IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 18, 2021, accepted August 23, 2021, date of publication August 24, 2021, date of current version August 31, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3107719

Successive Over Relaxation Recurrent Confidence Inference Network Based on Linear Extrapolation

WENKAI HUANG[1], YIHAO XUE[2], ZEFENG XU[2], AND LINGKAI HU[2]

[1]Center for Research on Leading Technology of Special Equipment, School of Mechanical and Electrical Engineering, Guangzhou University, Guangzhou 510006, China

[2]School of Mechanical and Electrical Engineering, Guangzhou University, Guangzhou 510006, China

Corresponding author: Wenkai Huang (16796796@qq.com)

This work was supported by Guangzhou Science and Technology Planning Project under Grant 202002030279.

ABSTRACT It is very important to be able to deduce an unknown conclusion from one or several known premises when solving a logical inference. The existing inference methods or models have certain logical inference abilities. However, because of the diversity of the forms of problems and the complexity of the derivation process, the scope of applying these methods is limited; this means the inference results are not ideal. Therefore, this paper proposes a new neural network model to solve the logic inference problem found in calculus. By using the successive over relaxation (SOR) method and the principle of recurrent confidence, the recurrent confidence inference network (RCI-Net) is built to solve the inference problem. The network simulates the solving process of the inference problem. Based on the known premise of this problem, it is calculated step by step so that the result of the calculation becomes gradually closer to the answer. At the same time, to make RCI-Net have stronger logical inference ability, our team uses the half mean squared error (HMSE) to construct the loss function of the model, improving the training efficiency of the model and preventing training collapse caused by the loss value exceeding the system's value range. Our team takes Sudoku reasoning problem as an example to carry out experiments. The results show that when the number of prompts of the reasoning problem is 17, the accuracy of the test set model can reach 99.67%, which is 3.07% higher than the existing models. It proves that the algorithm has better effect than the existing methods in solving logical reasoning problems.

INDEX TERMS Calculating logical inference, successive over relaxation, recurrent confidence, half mean squared error, deep learning.

I. INTRODUCTION

Calculus' logic inference is the thinking process of deriving a proposition from some facts and propositions according to logic rules. Solving all the various kinds of logical inference problems within the realm of calculus can help people correctly understand the nature and relationship between the numbers in propositions, help people analyze problems in an orderly and well-grounded way by using the logical inference methods related to thinking forms, make appropriate judgments, and carry out logical inference. How to deduce an unknown conclusion from existing known premises is the key to solving the problem of logical inference. Most mathematical inference processes are based on formal logic; the main purpose is to get the unknown knowledge from the known knowledge, especially the unknown knowledge that cannot be obtained through sensory experience; this makes the inference process of calculus logic more diverse and uncertain. In the process of this inference, when there is a change in the types of problems and known conditions, the workload for solving the inference problem will be doubled, and more time and energy will be required. Therefore, using deep learning method to solve reasoning problem is a hot research field. The trained neural network model that can automatically and reasonably assist people to solve logical inference problems will greatly improve people's work efficiency and greatly reduce the burden in solving logic inference problems.

There have been some algorithms developed to solve the various logic inference problems; here, some of these algorithms have been based on deep learning [1]. Reference [2] proposed a method to solve the uncertain maldistributed decision-making problem, which contains both uncertain

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Liu.

attribute values and uncertain attribute weights so that the evidential inference method can deal with the change of uncertain attribute weights in an appropriate way. Reference [3] proposed that message passing methods can be used to solve the logic inference problem of calculus, and reference [4] proposed an algorithm to solve the logic inference problem by using the concept of dual quantum computing, thus providing a theoretical framework for the application of binary quantum computing. In reference [5], the improved particle swarm optimization rule and a well-designed mutation operator can be used to solve the logic inference problem. Reference [6] introduces recurrent relational networks and achieves good results in logic inference. In reference [7], a differentiable maximum satisfaction solver is introduced and integrated into the loop of a larger deep learning system; here, the experiments show that the model can be used as a solution for the logical inference problem. Reference [8] uses relation networks to solve the problem of logical inference in calculus. Based on eliminating the computational burden of logical inference and reducing the complexity of the whole network, the ability to infer the relationship between entities and their attributes can be obtained. In reference [9], the optimization problem is integrated into a single layer of a neural network, and a sensitivity analysis, double-layer optimization, and implicit differentiation techniques are used to accurately distinguish these layers from the layer parameters; the experimental results show that this method can effectively solve the problem of logic inference. Based on previous studies on Hopfield neural networks [11], reference [10] uses Hopfield neural networks to solve the problem of logical inference or to solve constraint problems. Reference [12] draws lessons from the principle of spinning neural networks [13] and verifies that the method can effectively solve the logic inference problem by solving a Sudoku puzzle and map color problem. One study [14] uses an artificial neural network to solve the inference problem and verifies the effectiveness of the model in the logic inference of calculus through a Sudoku inference experiment.

All the above models have certain effects on solving inference problems and can obtain good results for simple calculus logic inference problems [15]. However, for more complex inference problems, where there is a decrease in the number of prompts and increase in the range of items to be inferred, many of the above networks have problems such as slow convergence, disappearance of gradients in iterative learning, and training crash caused by loss values exceeding the range of system values. Thus, the efficiency of model training and the accuracy of solving inference problems are reduced.

In order to solve the shortcomings of the current reasoning model, a more reliable reasoning model is established to help people solve the problem of calculus logic reasoning. In the current paper, we use the principle of successive over relaxation (SOR) [16] and recurrent confidence [17] to build a recurrent confidence inference network (RCI-Net) that can solve more complex inference problems. In the present paper, each cycle structure in RCI-Net is composed of convolution layers. The model uses the SOR method to learn iteratively, hence reducing the deviation of each unknown value in the logic inference problem, that is, reducing the difference between the unknown value and the correct solution, and automatically updating the parameters of the convolution kernel in the network. In the current paper, the network adopts a cyclic progressive structure and carries out the progressive operation to solve the logic inference problem of calculus. Similar to classical convolution neural networks such as deep convolutional neural networks (CNN) [18], recurrent CNN [19], encoder–decoder CNN [20], and multicrop CNN [21], RCI-Net also updates the parameters of the internal convolution kernel automatically through iterative learning. However, when the classical neural network uses the traditional Softmax [22] cross entropy as the training loss function, the model will collapse because of the loss value exceeding the system's value range, so the constraint of a half mean squared error (HMSE) is added to improve the ability of the whole network to deal with logical details and to prevent the training collapse caused by the loss value exceeding the system value range, hence improving the training efficiency of the model. The current paper takes the Sudoku inference problem as an example to conduct experiments. RCI-Net can learn the mapping between the known premise and the inference answer in the Sudoku inference problem data set, along with the characteristics of the Sudoku inference problem, to accurately deduce the answer to the number inference problem.

