Collaborative Framework of Accelerating Reinforcement Learning Training with Supervised Learning Based on Edge Computing

Yu-Shan Lin[1], Chin-Feng Lai[2], Chieh-Lin Chuang[2], Xiaohu Ge[3], Han-Chieh Chao[4]

[1] Department of Information Science and Management Systems, National Taitung University, Taiwan
[2] Department of Engineering Science, National Cheng Kung University, Taiwan
[3] Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China
[4] Department of Electrical Engineering, National Dong Hwa University, Taiwan
ysl@nttu.edu.tw, cinfon@ieee.org, aa770538@gmail.com, xhge@mail.hust.edu.cn, hcc@gms.ndhu.edu.tw

Abstract

In the reinforcement learning model training, it usually takes a lot of training data and computing time to find the law from the environmental response in order to facilitate the convergence of the model. However, edge nodes usually do not have powerful computing capabilities, which makes it impossible to apply reinforcement learning models to edge computing nodes. Therefore, the framework proposed in this study can enable the reinforcement learning model to gradually converge to the parameters of the supervised learning model within the shorter computing time, so as to solve the problem of insufficient terminal device performance in edge computing. Among the experimental results, the operating differences of hardware with different performance and the influence of the network environment and neural network architecture are analyzed based on the Mnist and Mall data sets. The result shows that it is sufficient to load the real-time required by users under the framework of collaborative training, and the time delay pressure on the model is caused by the application of different levels of complexity.

Keywords: Edge computing, Reinforcement learning, Collaborative framework, Supervised learning

1 Introduction

With the development of artificial intelligence technologies, the technologies have been applied in various fields. Also because of the increase in the amount of computing data, there are many hidden layers built for specific needs in the neural network architecture. and this will cause a certain degree of burden on performance. In order to make the neural network model have more real-time response, the concept of edge computing is proposed to solve this problem that a coherent task is able to be completed through distributed collaboration. The terminal device is used to handle tasks that require less performance, and the more computationally intensive part is completed by the cloud device. Although the prerequisite for this is that there must be a good communication state between devices, in the 5G architecture that will gradually take shape in the future. Therefore edge computing can be regarded as an effective method of operation. On the other side, the general supervised learning model has good performance after the model training is completed, but its characteristic is that the weights in the model cannot be updated in real time. When the external input changes, the parameters cannot be updated, and the model is generated iinapplicable issues. If the model needs to be updated, it usually requires a certain degree of retraining, and it is more time-consuming. Under the framework of reinforcement learning, the parameters of the model are mostly dynamically updated, and its update method requires feedback through the external environment. After the agent makes a corresponding decision on the input state, it gets a reward and punishment feedback, and then updates the weight of the neural network. However, in practical applications, not every task can get feedback from the user or the environment. In this case, the problem of reinforcement learning model construction will arise. It is also because on the terminal device, the hardware usually cannot have strong performance, so the model structure usually adopts a simpler design, so it needs efficient cloud computing to assist.

In this research, the concept of edge computing will be referred. A collaborative framework of accelerating reinforcement learning training with supervised learning is proposed. The reinforcement learning model will be used on the terminal device and the supervised learning model with a complex architecture will be built on the cloud server. The purpose is to assist in training the

*Corresponding Author: Chin-Feng Lai; E-mail: cinfon@ieee.org
DOI: 10.3966/160792642021032202001

reinforcement learning model on the terminal device through the supervised learning model in the cloud. The framework proposed in this research includes the following contributions:

1. Analyze independent event tasks using a reinforcement learning framework. At present, most reinforcement learning architectures are mostly continuous tasks. For example, after a reinforcement learning model makes a decision in a game environment, it will get the next state related to this action. This research will use a data set in which each state is an independent event as the main analysis content.
2. Use the pre-trained supervised learning model to assist the convergence of the terminal device model. The event task is divided into the terminal device and the cloud server, and the communication state between the two is maintained through a network connection. In addition, supervised learning is carried out with hardware devices with better performance, so that it can provide rewards and punishments for terminal devices, thereby making up for the defects of task analysis.
3. Design a value function suitable for class classification tasks. In the independently distributed data set, each data is regarded as an independent state. Therefore, a general-purpose value function is designed to approximate the best convergence result of the model.

The remainder of this paper is organized as follows. In ssection 2, the related works is discussedd and a collaborative framework of accelerating reinforcement learning training with supervised learning. Section 4 evaluates the proposed framework performance; finally, the conclusions is given in Section 5.

2 Related Works

2.1 Edge Computing

Edge computing is an emerging distributed computing framework that is designed to share part of the computing workload of cloud servers. The framework decomposes the large-scale services that were originally processed entirely by the central node, and distributes them to the edge nodes for processing. In other words, when the computing load can be shared by the terminal devices, it will reduce the computing load of the cloud server compared to the original centralized computing architecture. This will speed up the calculation and transmission of data, so as to reduce the calculation results. delay. In research [1], an edge node is defined as any node with computing power or network resources in the path of the data source and cloud computing center. For example, a smart phone is an edge device between the user and the cloud server. Therefore, from a definition point of view, edge computing is more biased towards one component of the internet of things. Therefore, edge computing has been applied to more and more areas of life today, such as: smart city, cloud offloading, movie analysis, smart home, etc.

