syntactic pattern recognition with heuristic search in artificial intelligence, and combine statistical or syntactic pattern recognition with machine learning of support vector machines. Artificial neural networks can be combined with various existing technologies, as well as expert systems and uncertain reasoning approaches in artificial intelligence, to gain a deep understanding of the effectiveness and potential of various tools, learn from each other's strengths and weaknesses, and create new prospects for pattern recognition applications. This article discussed the NN modeling approach based on EC to optimize the performance of PR model, and tested it on MNIST dataset. Firstly, the basic structure and training process of NN were introduced, and then the improvement ideas and implementation process of this approach were elaborated in detail. Finally, this article provided a specific implementation of a PR model based on this approach and conducted experimental verification. The research results of this article showed that the NN modeling approach based on EC is an effective approach for optimizing NN models. The compression and knowledge distillation of NN models can not only reduce the storage and computational resource consumption of NN models, but also enhance the accuracy and generalization ability of the models.

In the research of pattern recognition based on neural network under the framework of edge computing, Section 2 mainly introduces the basic structure and training process of neural network. In Section 3, this paper mainly analyzes the optimization of neural network modeling approaches based on edge computing, and deeply studies neural network model compression, distributed neural network model and knowledge distillation approaches. Section 4 discusses the construction of neural network pattern recognition models. Section 5 conducts experimental analysis and characterization and discussion of experimental results from several aspects, including accuracy, performance level, model parameter improvement effect, and application testing. Finally, a summary of the research findings, contributions, and impacts of this article is presented in Section 6.

2. Basic Structure and Training Process of NN

2.1. Basic structure of NN

NN are AI algorithms that mimic the working principles of biological neural systems. Its basic structure includes input layer, hidden layer, and output layer. The specific functions are shown in Table 1.

In a NN, each neuron is connected to other neurons and has weights. These weights are adjusted during the training process to minimize errors and maximize accuracy.

Table 1. Basic structure of NNs

Structure	Function
Input layer	The input layer of a NN accepts external input data and converts it into a form that the network can process, usually a set of numbers or vectors.
Hidden layer	The hidden layer is the core part of a NN, composed of multiple neurons. The neuron receives the input signal from the upper layer, performs weighted
Output layer	The output layer receives signals from the hidden layer and converts them into the final output result. The output layer is usually task related, for example, in image classification tasks, the output layer typically represents different categories.

2.2. Backpropagation algorithm

The backpropagation algorithm, as a commonly used optimization approach for training models in NN, updates the weights and biases of NN by calculating error gradients. The basic thought of the back propagation algorithm is to perform gradient descent optimization by calculating the partial derivative of the loss function for each weight [16–17]. The steps of the back-propagation algorithm are as follows: 1) Forward propagation: The input data is passed through the input layer of the NN, and the results of the output layer are finally obtained through the operation of the hidden layer and the activation function. 2) Calculation error: The output results of the NN are compared with the actual results, and the value of the loss function is calculated. 3) Back-propagation error: Starting from the output layer, the error gradient of each neuron is calculated and the error is propagated forward to the hidden layer and input layer. 4) Updating weight and offset: According to the error gradient, the weights and offsets of the NN are updated to minimize the loss function.

In practical applications, approaches such as Batch Gradient Descent or Stochastic Gradient Descent are usually used to update the weights for a batch of samples or a single sample. In addition, techniques such as Momentum optimization and Adaptive Learning Rate can also be used to improve training effectiveness.

2.3. NN training process

The training process of the NN is to update the weight and deviation of the NN by optimizing the back-propagation algorithm of the input data to minimize the loss function and maximize the prediction accuracy. The training process of the NN is as follows:

The first step is to initialize the weights and deviations, which need to be randomly initialized to a small value for each neuron's weight and deviation, usually using a normal or uniform distribution to generate random numbers.