The proposed method has the following advantages: (1) the construction of the RCI-Net model is based on the SOR method and recurrent confidence, so a recursive structure is adopted in the model, on which the problem of calculus logic inference is solved step by step, and the SOR method is integrated to reduce the deviation of each unknown value in the logic inference problem, that is, to reduce the difference between the unknown value and correct solution. Based on this, two successive replacement steps are used for linear extrapolation to improve the model solution and accuracy of solving problems. (2) Based on the prior knowledge of reasoning task rules, this paper designs a suitable neural network structure to improve the performance of neural network. This method is also suitable for other logical reasoning problems. (3) When the traditional Softmax cross entropy is used as the training loss function, the model may lead to a collapse in the training because the loss value exceeds the system value range. Therefore, we use HSME to construct the loss function of the RCI-Net model, which not only completely prevents a training collapse, but also maintains the high accuracy of the model. (4) After training, the RCI-Net model can be used independently, which can effectively solve the Sudoku inference problem and extend the application of this model to other logic inference problems. (5) In the current paper, the SOR method and recurrent confidence principle are integrated into the convolution model for the first time, and the HMSE function is used as a part of the loss function in the RCI-Net model; this means that prior knowledge is added,

which greatly improves the convergence speed and inference effect of the network.

II. STRUCTURE OF RECURSIVE CONFIDENCE INFERENCE NETWORK

The RCI-Net proposed in this paper is suitable for many complex logic reasoning problems. It uses SOR method to construct the neural network into a continuous multi-layer loop structure, and designs the suitable internal structure of the loop body based on the prior knowledge of the reasoning problem, then uses HMSE as the loss function of training, and finally uses the method of recurrent confidence to realize distributed reasoning. Because the solution to the Sudoku problem integrates a variety of logic inference methods, such as the exclusion method, hypothesis method, absurdity method, and so on, it has certain representativeness. Therefore, the current paper takes solving the Sudoku inference problem as an example to verify the RCI-Net model.

A. NETWORK CONSTRUCTION

In the present paper, we use the SOR method to reduce the deviation of each unknown value in the logic inference problem, that is, to reduce the difference between the unknown value and the correct solution. Based on this, two successive replacement steps are used for linear extrapolation to improve the accuracy of solving inference problems. The RCI-Net model contains a convolution layer, cyclic convolution layer, and anticonvolution layer with different step sizes. The work of feature extraction and logical inference is completed in iterative learning. To improve the overall stability of the RCI-Net model, except for the output convolution layer and input convolution layer, other convolution layers are added with the instance normalization layer [24] as the normalization function of the model. To speed up the convergence of the model, a convolution layer is used instead of a pooling layer. In the learning process of the model for inference problems, the known items and conclusions of the inference problem are input into the RCI-Net model for training so that the RCI-Net model can learn the mapping relationship among the known items, structure, and conclusions of the inference problem. After the error between the verification result of the model and the label of the verification set has been calculated, the model parameters can be adjusted according to the gradient descent method to achieve the goal of model autonomous learning. Finally, we tested the RCI-Net model with the test set and obtained the accuracy rate of the RCI-Net model for solving the Sudoku inference problem. The overall structure of the proposed RCI-Net model is shown in Figure 1, including the convolution layer and cyclic convolution layer. In the training process, we need to input the known premise, structure, and conclusion of the inference problem into the RCI-Net model or training and use the verification set to adjust the parameters of the model so that the model can obtain more accurate answers to the inference problems. The RCI-Net model combines the recurrent confidence principle with a deep convolution network. The internal structure of the model is shown in Figure 1.

In the RCI-Net model, we use instance normalization [24] as the normalization function and leaky ReLU as the activation function of the model. As shown in Figure 1, the RCI-Net model consists of a convolution layer and convolution loop. After the input layer, the convolution layer expands the number of feature graphs in the hidden layer to m_x. Each convolution loop is composed of an RCIconv layer. During the convolution cycle, the output data of the RCIconv layer are re-inputted into the same RCIconv layer. After y times of the convolution cycle, the data are input into the next loop body, where y depends on the boundary length of the smallest module in the Sudoku data set. After N loop operations, the feature map segmented along the last dimension of the hidden layer is used as the output of the neural network system. In the RCI-Net model, we hope to maintain the consistency of data distribution of the input tensor and output tensor in each loop body through a convolution loop structure. Some of the output characteristic graph of the convolution loop body represents the prediction of the result by the convolution loop body, and other characteristic graphs are used to save the intermediate state information. The next loop body makes further inferences based on the prediction and state information from the previous loop.

In addition, each convolution loop has the same parameter structure and shape. Therefore, in layer-by-layer training, after obtaining the parameters of the current fully trained cyclic body, the parameters can be used as the initialization parameters of the next untrained convolution loop. This operation makes the neural network fully trained, greatly improving the training efficiency.

B. RCICONV STRUCTURE IN THE MODEL

Based on the prior knowledge of Sudoku rules, each row, column and 3×3 region of Sudoku contains the logic information of its own specific regions, which is independent from other regions. However, when a deep convolution neural network is used for a Sudoku inference, the input numbers of the convolution kernel may come from different 3×3 regions or from the edge padding area of the hidden layer, as shown in Figure 2. This is contrary to the prior knowledge of Sudoku solution, cross region convolution operation will interfere with the operation results, and will cause a negative impact, resulting in the final solution's accuracy being reduced.