In traditional content delivery networks, only data will be cached on edge servers, so most of the calculations still rely on cloud computing. But this means that users need to wait a longer time, so many applications nowadays have made tradeoffs for cloud performance to solve cloud offloading [2-6]. However, if the edge device has a certain amount of computing power, the edge device can help to share the computing load of the cloud server. The map navigation system is one example that can distribute the task requirements of users to different edge nodes for processing according to different locations of users. In other words, there is no need to load all the maps at the same time, and the system only needs to provide the limited maps that users need in real time. However, this will cause synchronization problems between edge nodes. For example, when a navigation user moves within the service range of different nodes, the service node needs to be switched. This may be an issue in the development of algorithms, and it may even be possible to add machine learning to assist node distribution, but such decentralized operations can effectively improve the delay problem.

One of the most representative applications of edge computing is the camera on the street. Video data captured by the camera is usually not uploaded to the cloud due to traffic costs or privacy security considerations. Therefore, when doing analysis or when there is a special need to search for a specific purpose in the video, if the task is completely handed over to the cloud server, it will generate a huge performance load. The system can send task requirements to the edge nodes that will analyze the task data and then send the result back to the cloud server after the analysis is completed. This approach can make better use of performance and avoid overloading the cloud server.

In the field of machine learning, some large tasks usually have a large computing demand. If a single hardware is used to complete the task, it will cause a large computing load on the cloud server. Therefore, the concept of edge computing has recently begun to be added to the neural network architecture. In research [7], the detection model in the smart production line is split into different parts, and the neural network model is also the same. Separate the neural network layers responsible for different tasks in the complete large structure, and decentralize the neural network layer with simpler tasks to edge nodes with computing capabilities to share the overall amount of computing.

2.2 Reinforcement Learning

The training methods of reinforcement learning, supervised learning and unsupervised learning are

different, because the training methods of reinforcement learning are more instantaneous. When the reinforcement learning model is able to learn and update parameters while collecting data, there is no need to prepare a large amount of training data in advance. Therefore, reinforcement learning is often used for automatic control of equipment that is called Approximate Dynamic Programming (ADP) [8-9].

Therefore, the reinforcement learning model is based on exploration as the main training method, and the learning model is gradually trained in the exploration mode until convergence. The architecture of the reinforcement learning model includes the components "environment" and "agent". During training, the model's agent will act according to the state of the environment and receive rewards, thereby improving the accuracy of decision-making. According to Markov decision processes (MDPs) [10-11], under ideal conditions, the future depends on the present, so it is necessary to evaluate the rewards of actions performed in the action set that is called "policy".

In the training process, each complete MDP is called an episode, and to judge whether a model is properly trained, the rewards obtained for each step in the episode are usually summed up. According to MDPs, the reward and punishment scores for the next decision can be estimated, and the sum of reward and punishment scores can also be estimated, as in formula 1.

$$\begin{aligned} G_t &= R(s_t, a_t) + \gamma R(s_{t+1}, a_{t+1}) + \gamma^2 R(s_{t+2}, a_{t+2}) + \cdots \\ &= R(s_t, a_t) + \gamma[R(s_{t+1}, a_{t+1}) + \gamma R(s_{t+2}, a_{t+2}) + \cdots] \\ &= R(s_t, a_t) + \gamma G_{t+1} \end{aligned} \quad \textbf{(1)}$$

If γ is larger, it means that the expected reward is more important. The cumulative rewards that can be obtained for actions performed in different states by formula 1 are called value. The value must be constantly updated during the reinforcement learning training process to ensure that the decision-making meets the needs of any state in the environment. Table 1 is the general parameters of reinforcement learning and Figure 1 is the reinforcement learning process.

Table 1. Parameters of Reinforcement Learning

Parameter	Description
Set of State	The information that the agent obtains from the environment is called State, and all the states that may be encountered are called the state set, denoted as *S*. The state is marked as s_t, which refers to the state at a certain time or a certain number of iterations.
Set of Actions	All the actions that the agent can do are called action sets, and the agent will make decisions based on the state in the action set. The action set is marked as *A* and the action is marked as a_t, which refers to the action at a certain moment or a certain number of steps.
Reward	The agent transfers to s_t+1 the reward and punishment score obtained after taking an action in state s_t. $R(s_t, a) = R_a(s_t, s_t+1)$
Discount Factor	γis the variable used in estimating the overall reward and punishment score. $\gamma \in (0, 1)$

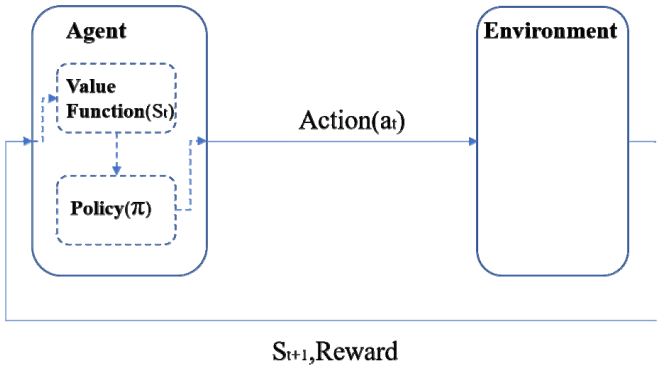

Figure 1. Reinforcement Learning Process

According to Bellman Equation [12], it can be known that in the case of strategy π, the state value function is V π(s), as shown in formula 2.