Forward propagation: The input data is passed through the input layer of the NN, and through the operation of the hidden layer and the activation function, and finally the outcome of the output layer is obtained. The forward calculations are done in terms of

$$\hat{y} = f^L(\dots f^2(f^1(x))) \tag{1}$$

$$p_i = soft\max(\hat{y}_i) \tag{2}$$

where x, y and p are input values, output values, and probabilities, respectively.

The loss function is calculated. The value of the loss function is calculated by comparing the difference between the output result of the NN and the actual result. The expression of the loss function is

$$Loss = -\frac{1}{N}\sum_{i}^{N}\sum_{c}^{C} y_{ic}\log(p_{ic}) \tag{3}$$

where N is the number of samples, and C is the number of categories.

Back propagation is used to update the weight and deviation of the NN by calculating the gradient of the loss function to the weight and deviation. The formulae specific to back propagation are

$$\delta_1 = \frac{\partial L}{\partial \alpha_1} = \frac{\partial L}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial \alpha_1} = \delta_2 W_1^T \tag{4}$$

$$W_{g1} = \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \alpha_2}\frac{\partial \alpha_1}{W_1} = \delta_2 \alpha_1 \tag{5}$$

wgere δ is the deviation, and W is the gradient of the weight.

It is necessary to repeat the steps of forward propagation, calculation of loss function and back propagation until the value of loss function is small enough or the training times reach the set value.

In the training process, attention should be paid to avoid overfitting, that is, the NN performs well on the training set, but poorly on the test set. To avoid overfitting, regularization approaches, such as l_1 and l_2 regularization, can be used to limit the weight. In addition, techniques such as data augmentation and dropout can also be used to improve the generalization ability of NN.

3. Optimization of NN Modeling Approaches Based on EC

In the EC environment, the NN model needs to satisfy the needs of real-time response and small computing resource consumption. For this reason, researchers have proposed a NN model modeling approach based on EC, and optimized the model structure and parameters to achieve efficient PR tasks on edge device. Common optimization approaches include NN model compression, distributed NN models, and knowledge distillation approaches. These approaches can design and optimize appropriate NN models according to different application scenarios and the characteristics of edge device to achieve efficient PR tasks on edge device.

The first network design model principle (past algorithms) used by the author for this task is to significantly increase the depth of the network, which has been elaborated in many existing studies. This principle is that the model can learn from deeper feature embeddings.

A major problem with this principle is that when the model is too deep, it is very difficult for the model to train on the riverside. In order to solve this problem and improve the accuracy of the model, the author proposed a second design principle (optimization algorithm), which is to add shortcut connections in the network. By adding shortcuts to the residual architecture, iterative optimization approaches, including random gradient descent, will be more effective even in deep networks.

3.1. NN model compression

NN model compression refers to the process of pruning, quantifying, and decomposing NN models to reduce model size, reduce computational complexity, and improve model efficiency. NN model compression technology is currently a hot research direction in the field of DL, with a wide range of applications, including DL applications on mobile devices and DL training on large-scale cloud computing platforms. Common NN model compression approaches include:

Weight pruning: By removing some weights from the NN, the goal is to reduce the size of the model. Pruning can be done based on weight size or through approaches such as clustering and cluster analysis. Weight pruning can reduce the model size to 10%–20% without significantly affecting the accuracy of the model. Channel pruning: For convolutional NN, channel pruning can delete some channels in the convolutional layer, reducing the number of parameters in the convolutional layer. Channel pruning can reduce the parameters in the convolutional layer to the original 20-30% without reducing the accuracy of the model. Quantization: By quantifying the parameters in the NN from high precision (such as 32-bit floating point) to low precision (such as 8-bit integer), the storage and calculation overhead of the NN model can be reduced. Without reducing the accuracy of the model, quantization can reduce the size of the model to more than four times its original size. Decomposition: By decomposing the convolutional or fully connected layers in a NN into multiple sub layers, the number of model parameters can be reduced, thereby reducing model size and computational complexity. Decomposition can be carried out through approaches such as low-rank matrix factorization.