The RCIconv layer used in the RCI-Net model can avoid the above problems. The RCIconv layer that we designed has a certain degree of relational inference ability. It is a combination of multiple-step convolution and deconvolution. In the RCIconv layer, the SOR principle is used to gradually reduce the deviation of each unknown value; that is, to reduce the difference between the unknown value and the correct solution in Sudoku inference problem. Based on this, a linear extrapolation of two successive substitution steps is used. To visually describe the RCIconv layer, the structure is visualized, as shown in Figure 3.

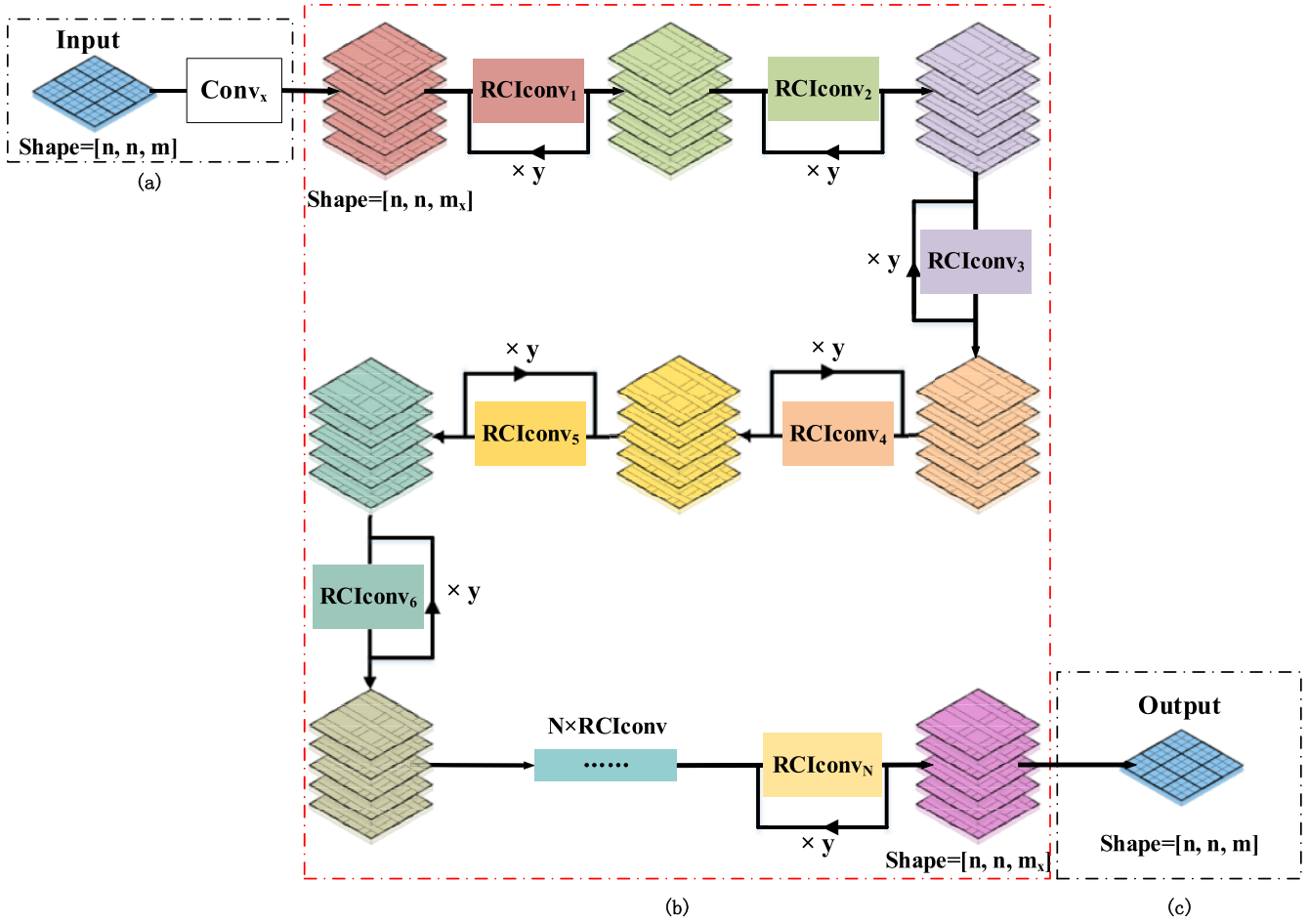

FIGURE 1. The specific structure of RCI-Net.

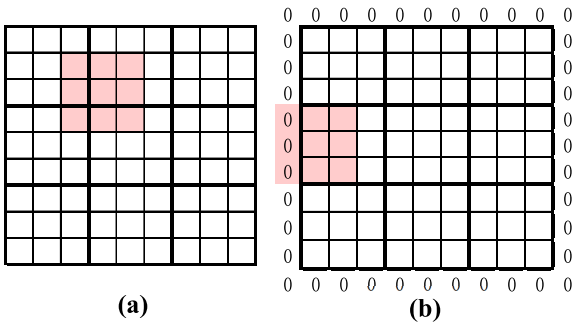

FIGURE 2. In (a), the input numbers of the convolution kernel come from different regions; in (b), the input numbers of the convolution kernel come from the edge padding area of the hidden layer.

In Figure 3, the arrows in the figure indicate the input–output relationship of each convolution layer, and the convolution operation is shown in equation (1).

$$o = \left[\frac{i + 2p - k}{s} + 1 \right] \tag{1}$$

In equation (2), the parameter i represents the input size of the convolution layer, k represents the kernel size, s represents the size of the stripe in the convolution layer, p represents the region where the input boundary is expanded, and o is the final output size. The deconvolution operation in the RCI-Net model is shown in equation (2).

$$o = s\,(i - 1) + 2p - k + 2 \tag{2}$$

As shown in Figure 5, the RCIconv layer in the RCI-Net model is composed of a convolution layer and an anticonvolution layer. According to the logic and principle of solving the Sudoku problem, the Sudoku problem is calculated step by step to improve the solving accuracy. The parameters of each convolution layer and anticonvolution layer in Figure 5 are shown in Table 1.