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[G_t \mid s_t = s] = \mathbb{E}_{\pi}[R(s,a) + \gamma G_{t+1} \mid s_{t+1} = s] \\ &= \sum_{a} \boldsymbol{\pi(a \mid s)} \sum_{s'} \sum_{\gamma} \boldsymbol{p(s', r \mid s, a)[r + \gamma \mathbb{E}_{\pi}[G_t \mid s_t = s']} \\ &= \sum_{a} \boldsymbol{\pi(a \mid s)} \sum_{s'} \sum_{\gamma} \boldsymbol{p(s', r \mid s, a)[r + \gamma V^{\pi}(s')]} \\ &= \mathbb{E}_{\pi}[R(s,a) + \gamma V^{\pi}(s_t + 1) \mid s_t = s] \end{aligned} \quad \textbf{(2)}$$

The strategy π(a|s) represents the probability of taking a certain action a in the s state. The common strategies include greedy algorithm, softmax policy and so on. The value function of the action is also defined as follows.

$$
\begin{aligned}
Q^{\pi}(s,a) &= \mathbb{E}[G_t \mid s_t = s, a_t = a] \\
&= \mathbb{E}_{\pi}[R(s,a) + \gamma G_{t+1} \mid s_t = s, a_t = a] \\
&= \sum_{s'}\sum_{\gamma} p(s', \gamma \mid s, a)[r + \gamma \mathbb{E}_{\pi}[G_{t+1} \mid s_{t+1} = s']] \\
&= \sum_{s'}\sum_{\gamma} p(s', r \mid s, a)[r + \gamma V^{\pi}(s')] \\
&= \mathbb{E}[R(s,a) + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a]
\end{aligned}
\tag{3}
$$

Through formula 1 and formula 2, the value of each state and action can be calculated, and appropriate actions can be taken, usually to find the maximum value of the value function, called the optimal value function.

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{4}$$

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{5}$$

After substituting the greedy algorithm, it can be ignored that the state value of formula 1 is affected by the occurrence probability of the action, so that the occurrence probability of the action with the highest value is 1, and the rest are 0. Therefore, formula 3 can be calculated as follows.

$$
\begin{aligned}
V^{*}(s) &= \max_{a \in A} Q^{*}(s,a) \\
&= \max_{a} \Sigma_{s'} \Sigma_{\gamma} p(s', r \mid s, a)[r + \gamma V^{*}(s')]
\end{aligned}
\tag{6}
$$

$$
\begin{aligned}
Q^{*}(s,a) &= \mathbb{E}[R(s,a) + \gamma V^{*}(s_{t+1} \mid s_t = s, a_t = a)] \\
&= \mathbb{E}[R(s,a) + \gamma \max_{a'} Q^{*}(s_{t+1}, a) \mid s_t = s, a_t = a)] \\
&= \Sigma_{s'} \Sigma_{\gamma} p(s', r \mid s, a)[r + \gamma \max_{a'} Q^{*}(s', a')]
\end{aligned}
\tag{7}
$$

However, the greedy algorithm also has its shortcomings. It does not guarantee that it can find the global best solution. Therefore, in 1989, Christopher J. C. H. Watkins proposed ∈ greedy [13-14], based on the greedy algorithm and adding the concept of exploration, there is a certain probability that actions can be randomly generated. Because the probability depends on ∈, in this case, as long as the number of training is enough, the global optimal solution can be found.

3 Collaborative Framework of Accelerating Reinforcement Learning Training with Supervised Learning

Due to the trend of edge computing in recent years, this research hopes to implement a framework for supervised learning and training of reinforcement learning models, and to conduct collaborative training with reinforcement learning models through network connections. With the general lack of efficiency in the terminal environment, it is impossible to immediately have a good performance at the beginning of the landing. The training method of reinforcement learning is based on the interaction between the agent and the environment. Therefore, it is necessary to calculate the reward and punishment scores to update the parameters through the interaction between the two when actually landing.

In this study, the method of pre-training with supervised learning is selected to train a model that can provide a basis for reward calculation for the reinforcement learning model. The reinforcement learning model can not only grow gradually on low-performance devices, but also operate normally in a large field, and it can also avoid the operating window period caused by the retraining of the supervised model. Therefore, the proposed framework is shown in Figure 2.

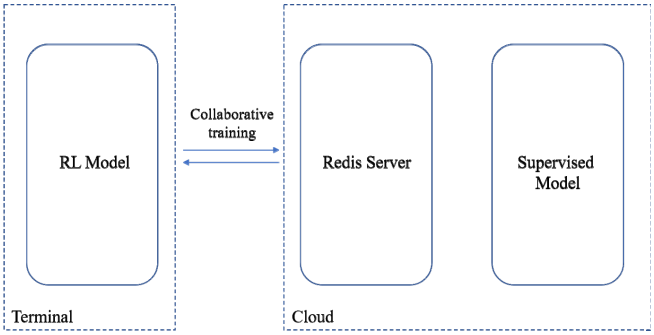

Figure 2. Framework Architecture

3.1 Supervised Learning Model on Cloud Server

The supervised learning model is calculated on the cloud device, and after pre-training, it serves as a source for providing the basis for the calculation of rewards and punishments for reinforcement learning. Before the data is sent to the model for training, it needs to undergo normalization pre-processing, the purpose of which is to make the range of pixel values fall within the range of 0~1. In the deep learning model, the backpropagation algorithm is usually used to calculate the error between the output layer and the hidden layer and update the weight. Therefore, the partial derivative of the calculation error with respect to the weight in the neuron is obtained through the chain lock rate. The method is as follows Formula 8.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} \tag{8}$$

w_{ji} represents the weight between neuron i and j, and net_j is the input of neuron, which is the weighted sum of the output of all previous neurons oi. Similarly, o_j is the output of neuron j. It can be seen from formula 8 that if the neuron is located in the output layer, then

$o_j = y$, then $\frac{\partial E}{\partial o_j}$ can be easily calculated.