In summary, NN model compression is a hot research direction in the field of DL, which has broad application prospects in reducing model size, reducing computational complexity, and improving model operation efficiency.

3.2. Distributed NN model

Distributed NN model refers to distributing the calculation and storage of NN models across multiple computing nodes, and improving the efficiency and speed of model training through parallel computing [18]. It typically includes two key technologies: data parallelism and model parallelism.

Data parallelism refers to spreading training data across multiple computing nodes. Each node uses the same NN model for training, but uses different data. The backpropagation of each node calculates the gradient and aggregates it to the central node. The central node uses the summarized gradient to update the parameters of the NN model. Model parallelism refers to the allocation of different layers of a NN model to different computing nodes, with each node responsible for calculating the forward and backward propagation of the NN layers assigned to it. Data exchange and synchronization between different layers are achieved through message passing between each node. Model parallelism can solve the problem of large NN models being unable to store and compute on a single computing node.

Distributed NN model is widely applied in the field of DL, especially in large-scale DL training. At the same time, distributed NN models also bring new challenges and problems, such as communication and synchronization issues between computing nodes, and handling node failures. Therefore, in the design and implementation of distributed NN models, it is necessary to consider these issues and select appropriate algorithms and frameworks for implementation.

3.3. Knowledge distillation approaches

Knowledge distillation is an approach used for NN model compression, which introduces knowledge from larger models into smaller models to achieve similar performance on smaller models. The basic idea of knowledge distillation approach is to use the output of larger models (that is, probability distribution) as the soft targets of smaller models, and use these soft targets to guide the training of smaller models to accomplish better performance. Specifically, the knowledge distillation approach includes the following steps: The first step is to train a larger model, such as a deep NN with many layers and parameters, to provide "soft targets". The next step is to use this larger model to generate soft targets for the training set, and use these soft targets as objective functions for the smaller model. Smaller models are trained based on these soft targets to achieve performance similar to larger models. Finally, when generating soft targets, a "distillation temperature" parameter can also be used to control the "softness" of the soft target. Higher temperatures can lead to softer targets, thereby improving the training speed and stability of smaller models, but may reduce performance. The lower temperature would lead to a harder target, thus improving performance, but may lead to overfitting of smaller models.

The knowledge distillation approach can enable smaller models to have similar performance to larger models, and can reduce the computational and storage costs of the model.

4. Implementation of PR Model

4.1. Test dataset

The MNIST dataset takes a significant part in the implementation of PR models. Firstly, the MNIST dataset has become one of the standard benchmark test sets in the field of machine learning, widely used for algorithm performance evaluation and comparison in fields such as image classification and PR [19]. Therefore, researchers can use the MNIST dataset to evaluate the performance of their models in handwritten digit recognition tasks. Secondly, the MNIST dataset can serve as an entry-level dataset to help beginners understand the basic structure and training process of NN. Due to the clear labeling of the MNIST dataset, learners can quickly build and train a simple NN model, thereby mastering the basic principles and training process of NN. In addition, MNIST data sets can also be used to train and debug various DL models, such as convolutional NN, recurrent NN and multilayer perceptron. These models can be used for recognizing handwritten numbers, other image classification and PR tasks. Therefore, the MNIST dataset plays an important role in the implementation of PR models.

4.2. Model implementation steps and iterative optimization

The PR model of NN is the process of automatically determining the category of the target or sample to be recognized according to its characteristics. In PR, there are two main types of NN: feedforward NN and feedback NN. The feedforward NN processes the input signal through the weight and offset between the input layer and the output layer. It is composed of a hidden layer and an output layer. Feedback NN is a special type of NN that introduces feedback information on the basis of feedforward NN. The implementation process of NN PR models usually includes the following steps:

In the data preprocessing stage, it is necessary to load and process the raw data to make it suitable for the training of NN. Data preprocessing mainly includes data cleaning, data conver-

sion, and data normalization. Data cleaning refers to removing unnecessary data, processing missing data, removing outlier, etc. Data conversion is the process of converting raw data into a format that can be processed by NN, such as converting text data into numerical data. Data normalization is the process of normalizing data to ensure the same range of values for different features and to make the training process more stable.