This study attempts to simply and effectively determine the impact of a single Sudoku sample added to the sample set on the training results. On the one hand, when a new sample appears, it can be optimized on the original training results without starting over; On the other hand, let the new Sudoku training samples into the sample set one by one, so as to

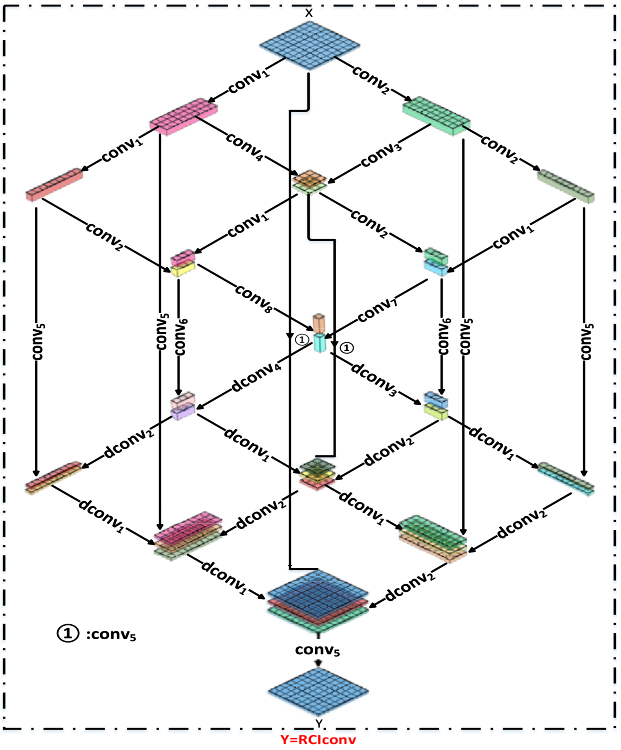

FIGURE 3. The visualization image of the RCIconv structure in this model.

simplify the optimization process and improve the speed of the algorithm.

In this model, we use two successive substitution steps of the linear extrapolation method to achieve SOR and to gradually reduce the deviation of the unknown value in the Sudoku inference problem. Here, deviation refers to the difference between the unknown value and the correct solution in the Sudoku inference problem. This method can solve the Sudoku problem with a size of $n \times n$. Therefore, the current paper assumes that the equation of vector x^n in the kth iteration of the model is as follows:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ik}x_k = b_i \quad (i = 1, 2, \cdots, ; k = 0, 1, 2, \cdots) \tag{3}$$

Thus, the iterative formula of the SOR iteration of the model is shown in equation (4), where ω represents the relaxation factor.

$$x_i^{k+1} = \frac{\left\{\omega b_i + (1-\omega)a_{ii}x_i^{(k)} - \omega\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)} - \omega\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right\}}{a_{ii}} \quad (i = 1, 2, \cdots, ; k = 0, 1, 2, \cdots) \tag{4}$$

In this study, SOR iteration method is applied to model training, where the value of relaxation factor will directly affect the convergence and convergence speed of the algorithm. In order to ensure the convergence of the iteration process, $0 < \omega < 2$ must be required.

TABLE 1. Parameters of each convolution layer and deconvolution layer in the RCIconv layer.

Name	*Form*	*Number of filters*	*Filter size*
***conv*$_1$**	convolution	2	(1, 3)
***conv*$_2$**	convolution	2	(3, 1)
***conv*$_3$**	convolution	1	(1, 3)
***conv*$_4$**	convolution	1	(3, 1)
***conv*$_5$**	convolution	1	(1, 1)
***conv*$_6$**	convolution	2	(1, 1)
***conv*$_7$**	convolution	4	(1, 3)
***conv*$_8$**	convolution	4	(3, 1)
***dconv*$_1$**	deconvolution	1	(1, 3)
***dconv*$_2$**	deconvolution	1	(3, 1)
***dconv*$_3$**	deconvolution	2	(1, 3)
***dconv*$_4$**	deconvolution	2	(3, 1)

C. HALF MEAN SQUARED ERROR LOSS FUNCTION

In the early experimental stage, training collapse often occurs in the training process. This occurs because of using Softmax cross entropy as the loss function of the model. The Softmax cross entropy loss function is as follows:

$$y_i' = \frac{e^{z_i}}{\sum_{k=1}^{m} e^{z_k}} \tag{5}$$

$$loss\left(y_i, y_i'\right) = -y_i \cdot \log(y_i') \tag{6}$$

$$L\left(y, y'\right) = \frac{1}{m \cdot n} \cdot \sum\nolimits_{j=1}^{n} loss\left(y_j, y_j'\right) \tag{7}$$

where z is the output of the neural network, y' is the prediction value of the neural network, y is the tag value, and the Softmax function is output y_j' and loss $L\left(y, y'\right)$ for z_i produces a partial derivative.

$$\frac{\partial y_j'}{\partial z_i} = \begin{cases} \dfrac{-e^{z_j} \cdot e^{z_i}}{\left(\sum_{k=0}^{m} e^{z_k}\right)^2}, & (j \neq i) \\ \dfrac{(-e^{z_i} + \sum_{k=1}^{m} e^{z_i})e^{z_i}}{\left(\sum_{k=1}^{m} e^{z_i}\right)^2}, & (j = i) \end{cases} = \begin{cases} -y_j' \cdot y_i', & (j \neq i) \\ (1 - y_i') \cdot y_i', & (j = i) \end{cases} \tag{8}$$