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t-y)^2 = y - t \quad \textbf{(9)}$$

Among them, y is the output result of the neural network, and t is the target output, which is the result that the model needs to approximate. It can be found that only net_j is related to w_{ji}. Before this, several items can be obtained from the output layer through the chain rate. It can be known from formula 8 that the weight w of each layer will be correlated when the parameters are updated. Therefore, if the input x is larger, the gradient obtained by the backpropagation algorithm will be larger. And if the gradient is larger, the learning rate needs to be adjusted smaller, then when the x range is larger, the learning rate needs to be adjusted according to the range. In addition, the suitable learning rate for w_1 is not necessarily suitable for w_2, so when the value is normalized and narrowed the range, the learning rate does not have to be adjusted according to the size of the value range, and at the same time, it avoids the learning rate from not being able to be applied to each layer. The question of the weight of the neural network.

3.2 Reinforcement Learning Model on Terminal Device

The reinforcement learning model is applied to the terminal device, so the computing performance is low, and the default is no pre-training. Therefore, the value evaluation model and decision model with fewer layers are used in the configuration, and in principle, the goal is not to use the convolutional layer. In the value evaluation model, only a single-layer fully connected layer network is used to output values and predict the results.

Among the reinforcement learning models in this study, the similarity to the supervised learning model is that the data input model will first go through a value evaluation model. In this study, in order to save energy and energy costs, the convolutional layer is discarded. After the data is input, it is directly sent to the fully connected layer to predict the action value, and the data needs to be pre-processed before entering the overall model. Because the memory of the terminal device configuration in this study is relatively short, if the original size data is used for training, the system will crash. Therefore, after trying various size reductions, it is found that the data can be trained normally after the data is reduced by 0.5 times.

In the way of decision-making, this study chooses $\in$ greedy as the main decision-making method, and defines the value of $\in$ as follows.

$$\varepsilon = \max\{\varepsilon_{\min}, \varepsilon_{\max} - \frac{(\varepsilon_{\max} - \varepsilon_{\min})X_{step}}{total}\} \quad \textbf{(10)}$$

In DQN, the data of each iteration of the agent is put into the memory pool during the training process as candidate data for updating parameters later, and the training parameters of historical training data are extracted from the memory pool after the number of training times reaches the specified number. Update to the model. The difference from the general training task is that this research does not have a continuous relationship between each data in the data set, which means that they are independent of each other. When extracting data from the memory pool for model training, the original value function update method is as in formula 11.

$$\begin{aligned} &Q(s_t, a_t) \leftarrow \\ &Q(s_t, a_t) + a \cdot [R(s_t, a_t) + \gamma_{a \in A}^{\max} Q(s_t, a_t) - Q(s_t, a_t)] \end{aligned} \quad \textbf{(11)}$$

Among the parameters required by the Qlearning formula, one item represents the long-term rewards obtained in the future, and is multiplied by the decay coefficient γ, which is $\gamma_{a \in A}^{\max} Q(s_{t+1}, a)$ in formula 11. When γ is larger, It means that the more important it is to future rewards, and vice versa. Therefore, when each data is independent of each other, the estimated value of the next state is not important. Therefore, in this study, γ is set to 0, which is wrong. Bellman's equation is expanded and formula 12 is derived.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a \cdot R(s_t, a_t) \quad \textbf{(12)}$$

3.3 Redis Server for Collaborative Training

Redis Server provides data exchange services between reinforcement learning models and supervised learning models during collaborative training as shown as Figure 3. Therefore, during the training process, the environment will give the corresponding reward according to the action given by the agent. In this research, the initial definition of Reward Function is to compare the action made by the agent with the label of the data set, and Reward is provided based on the closeness of the two, the closer the two are, the larger the reward will be, and vice versa. After the model's main architecture is changed to collaborative training, the tag data in the terminal device can be removed, freeing up a part of the hardware space, and the cloud device will give its predicted results through a network connection. In order to calculate the Reward, although such an architecture needs to rely on the quality of the network connection, it needs to be performed in a field with a good network status, but in this way we also solve the problem that all training data needs to be labeled. Reward can also be obtained for training when labeling data.

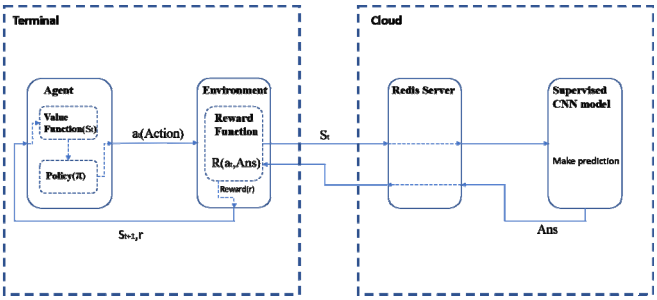

Figure 3. Collaborative Training

If the Reward Function is not well defined, good training results will not be obtained, but the prediction task in this study will not be able to define the Reward Function in this way. In terms of definition, it is hoped that the model can use approximation to gradually approach the maximum value of Reward. As long as the predicted target is closer to the prediction given by the supervised model, the higher the Reward is obtained. In the definition of Reward Function, this study defines it as formula 13.

$$Reward = (-2)X \mid C - action \mid +10 \qquad \textbf{(13)}$$

The variable C represents the predicted value returned by the supervised model, and the action represents the action taken by the reinforcement learning model to the environment.