The design of NN structure is an important step in model implementation. When designing a network structure, it is necessary to consider factors such as the characteristics of the data, the interpretability of the model, and the performance of the model. Common network structures include feedforward NN, convolutional NN, cyclic NN, etc. The network structure design also needs to determine the activation function, loss function, optimization algorithm, etc.

Model training is a process of updating network parameters through training data. The commonly used optimization algorithms include random gradient descent approach, momentum approach, Adam (adaptive moment estimation), etc. During the model training process, it is necessary to divide the training data into training sets, validation sets, and testing sets.

Model evaluation evaluates the performance of a model through a test set, typically using metrics such as accuracy, recall, and accuracy. When evaluating the performance of the model, it is also necessary to consider the generalization ability and overfitting of the model.

Model tuning is the optimization of a model to improve its performance. The model tuning approaches include adjusting the network structure, changing the learning rate, and adding regularization items. During the model tuning process, it is necessary to continuously adjust the model parameters based on the model evaluation results.

Model application is the application of a trained model to practical scenarios. The application approaches of the model include batch processing and online processing. In the process of model application, it is necessary to consider factors such as the running speed, reliability, and security of the model. Overall, the implementation process of NN PR models is an iterative process that requires continuous data preprocessing, network structure design, model training, model evaluation, and tuning to achieve optimal recognition results.

5. Experimental Results and Discussion

At present, deep neural networks mainly optimize their models through gradient descent, which can be divided into three forms: batch gradient descent, random gradient descent, and small batch gradient descent. The optimization problem includes the following directions: in addition to adjusting the learning rate, the direction of parameter updates can also be replaced by using the average gradient of the recent period of time as the current gradient. The Adaptive Moment Estimation (Adam) algorithm can be seen as a combination of momentum approach and RMSprop. It not only uses momentum as the parameter update direction, but also can adaptively adjust the learning rate.

The training process of the model is very important. The training set is used to fit the model and train the classification model by setting the parameters of the classifier. When combined with the validation set, different values of the same parameter will be selected to fit multiple classifiers. The purpose of a Cross Validation set is to use each model to predict the validation set data and record the accuracy of the model after training multiple models to find the best performing model. This article selects the parameters corresponding to the best performing model, which are used to adjust model parameters, such as parameter c and kernel function in SVM. Test set refers to the use of the training set and validation set to obtain the optimal model, and then use the test

set for model prediction. It can be used to measure the performance and classification ability of the optimal model, that is, the test set can be treated as a non-existent dataset. After the model parameters have been determined, the test set can be used for model performance evaluation.

When deploying branch EC based neural network modeling in the edge computing scenario, it first needs to complete the model training process on the ECS, which is delay tolerant. This paper focuses on the process of model task inference. The structure of EC based neural network modeling is an ordered sequence between layers, where each layer receives the output data from the previous layer and processes it before transmitting it to the next layer. By utilizing this feature, EC based neural network modeling can be constructed as an EC based neural network modeling with multiple branches, each consisting of a specific neural network layer, which together form a complete EC based neural network modeling. In response to the problem of excessive memory space and resources required for convolutional neural network computation in resource limited embedded devices, the optimization algorithm proposed in this paper combines network weight pruning and dynamic fixed-point quantization approaches for embedded platforms. This can effectively compress the neural network model and reduce computational consumption.

In neural networks, parameters are usually weights and biases, which control the model's transformation of input data. The goal of the training process is to adjust these parameters so that the model can minimize the error between prediction and actual labels on the training data, thereby achieving better generalization ability. The learning rate, weight and biases, batch size, epochs are selected by the user, and the activation function is randomly selected. Through appropriate optimization algorithms and techniques, the accuracy and performance of the model can be significantly improved. Firstly, the model was subjected to training and testing errors, as shown in Table 2. Ten tests were conducted on the accuracy of the model on the training set, validation set, and testing set. The training error reflects the fitting degree of the model on the training set, and the testing error reflects the generalization ability of the model on unprecedented data.