$$\begin{aligned}\frac{\partial L\left(\mathrm{y}, \mathrm{y}'\right)}{\partial \mathrm{z_i}} &= \frac{1}{\mathrm{m}\cdot\mathrm{n}}\cdot\sum\nolimits_{\mathrm{j}=1}^{\mathrm{n}}\left(\frac{\partial \mathrm{loss}\left(\mathrm{y_j}, \mathrm{y}'_{\mathrm{j}}\right)}{\partial \mathrm{y}'_{\mathrm{j}}}\cdot\frac{\partial \mathrm{y}'_{\mathrm{j}}}{\partial \mathrm{z_i}}\right)\\ &= \frac{1}{\mathrm{m}\cdot\mathrm{n}}\cdot\sum\nolimits_{\mathrm{j}=1}^{\mathrm{n}}\left(-\frac{\mathrm{y_j}}{\mathrm{y}'_{\mathrm{j}}}\cdot\frac{-\mathrm{e}^{\mathrm{z_j}}\cdot\mathrm{e}^{\mathrm{z_i}}}{\left(\sum_{\mathrm{k}=0}^{\mathrm{m}}\mathrm{e}^{\mathrm{z_k}}\right)^2}\right)\\ &\quad -\frac{\mathrm{y_i}}{\mathrm{y}'_{\mathrm{i}}}\cdot\frac{\mathrm{e}^{\mathrm{z_i}}\cdot\mathrm{e}^{\mathrm{z_i}}}{\left(\sum_{\mathrm{k}=1}^{\mathrm{m}}\mathrm{e}^{\mathrm{z_k}}\right)^2}\\ &\quad -\frac{\mathrm{y_i}}{\mathrm{y}'_{\mathrm{i}}}\cdot\frac{\left(-\mathrm{e}^{\mathrm{z_i}}+\sum_{\mathrm{k}=1}^{\mathrm{m}}\mathrm{e}^{\mathrm{z_i}}\right)\mathrm{e}^{\mathrm{z_i}}}{\left(\sum_{\mathrm{k}=1}^{\mathrm{m}}\mathrm{e}^{\mathrm{z_i}}\right)^2}\\ &= \frac{1}{\mathrm{m}\cdot\mathrm{n}}\cdot\sum\nolimits_{\mathrm{j}=1}^{\mathrm{n}}\left(-\frac{\mathrm{y_j}}{\mathrm{y}'_{\mathrm{j}}}\cdot\left(-\mathrm{y}'_{\mathrm{j}}\cdot\mathrm{y}'_{\mathrm{i}}\right)\right)\\ &\quad -\frac{\mathrm{y_i}}{\mathrm{y}'_{\mathrm{i}}}\cdot\left(\mathrm{y}'_{\mathrm{i}}\cdot\mathrm{y}'_{\mathrm{i}}\right)-\frac{\mathrm{y_i}}{\mathrm{y}'_{\mathrm{i}}}\cdot(1-\mathrm{y}'_{\mathrm{i}})\cdot\mathrm{y}'_{\mathrm{i}}\\ &= \frac{1}{\mathrm{m}\cdot\mathrm{n}}\cdot\sum\nolimits_{\mathrm{j}=1}^{\mathrm{n}}\left(\mathrm{y_j}\cdot\mathrm{y}'_{\mathrm{i}}\right)-\mathrm{y_i}\\ &= \frac{\mathrm{y}'_{\mathrm{i}}-\mathrm{y_i}}{\mathrm{m}\cdot\mathrm{n}} \end{aligned} \tag{9}$$

According to equation (9), when using Softmax cross entropy as the loss function, z_i will infinitely approach $+\infty$ or $-\infty$. Because the input and label of the RCI-Ne model are one hot discrete data, some data have a clear mapping relationship, so z_i tends to $+\infty$ or $-\infty$ faster, and y'_i is very close to 1 or 0. So when y'_i tends to zero and $y_i = 1$, the loss value can easily exceed the numerical range of the system, resulting in a collapse of the training.

Because the Sudoku solution system is a cyclic structure, the output of the neural network needs to be taken as the input of the same neural network in the operation process, and the cycle is carried out. Therefore, to ensure the stability of the training, the selected loss function should make the input and output of the neural network maintain a similar data distribution. When the Softmax cross entropy is used as the loss function, the output value Z cannot be used as the input of the next cycle because of its large absolute value. If the Softmax function output y' is used as the input of the next cycle, the input and output of the neural network can keep a similar data distribution, but the gradient in the training easily approaches 0, which makes the training difficult. Below is the mathematical proof.

Suppose that the output of a certain RCIconv layer is z, input Softmax function to get y', and then input y' into the SDconv structure of the next layer. Suppose that the gradient of loss value for $[y'_1, y'_2, \cdots, y'_m]$ is $[a_1, a_2, \cdots, a_m]$, and the gradient of loss value for z_i can be obtained from equation (10).

$$\begin{aligned}\frac{\partial Loss}{\partial z_i} &= a_i\cdot y'_i-\sum\nolimits_{j=0}^{m}\left(a_j\cdot y'_j\cdot y'_i\right)\\ &= y'_i\left(a_i-\sum\nolimits_{j=0}^{m}\left(a_j\cdot y'_j\right)\right)\end{aligned} \tag{10}$$

According to formula (10), when y'_i is close to 0, the gradient of z_i is close to 0. However, when y'_i is close to 1, this is because of:

$$\begin{cases}\sum_{j=0}^{m} y'_j = 1\\ y'_j > 0\end{cases} \tag{11}$$

So:

$$\sum\nolimits_{j=0}^{m}\left(a_j\cdot y'_j\right)\approx a_i \tag{12}$$

$$\frac{\partial Loss}{\partial z_i} = y'_i\left(a_i-\sum\nolimits_{j=0}^{m}\left(a_j\cdot y'_j\right)\right)\approx y'_i\left(a_i-a_i\right)=0 \tag{13}$$

That is, when y'_i is close to 1, the gradient of z_i is also close to 0. To sum up, if the output y' of the Softmax function is used as the input of the next cycle, the gradient will disappear easily in the training, which makes the training difficult. Therefore, our team uses the HMSE loss function instead of the traditional Softmax loss function to solve the above problems.