In this study, the allowable predicted value and the accurate value have an error space of ±5. The predicted value in the error space can get positive feedback, and the larger the difference, the greater the negative feedback. The original design was to predict the hit to have a positive feedback, which is all 0, as shown in formula 14.

$$Reward = \begin{cases} 1 & action = 0 \\ 0 & else \end{cases} \qquad \textbf{(14)}$$

However, it was later discovered that in tasks with more actions, such a reward function approaching a zero-sum game would make it difficult for the model to guess the target at the beginning, resulting in slower model convergence.

4 Experiment Results

There are two data sets used for the experiment. The first data set used in this study is Mall Dataset [15-18], and its data source is a shopping mall camera for crowd counting and analysis. Because the image size of Mall Dataset is large, its content composition is also more complicated, and the task goal is also a bit difficult. Therefore, this research also adds the Mnist handwriting recognition data set to the training task to observe the effect of the image size on the training. The extent of the impact. The Mnist data set is the second data set provided by Yann LeCun and others. It is often used as an introductory project for beginners of machine learning. It provides 55,000 training data and 5,000 verification data. The training materials provided are handwritten digital images ranging from 0 to 9, the image size is 28 × 28, and the label content is also the onehot encoding vector ranging from 0 to 9. Mnist is usually used for supervised learning model training, with classification as task orientation, softmax is usually used as the output layer, and the loss function is CrossEntropy. In this study, the same is changed to a non-classification task, the general fully connected layer is used to directly predict the number, and the MSE is used as the loss function.

The purpose of this experiment is because reinforcement learning has a stronger ability to adapt to the environment than supervised learning, but in the reinforcement learning model, it is necessary to get feedback from the environment. In practice, when setting up a model on a terminal device, you may encounter situations where feedback cannot be obtained. And because the terminal device usually does not have too good performance, it takes a longer time for the model to converge. Therefore, this research hopes to propose a system that assists the convergence of the reinforcement learning model with a supervised learning model to improve the above-mentioned possible problems.

4.1 Mall Dataset

According to formula 13, this experiment sets the upper limit of the reward of a single epoch to 5120 and the lower limit to -56320. When the value evaluation model uses a single-layer fully connected layer, it takes a total of 68:38:49, with an average of 123.56 seconds per epoch and 0.241 seconds per image to complete the analysis.

It can be observed in Figure 4 and Figure 5 that when the number of training reaches 2000 times, the model can converge to a Reward of nearly 400 points, but on the Loss trend graph, there are abnormally rising data in the later period, which is presumed to be over-fitting. The situation arises. In the collaborative training architecture, theoretically the convergence speed will be roughly the same. The only different variable is that the State in the reinforcement learning model, which is the analyzed image, must be sent to the cloud server and wait for the prediction answer to be returned. It will take longer than the first experiment. In Figure 6 and Figure 7, it can be found that the abnormal rise of Loss in the later stage only occurs when the value evaluation model of a single-layer fully connected layer is used for training. It is speculated that the neural network can have sufficient parameters to adjust to avoid over-fitting when the neural network is deep. The situation happened. Different from the expected result, it was not expected that it would take such a long time to transmit the complete picture to the cloud in the collaborative training, so it would take a long time.

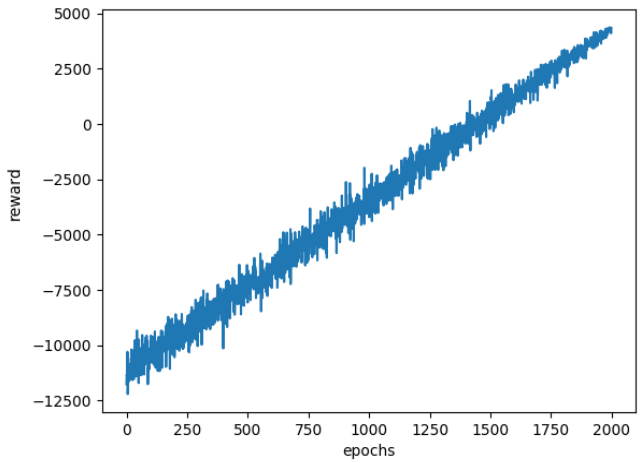

Figure 4. Reward of Neural Network Stand-alone Training

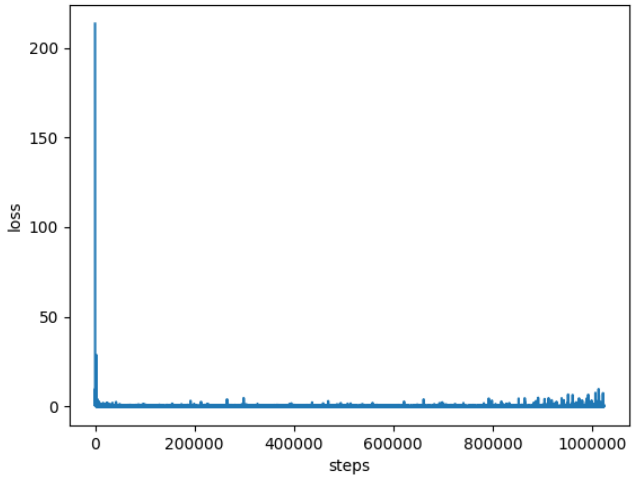

Figure 5. Loss of Neural Network Stand-alone Training

In the case of the value evaluation model using multi-layer convolutional neural network layers, it took a total of 98:51:36. On average, each epoch took 177.95 seconds, and each picture took 0.347 seconds to complete the analysis.