Table 2. Model test results

	Training set	Adjusted training set	Validation set	Adjusted validation set	Test set	Adjusted test set
Test1	82	82	85	84	70	80
Test2	83	83	79	80	74	78
Test3	79	83	78	84	69	81
Test4	81	79	83	82	70	79
Test5	80	79	79	81	72	82
Test6	83	84	83	81	73	80
Test7	78	85	85	83	71	80
Test8	82	81	84	80	72	84
Test9	84	83	83	82	70	80
Test10	84	82	84	79	68	84

Usually, the training error and testing error of the model were to some extent related, as they were all evaluated on the same dataset. However, it can be found that the model performed very well on the training set in this test, but performed poorly on the test set, which meant that the model may overfitting the training set data. If the training error of the model is low, but the testing error is high, it is necessary to check whether the data preprocessing is correct or whether there are issues such as data leakage.

Therefore, by using the regularization approach, such as l_1 and l_2 regularization, the second test was conducted after adjusting the model. The test results are shown in Fig. 1.

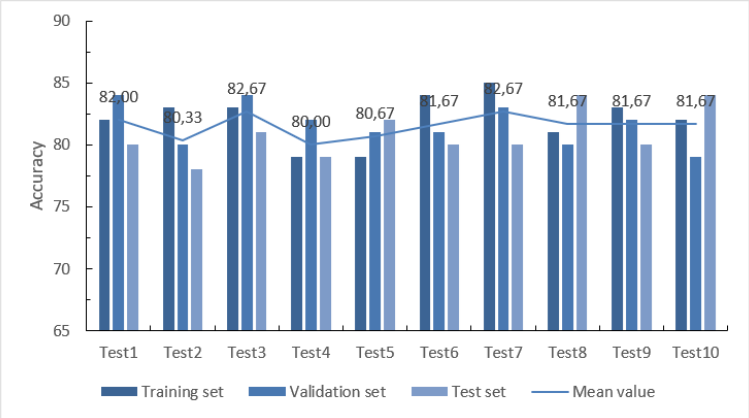

Fig. 1. The number of participants to online conferences organized by the Romanian Academy in the period April 2020- August 2023. Source: Authors' graph.

The training error and test error of the model basically maintain a positive correlation. However, at this point, the overall accuracy of the model is only about 80%, and a lot of training is needed to improve the accuracy of the model.

A large amount of training was conducted on the NN model to test its performance level in Fig. 2.

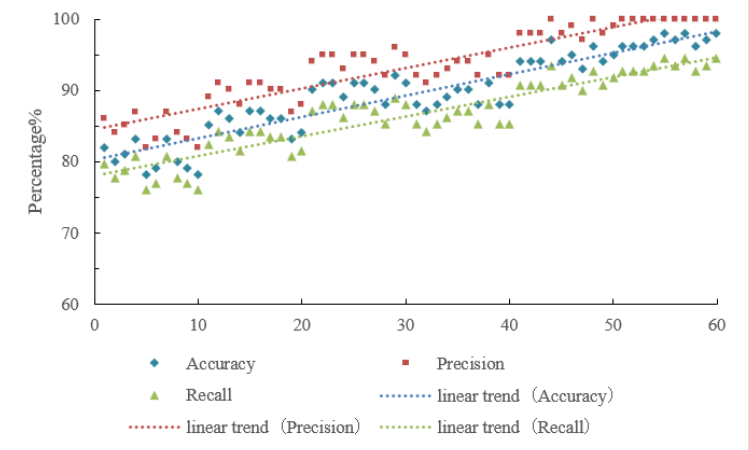

Fig. 2. The number of participants to online conferences organized by the Romanian Academy in the period April 2020- August 2023. Source: Authors' graph.