The function form of the new HMSE loss function proposed by our team is shown in equation (14):

$$L\left(y, z\right) = \frac{1}{n}\sum\nolimits_{i=1}^{n}\Big((1-y_i)\cdot\max(0, z_i)^2+y_i\cdot min\left(0, z_i-1\right)^2\Big) \tag{14}$$

where z is the output of the neural network, y is the label value, $max\left(0, z_i\right)$ is the maximum value of 0, and z_i, $min\ \left(0, z_i-1\right)$ is the minimum value of 0 and (z_i-1). The partial derivative of loss value $L\left(y, z\right)$ for z_i is obtained as follows:

$$\begin{aligned}&\frac{\partial L\left(y, z\right)}{\partial z_i}\\ &= \frac{1}{n}\cdot\frac{\partial\left((1-y_i)\cdot max\left(0, z_i\right)^2+y_i\cdot min\ \left(0, z_i-1\right)^2\right)}{\partial z_i}\\ &= \begin{cases}\dfrac{2y_i(z_i-1)}{n}, & if\ (z_i<0)\\ \dfrac{2(z_i-y_i)}{n}, & if\ (0\le z_i<1)\\ \dfrac{2(1-y_i)z_i}{n}, & if\ (1\le z_i)\end{cases}\end{aligned} \tag{15}$$

Let $\mathrm{n}=1$, when the tag value y is 0 and 1, the relationship between z_i and $L(y_i, z_i)$ and the relationship between z_i and gradient are shown in Figure 4.

The HMSE loss function proposed in the current paper is based on the principle of the mean squared error (MSE) loss function [25]. In the training process, the MSE loss function tends to let z_i converge to 0 and 1. Different from the MSE loss function, in the HMSE loss function, the gradient of z_i is 0, while in other parts, the absolute value of the z_i gradient increases with the increase of distance from y_i, as shown in Figure 7. Therefore, when the HMSE loss function is used, the distance between z_i and y_i can be used to represent the confidence degree of the neural network, which has good performance in inference problem-solving tasks. The results are verified by comparative experiments in Sections 3 and 4.

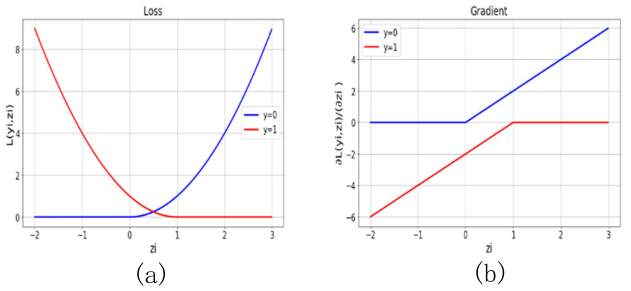

FIGURE 4. When the tag value y is 0 and 1, respectively, the relationship between z_i and $L(y_i, z_i)$ is shown in (a), and the relationship between z_i and the gradient is shown in (b).

III. EXPERIMENTAL RESULTS AND ANALYSIS

A. DATA SET COMPOSITION

The Sudoku inference data set adopted by our team comes from [26]. There are 180,000 questions in the training set, 18,000 questions in the test set, and 18,000 questions in the verification set; in addition, the number of known numbers in the three data sets ranges from 17 to 34. In the training set, each known number corresponds to 10,000 questions. In the test set and verification set, each prompt number corresponds to 1,000 questions. We first set the Sudoku question into a matrix with the shape of 9 × 9 and set the unknown number to 0 so that the number is in the range of 0–9.

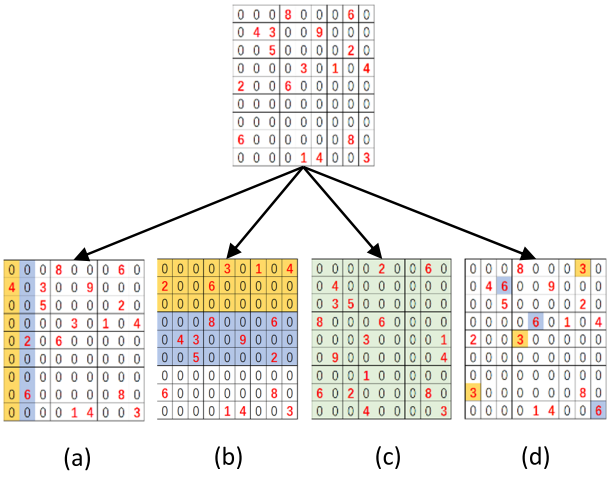

FIGURE 5. The operation diagram of our team for data enhancement of the Sudoku inference data set.

To prevent overfitting of the neural network in training, we designed a data enhancement method according to Sudoku rules. As shown in Figure 5, the data sets are randomly combined according to the transformation mode in the figure, hence greatly increasing the number of training sets.

As shown in Figure 5, where (a) represents the row (or column) exchange in the 9 × 3 (or 3 × 9) block in the same Sudoku inference problem, (b) represents a different 3 × 9 (or 9 × 3) block exchange, (c) represents the transposition of the matrix, and (d) represents the numerical exchange of the known numbers. In the Sudoku matrix mentioned above, because the number form is a character, it is not suitable for being directly applied to neural networks. Therefore, it needs to be transformed into one hot tensor form. Therefore, when inputting a neural network, the Sudoku question is one hot tensor with a shape of 9 × 9 × 10. The number range of the Sudoku answer is 1–9, so we converted it into a tensor of 9 × 9 × 9.

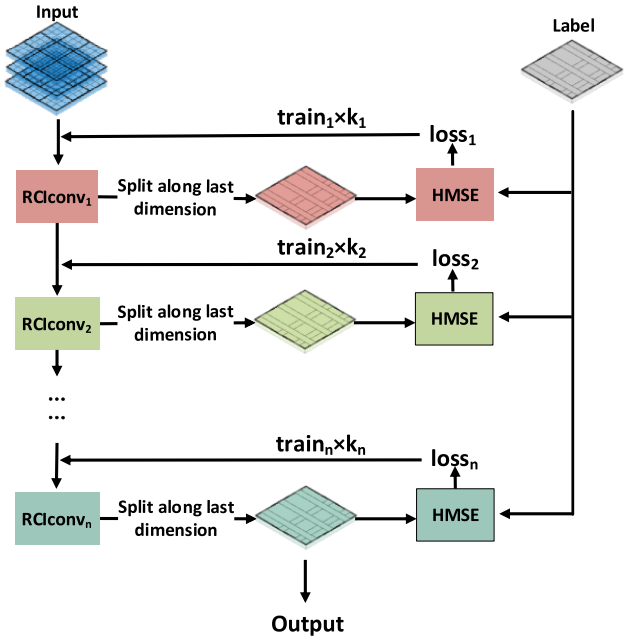

FIGURE 6. The operation and training process of the data in the first loop.