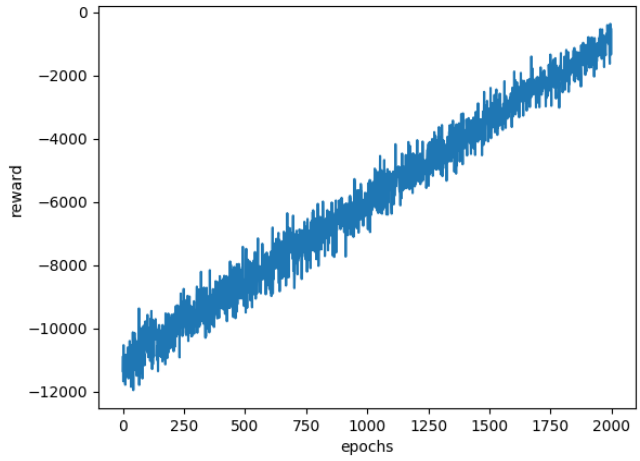

Figure 6. Reward of Neural Network Collaborative Training

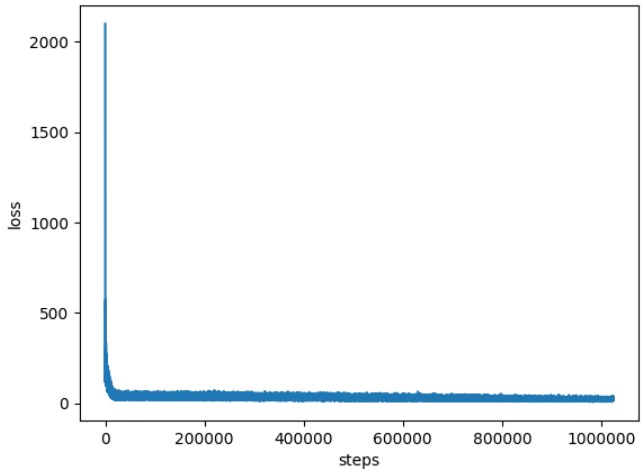

Figure 7. Reward of Neural Network Collaborative Training

4.2 Mnist Dataset

According to formula 14, this experiment sets the upper limit of the reward of a single epoch to 512 and the lower limit to -512. When the value evaluation model uses a single-layer fully connected layer, it takes a total of 4:22:31, with an average of 7.88 seconds per epoch, and 0.015 seconds per picture to complete the analysis.

From Figure 8 and Figure 9, it can be seen that the reward obtained by the model as a whole is gradually increasing, which means that the model can converge, but when the number of training times reaches 2000, the converged score has not yet reached a positive number, which represents this simple The neural network architecture is indeed slow in convergence speed. In Figure 10 and Figure 11, the training time is much longer than other experiments. The growth trend of Reward is roughly the same, but the training time is nearly 4 times that of it. This is because every graph in the Mall Dataset is (640, 480), even if we resize it to (320, 240), it is still not a small size, so it takes a lot of time to transmit on the network. But this is from the point of view of the complete training process. If you look at a single sheet, it only takes 0.858 seconds for each analysis. Therefore, in general, it should be enough to load in practice.

In the case of the value evaluation model using a single-layer fully connected layer and co-training, it takes a total of 6:34:02, an average of 11.82 seconds per epoch, and 0.023 seconds per picture to complete the analysis.

5 Conclusion

This research proposes a method of collaborative training with a supervised learning model assisted by a reinforcement learning model, and refers to the concept of edge computing to design different performance scenarios for training. According to the results of