This test mainly tested the PR function of the model. 50 target images and 50 non target images were selected for each round of testing, and a total of 60 rounds of testing were conducted to test the accuracy, precision, and recall of the model. Through testing, it was found that

the overall accuracy, precision, and recall of the model showed an upward trend, and with the increase of training times, the accuracy, precision, and recall of the model gradually stabilized. Overall, the performance of the model fluctuates greatly in the early stages, gradually improves with the increase of training times, and tends to stabilize.

In terms of optimizing the model, NN model compression, distributed NN model, and knowledge distillation approaches were used to test the improvement effect of different approaches on model parameters, as shown in Fig. 3.

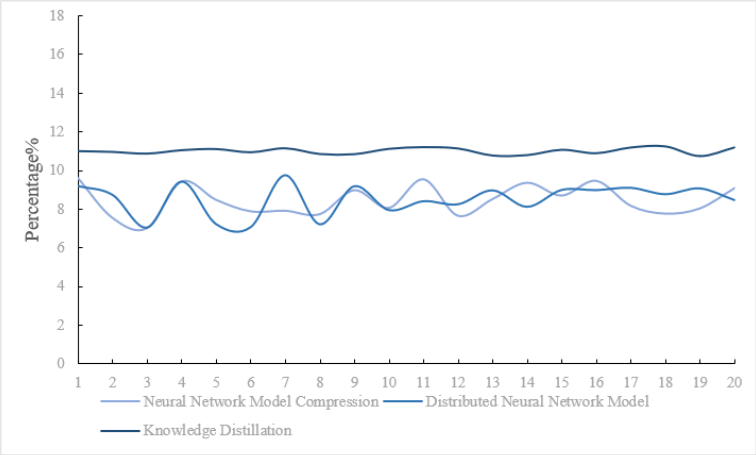

Fig. 3. The number of participants to online conferences organized by the Romanian Academy in the period April 2020- August 2023. Source: Authors' graph.

The experimental results showed that these approaches can effectively optimize the model structure and parameters, reduce the computational resource consumption of the model, and maintain high recognition accuracy. By comparing the experimental results of different optimization approaches, it can be concluded that the knowledge distillation approach was slightly better than the NN model compression and distributed NN model. The application of different optimization approaches can improve the overall efficiency of the model by 6%-12%. The optimization effect of the knowledge distillation approach reached over 10%. The compression of NN models, distributed NN models, and knowledge distillation approaches have improved the efficiency of NN models. In this experiment, this article used the classic MNIST dataset for PR tasks. Experimental results showed that the NN model based on EC has less computing resource consumption. The fundamental purpose of establishing a NN model is to apply the trained model to practical scenarios. Therefore, in the process of model application, it is necessary to consider factors such as the running speed, reliability, and safety of the model.

As shown in Fig. 4, the model was subjected to application testing, selecting indicators such as model recognition speed, information synthesis ability, and information integrity for multiple rounds of testing. Among them, the recognition speed of the model for each frame of image reached 20–30 milliseconds, and the information comprehensive ability index and information integrity index were basically between 8–9. Overall, in practical applications, the running speed of the model meets the standards, and its reliability and safety are guaranteed.

In the neural network modeling of ordinary edge computing, multiple branch network models are added and early exit points are deployed. When the system is confident about the inference