B. MODEL PRETRAINING

In the solution for the calculus logical inference problem, the parameters of each loop's body should be initialized according to the parameters of the previous loop's body. Therefore, our team pretrained the first loop body RCIconv_1 in the RCI-Net model. After the first circulation body when RCIconv_1 is pretrained, there is no need to do the pretraining from the second cycle body, and the training method of the subsequent circulation body is the same, that is, the output segmentation after four cycles, to obtain the predicted value and loss value. The operation and training process of the data in the first loop body are shown in Figure 6.

When training the RCIconv_1 structure, we first segmented the output of a single cycle to get the predicted value and then input this into the HMSE loss function to obtain the loss value $loss_1$. Here, the goal of $train_1$ was to reduce $loss_1$. After completing the pretraining of the first cycle body RCIconv_1, the parameters of the first loop body are used to initialize the second circulation body RCIconv_2. After training the first cycle body RCIconv_1 k_1 times, the parameters of the first loop body are used to initialize the second circulation body RCIconv_2. In the same way, after training the second cycle RCIconv_2 k_2 times, the third cycle to the N-cycle is gradually trained until loss_n tends to be stable. Then, we get the output

TABLE 2. The procedure of the model pretraining algorithm.

Function Train (X, Y, n)

For i from 1 to n do

Net=SDconv$_1$ (X, 256)

Net=Split (Net)

Loss$_1$=HMSE (Net, Y)

train←minimize (Loss$_1$)

End For

For i from 1 to n do

Net=SDconv$_1$ (X, 256)

Net=SDconv$_1$ (Net, 256)

Net=Split (Net)

Loss$_2$=HMSE (Net, Y)

train←minimize (Loss$_2$)

End For

.....

For i from 1 to n do

Net=SDconv$_1$ (X, 256)

Net=SDconv$_1$ (Net, 256)

Net=SDconv$_1$ (Net, 256)

Net=SDconv$_1$ (Net, 256)

Net=Split (Net)

Loss$_n$=HMSE (Net,Y)

train←minimize (Loss$_n$)

End For

End Function

after N cycles, and segment it to get the predicted value and loss value. The pretraining steps of this model are shown in Table 2.

C. TRAINING PROCESS AND RESULTS

In the experiment, our team used the RCI-Net model to test the Sudoku inference data set and compared it with the latest inference model. We used deep conv [27] and RCIconv, model single cycle, and complete cycle for comparative training; we set up two groups of comparative experiments. At the same time, the HMSE loss function and Softmax loss function are compared to prove the effectiveness of this research technology.

We use tensorflow [28] to build a neural network, we use an Adam optimizer [29] to optimize, and we train the model on NVIDIA GTX 1080ti GPU. In the pretraining, the learning rate was 0.0128. In the following training, the learning rate becomes half of the original one every time we enter the next loop. During the training, the batch size was 128.

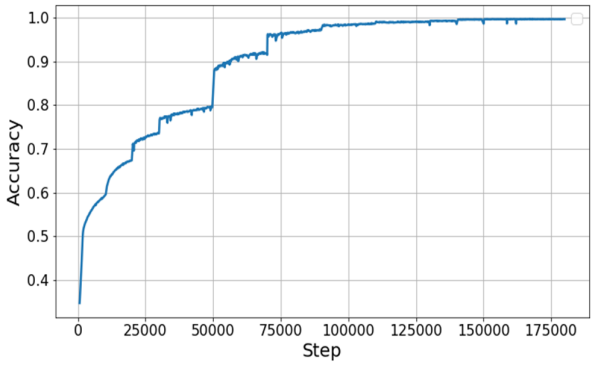

FIGURE 7. In the training process, the filling accuracy curve of the model in the training set.

Figure 7 shows the change in the filling accuracy of the RCI-Net model on the Sudoku inference training set with the number of iterations.

It can be seen from Figure 7 that when the number of iterations of the whole network is 125,000, the accuracy rate of the RCI-Net model tends to be stable, and it has a good ability to solve Sudoku inference problems. Finally, the accuracy of the RCI-Net model in the training set was found to be 99.71%. As can be seen from the figure, the accuracy rate of the accuracy curve at some positions will increase rapidly. This is because when the RCI-Net model is trained a certain number of times, the convolution cycle structure in the model will switch progressively, going from the previous loop to the next loop body. At this time, the filling accuracy value is also based on the output of the next loop.

After the training, we tested the complete training set, test set, and verification set and calculated the Sudoku solution accuracy rate of the RCI-Net model in a single cycle operation. The experiment shows that the accuracy rate of the model for the test set can reach 97.24% when the number of prompts in the inference question is 17. The accuracy rate of a single cycle solution of the RCI-Net model with an increase in the number of prompts is shown in Figure 8.

Next, we further tested the RCI-Net model, using the model for complete recursive confidence loop operation, testing the

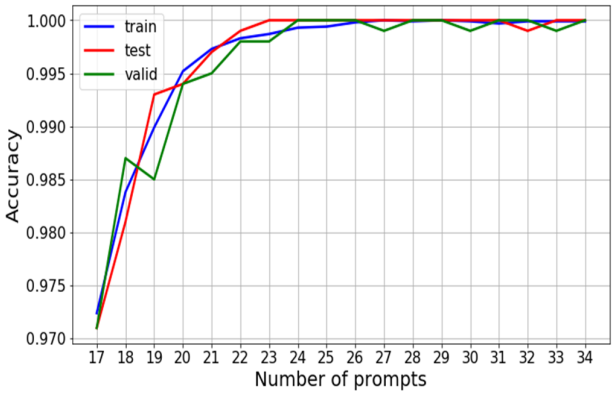

FIGURE 8. Accuracy of Sudoku inference in the RCI-Net model after a single cycle operation.

complete training set, test set, and verification set, and calculating the accuracy rate of the model for the Sudoku solution after a complete cycle. The experimental results show that the accuracy of the model for the test set can reach 99.67% when the number of prompts in the inference question is 17. The accuracy of the RCI-Net model in the complete cycle with an increase in the number of prompts is shown in Figure 9.