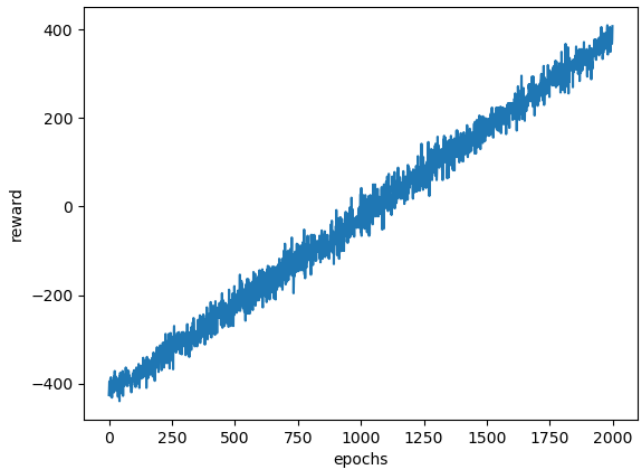

Figure 8. Reward of Neural Network Stand-alone Training

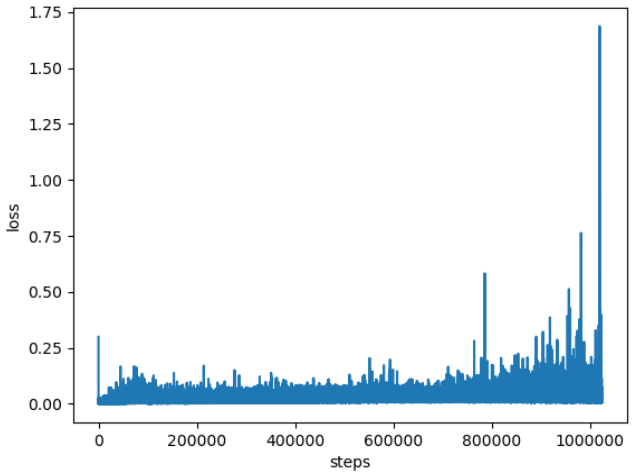

Figure 9. Loss of Neural Network Stand-alone Training

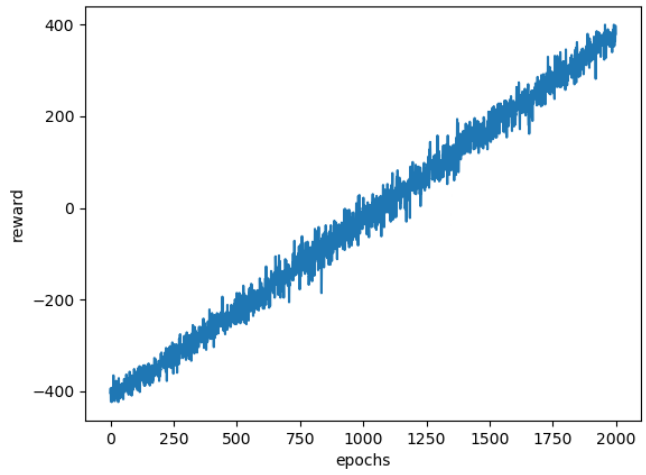

Figure 10. Reward of Neural Network Collaborative Training

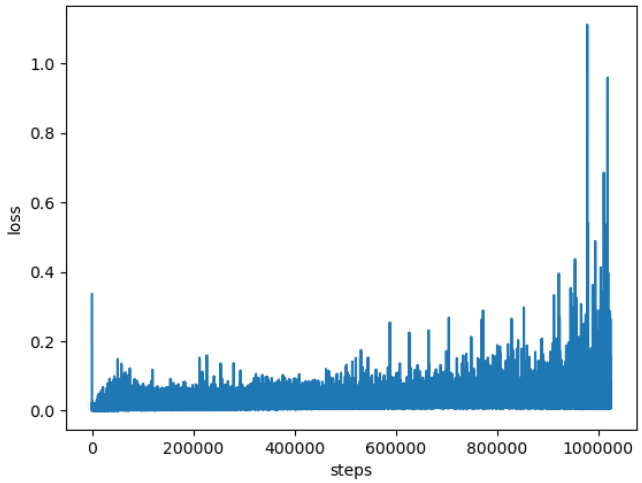

Figure 11. Reward of Neural Network Collaborative Training

model convergence, the contributions of this research can be sorted into the following points.

1. Solve the problem of the actual implementation of the reinforcement learning model when there is no feedback. The prediction result of the supervised model is used as the basis for the Reward calculation required by the reinforcement learning model to solve the problem that the reinforcement learning model cannot update the parameters when it is difficult to obtain feedback.

2. Improve the problem that the terminal device cannot converge and perfect when the performance is low. Because the performance of the terminal device usually does not have very good performance, if the supervised learning model is used, it takes a long time to converge. The supervised learning model cannot continue to make predictions during training, which will cause the system to stall. This research uses a reinforcement learning model and a supervised learning model for collaborative training, so that the reinforcement learning model can gradually converge to the parameters of the supervised learning, and the continuous operation of the system makes the model converge complete.

3. Solve the problem that the system needs to shut down when the model updates its parameters. When the environment changes, the supervised learning model will usually be disabled. Although retraining can use some parameters of the old model for transfer learning, the system still needs to be shut down. The reinforcement learning model can continuously output results. In the collaborative training, when the supervised learning model is retraining or updating parameters, the method of returning pseudo feedback can be used to make the reinforcement learning model free from the suspension problem of model parameter update.

Acknowledgments

Xiaohu Ge would like to acknowledge the support from the National Key Research and Development Program of China under Grant 2017YFE0121600. Han-Chieh Chao would like to acknowledge the support from Taiwan Ministry of Science and Technology under Grant 107-2221-E-259-005-MY3.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, Edge computing: Vision and challenges, *IEEE internet of things journal*, Vol. 3, No. 5, pp. 637-646, October, 2016.

[2] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, Clonecloud: elastic execution between mobile device and cloud, in *Proceedings of the sixth conference on Computer systems*, Salzburg, Austria, 2011, pp. 301-314.

[3] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, Saving portable computer battery power through remote process execution, *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 2, No. 1, pp. 19-26, January, 1998.

[4] G. Hunt and M. Scott, The coign automatic distributed partitioning system, in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, New Orleans, Louisiana, USA, 1999, pp. 187-200.

[5] K. Kumar and Y. H. Lu, Cloud computing for mobile users: Can offloading computation save energy?, *Computer*, Vol. 43, No. 4, pp. 51-56, April, 2010.