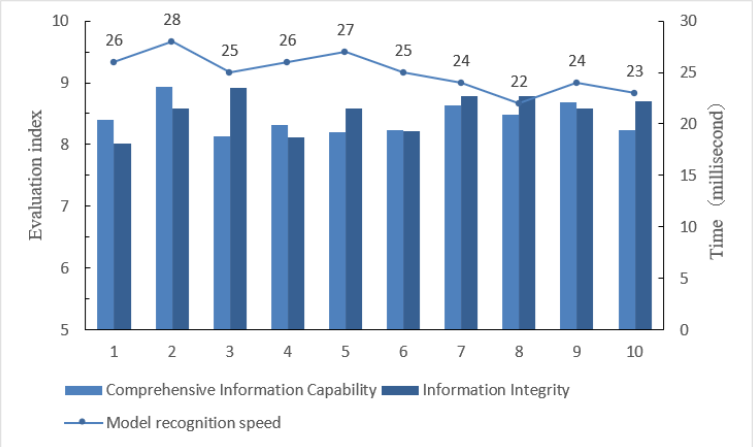

Fig. 4. The number of participants to online conferences organized by the Romanian Academy in the period April 2020- August 2023. Source: Authors' graph.

task, it can exit the task from the branch added in the middle of the model without making samples go through all levels of neural network modeling of edge computing, thus reducing the inference delay and improving the operation efficiency. To be effective, feature selection should improve classification performance while reducing the number of features. The existing algorithm can be adjusted and modified as a feature selector. The implementation of the anarchic social optimization algorithm is a human inspired algorithm that serves as a feature selector. Someone has proposed that the nearest meta heuristic optimization algorithm, the Grey Wolf optimizer, can be used to optimize and adjust the membership function parameters and rules of the proposed fuzzy controller [20–21]. Evolutionary algorithms are effective approaches for solving complex optimization problems with multiple constraints. In response to the local adjustment characteristics and cross recombination of cubic B-spline curves in differential evolution algorithms, someone has designed and implemented a *global optimal brainstorming optimization* (GBSO) algorithm based on cross recombination to solve the multi constraint 3D path planning problem considering path continuous curvature [22].

6. Conclusions

Based on the NN modeling approach of EC, this article discussed the optimization approach of NN in the field of PR. A detailed introduction was given to the basic structure, training process, backpropagation algorithm, model compression, distributed NN model, and knowledge distillation approach of NN. The implementation process of NN models was demonstrated using the MNIST dataset as an example. In PR tasks, NN models have become a very effective tool. The process of modeling approach preference and implementation of NN models in PR tasks can be seen through the presentation of this article. Among them, the training process of the model is very important, and through appropriate optimization algorithms and techniques, the accuracy and performance of the model can be significantly improved. In practical applications, as the amount of data continues to increase, the model size and computational complexity also

continue to increase. Therefore, model compression and distributed NN models have become research hotspots. The knowledge distillation approach can reduce the size and computational complexity of the model while ensuring its accuracy. In conclusion, the NN modeling approach based on EC can optimize the performance and efficiency of the NN model in the EC environment.

References

[1] I. CONG, S. CHOI and M. D. LUKIN, *Quantum convolutional neural networks*, Nature Physics **15**(12), 2019, pp. 1273–1278.

[2] D. BAU, J.-Y. ZHU, H. STROBELT, A. LAPEDRIZA, B.-L. ZHOU and A. TORRALBA, *Understanding the role of individual units in a deep neural network*, Proceedings of the National Academy of Sciences **117**(48), 2020, pp. 30071–30078.

[3] K. YAO, J. E. HERR, D. W. TOTH, R. MCKINTYRE and J. PARKHILL, *The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics*, Chemical Science **9**(8), 2018, pp. 2261–2269.

[4] N. KRIEGESKORTE and T. GOLAN, Neural network models and DL, Current Biology **29**(7), pp. 231–236, 2019.

[5] C.-L. Liu, T. ARNON, C. LAZARUS, C. STRONG, C. BARRETT and M. J. KOCHENDERFER, *Algorithms for verifying deep neural networks*, Foundations and Trends in Optimization **4**(3), 2021, pp. 244–404.

[6] J. T. HANCOCK and T. M. KHOSHGOFTTAR, *Survey on categorical data for neural networks*, Journal of Big Data **7**(1), 2020, pp. 1–41.

[7] A. BASHAR, *Survey on evolving DL neural network architectures*, Journal of Artificial Intelligence **1**(2), 2019, pp. 73–82.