[6] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in *2012 Proceedings IEEE Infocom*, Orlando, FL, USA, 2012, pp. 945-953.

[7] L. Li, K. Ota, and M. Dong, Deep learning for smart industry: Efficient manufacture inspection system with fog computing, *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, pp. 4665-4673, October, 2018.

[8] D. P. Bertsekas, *Dynamic programming and optimal control*, Vol. 1, Athena scientific Belmont, MA, 1995.

[9] L. Buşoniu, B. De Schutter, and R. Babuška, Approximate dynamic programming and reinforcement learning, in: R. Babuška, F. C. A. Groen (Eds.), *Interactive collaborative information systems*, Springer, 2010, pp. 3-44.

[10] R. Bellman, A markovian decision process, *Journal of mathematics and mechanics*, Vol. 6, No. 5, pp. 679-684, 1957.

[11] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2014.

[12] R. Bellman, Dynamic programming, *Science*, Vol. 153, No. 3731, pp. 34-37, July, 1966.

[13] C. J. C. H. Watkins, *Learning from delayed rewards*, Ph. D. Thesis, King's College, Cambridge, 1989.

[14] C. J. Watkins and P. Dayan, Q-learning, *Machine learning*, Vol. 8, No. 3-4, pp. 279-292, May, 1992.

[15] C. C. Loy, S. Gong, and T. Xiang, From semi-supervised to transfer counting of crowds, in *Proceedings of the IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 2256-2263.

[16] K. Chen, S. Gong, T. Xiang, and C. C. Loy, Cumulative attribute space for age and crowd density estimation, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Portland, OR, USA, 2013, pp. 2467-2474.

[17] C. C. Loy, K. Chen, S. Gong, and T. Xiang, Crowd counting and profiling: Methodology and evaluation, in: S. Ali, K. Nishino, D. Manocha, M. Shah (Eds.), *Modeling, simulation and visual analysis of crowds*, Springer, 2013, pp. 347-382.

[18] K. Chen, C. C. Loy, S. Gong, and T. Xiang, Feature mining for localised crowd counting, in *Proceedings of British Machine Vision Conference (BMVC)*, Vol. 1, Surrey, UK, 2012, pp. 21.1-21.11.

Biographies

Yu-Shan Lin is an associate professor in the Department of Information Science and Management Systems, National Taitung University, Taitung, Taiwan. She received Ph.D. degree of Business Management from National Sun Yat-sen University, Kaohsiung, Taiwan, majoring in Marketing. Her research interesting areas include e-Learning, Precision education, Internet marketing, Tourism marketing, and Tourism Consumer behavior. She has published about 20 journal papers, and some are highly-cited papers. Moreover, she received the Best Paper Award from IEEE ICASI 2018 and IC3 2018.

Chin-Feng Lai is a professor at Department of Engineering Science, National Cheng Kung University and Department of Computer Science and Information Engineering, National Chung Cheng University since 2016. He received the Ph.D. degree in Department of Engineering Science from National Cheng Kung University, Taiwan, in 2008. He received Best Paper Award from IEEE 17th CCSE, 2014 International Conference on Cloud Computing, IEEE 10th EUC, IEEE 12th CIT . He has more than 100 paper publications and 9 papers selected to TOP 1% most cited articles by Essential Science Indicators (ESI). He is an associate editor-in-chief for Journal of Internet Technology and serves as editor or associate editor for IET Networks, International Journal of Internet Protocol Technology, KSII Transactions on Internet, Information Systems and Journal of Internet Technology. His research focuses on Internet of Things, Big Data Analysis and Edge Computing etc. He is an IEEE Senior Member since 2014.

Chieh-Lin Chuang received the master degree in Department of Engineering Science, National Cheng Kung University in 2020. He has been working in Taiwan Semiconductor Manufacturing, Taiwan for AIoT product developments.

Xiaohu Ge received the Ph.D. degree in communication and information engineering from the Huazhong University of Science and Technology (HUST), China, in 2003. He has been worked with HUST, since November 2005. Prior to that, he worked as a Researcher at Ajou University, South Korea, and the Politecnico Di Torino, Italy, from January 2004 to October 2005. He is currently a full Professor with the School of Electronic Information and Communications, HUST. He is also an Adjunct Professor with the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Australia. He has published about 200 articles in refereed journals and conference proceedings. He has been granted about 25 patents in China. His research interests are in the area of mobile communications, traffic modeling in wireless networks, green communications, and interference modeling in wireless communications. He services as an IEEE Distinguished Lecturer and an Associate Editor for the IEEE Access, the IEEE Wireless Communications and the IEEE Transactions on Vehicular Technology, etc.

Han-Chieh Chao received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1993, respectively. He is currently a Professor with the Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan, where he also serves as the President. He is also with the Department of Computer Science and Information Engineering and the Department of Electronic Engineering, National Ilan University, Yilan, Taiwan, the College of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan, China, and the Fujian University of Technology, Fuzhou, China. He serves as the Editor-in- Chief for the Institution of Engineering and Technology Networks, the Journal of Internet Technology, the International Journal of Internet Protocol Technology, and the International Journal of Ad Hoc and Ubiquitous Computing. He is a Fellow of the Institution of Engineering and Technology (Institution of Electrical Engineers) and a Chartered Fellow of the British Computer Society.