[8] J.-S. CHEN and X.-K. RAN, *DL with edge computing: A review*, Proceedings of the IEEE **107**(8), 2019, pp. 1655–1674.

[9] F. LIU, G.-M. TANG, Y.-H.-Z. LI, Z.-P CAI, X. Z. ZHANG and T.-Q. ZHOU, *A survey on edge computing systems and tools*, Proceedings of the IEEE **107**(8), 2019, pp. 1537–1562.

[10] G. PREMSANKAR, M. D. FRANCESCO and T. TALEB, *Edge computing for the Internet of Things: A case study*, IEEE Internet of Things Journal **5**(2), 2018, pp. 1275–1284.

[11] O. KRESTINSKAYA, A. P. JAMES and L. O. CHUA, *Neuromemristive circuits for edge computing: A review*, IEEE Transactions on Neural Networks and Learning Systems **31**(1), 2019, pp. 4–23.

[12] Y. AI, M.-G. PENG and K.-C. ZHANG, *Edge computing technologies for Internet of Things: a primer*, Digital Communications and Networks **4**(2), 2018, pp. 77–86.

[13] Z. PREITL, R.-E. PRECUP, J. K. TAR and M. TAKACS, *Use of multi-parametric quadratic programming in fuzzy control systems*, Acta Polytechnica Hungarica **3**(3), 2006, pp. 29–43.

[14] R.-E. PRECUP, E.-L. HEDREA, R.-C. ROMAN, E. M. PETRIU, A.-I. SZEDLAK-STINEAN and C.-A. BOJAN-DRAGOS, *Experiment-based approach to teach optimization techniques*, IEEE Transactions on Education **64**(2), 2021, pp. 88–94.

[15] I. A. ZAMFIRACHE, R.-E. PRECUP, R.-C. ROMAN and E. M. PETRIU, *Neural network-based control using actor-critic reinforcement learning and grey wolf optimizer with experimental servo system validation*, Expert Systems with Applications **225**, 2023, paper 120112.

[16] L. G. WRIGHT, T. ONODERA, M. M. STEIN, T.-Y. WANG, D. T. SCHACHTER, Z. HU and P. L. McMAHON, *Deep physical neural networks trained with backpropagation*, Nature **601**(7894), 2022, pp. 549–555.

[17] C.-J. SUN, K.-X LI, J.-L ZHANG and C. HUANG, *Prediction of phthalates concentration in household dust based on back propagation neural network*, Indoor and Built Environment **31**(1), 2022, pp. 230–244.

[18] W. WANG, Y.-J. YANG, X. WANG, W.-Z. WANG and J. LI, *Development of convolutional neural network and its application in image classification: a survey*, Optical Engineering **58**(4), 2019, pp. 040901–040901.

[19] Z.H. PERIC, B. D. DENIC, M. S. SAVIC, N. J. VUCIC and N. B. SIMIC, *Binary quantization analysis of neural networks weights on MNIST dataset*, Elektronika ir Elektrotechnika **27**(4), 2021, pp. 55–61.

[20] U. KILIC, E. S. ESSIZ and M. K. KELES, *Binary anarchic society optimization for feature selection*, Romanian Journal of Information Science and Technology **26**(3), 2023, pp. 351–364.

[21] C.-A. BOJAN-DRAGOS, R.-E. PRECUP, S. PREITL, R.-C. ROMAN, E.-L. HEDREA and A.-I. SZEDLAK-STINEAN, *GWO-based optimal tuning of type-1 and type-2 fuzzy controllers for electromagnetic actuated clutch systems*, IFAC-PapersOnLine **54**(4), 2021, pp. 189–194.

[22] Z. QIAN, *Crossover recombination-based global-best brain storm optimization algorithm for UAV path planning*, Proceedings of the Romanian Academy Series A-Mathematics Physics Technical Sciences Information Science **23**(2), 2022, pp. 207–216.