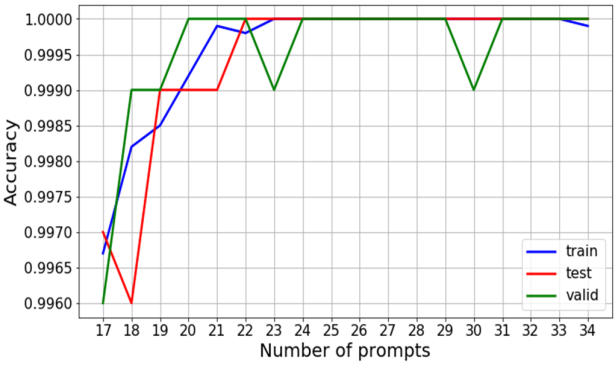

FIGURE 9. Accuracy of Sudoku inference in the RCI model after complete cycle operation.

D. COMPARATIVE EXPERIMENT

In the training process of the RCI-Net model, when the traditional Softmax cross entropy is used as the loss function of the model, the model may collapse because the loss value exceeds the system value range. Therefore, we constructed the loss function of the RCI-Net model by using HMSE, which not only completely prevented training collapse, but also maintained the high accuracy of the model. To verify the effectiveness of the HMSE loss function, we used the Softmax cross entropy loss function and HMSE loss function to conduct comparative training on the RCI-Net model. The training results are shown in Figure 10.

In the contrast experiment, the neural networks used in the two experiments are a single cycle RCI-Net model. In the first experiment, the HMSE loss function is used to train the neural network, and the Softmax cross entropy loss function is used to train the neural network in the second experiment. The two groups of experiments were trained in the same environment and were trained 40,000 times. The accuracy of filling in the training process was compared between the two groups. It can be seen from Figure 9 that when training for 10,000 times, the model using the Softmax cross entropy loss function will collapse the training because the loss value exceeds the system value range, so the training cannot continue. The HMSE loss function proposed in the current paper allows the model to train normally. To clearly show the advantages of the HMSE loss function compared with Softmax cross entropy loss function and better show the reason why the training of the model using Softmax cross entropy loss function collapsed, we have visualized the data distribution of the output layer of two experiments during the training process, as shown in Figure 11.

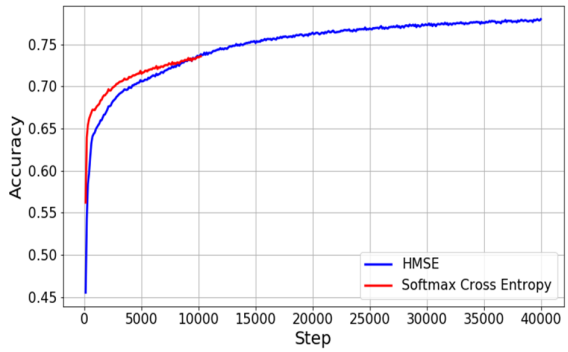

FIGURE 10. The accuracy curve of the RCI-Net model using two different loss functions.

FIGURE 11. The data distribution of the output values of the two groups of neural networks during the training process.

In Figure 11, we show the model (a) using HMSE loss function training and (b) using Softmax cross entropy loss function training. Here, when the model is trained with the HMSE loss function, the data distribution of the model output value can be kept stable and the distribution range small. When the Softmax cross entropy loss function is used for training, the data distribution of the model output value increases with the increase of training times. This proves that the HMSE loss function has the ability to keep the output layer data distribution stable.

TABLE 3. The comparison between the accuracy of the proposed model and the latest models.

Method	Number of prompted numbers	Accuracy
Loopy BP, parallel [3]	17	53.2%
Loopy BP, random [3]	17	61.7%
Loopy BP, modified [4]	17	92.5%
Deep convolutional network [30]	24-36	70%
Recurrent relational network [6]	17	96.6%
Satnet [7]	31-42	98.3%
RCI-Net (single cycle)	17	97.24%
RCI-Net (complete cycle)	17	99.67%

E. COMPARISON OF EXISTING TECHNOLOGIES

In the algorithm verification stage, we use the Sudoku inference test set, use the accuracy rate of the model solution and time-consuming time of the model as indicators, and make a detailed calculation and comparative analysis of the test results of the proposed RCI-Net model and the latest inference solution model. Compared with the latest model, the inference problem-solving model has a better ability to solve inference problems.

In a common inference process, the output rate of a problem is often measured. The samples can be divided into four cases according to the combination of their real categories and the categories predicted by the classifier: true positions (TP), false positions (FP), true negatives (TN), and false negatives (FN). The definition of accuracy is shown in equation (16):

$$acc = \frac{TP + TN}{total} \times 100\% \tag{16}$$

In the process of testing the model, we first must input the Sudoku inference test set into the RCI-Net model to obtain the solution accuracy of the model. At the same time, the same test set is input into other inference models to obtain the solution accuracy of these algorithm models. After the above steps were completed, we were able to compare the accuracy results obtained; the results are shown in Table 3.

Because the data distribution of the data sets used by each research is different, the data sets with more prompts are used in [7], [30] and the data sets with fewer prompts in [3], [4], [6]. To unify the criteria for judging the pros and cons of each scheme, we used the same data set for these models and unified the number of prompts to 17 to better evaluate the advantages and disadvantages of the different models. As shown in Table 3, the RCI-Net model has the highest accuracy among the Sudoku solution models. Even if the RCI-Net model only carries out a single cycle, its solution accuracy is still better than the most recent model. By inputting the same Sudoku inference data, the RCI-Net model can obtain a higher solution accuracy and has strong practicability in solving the logic inference problems of calculus.

F. LIMITATIONS

In the current paper, we have proposed an RCI-Net model based on the SOR method and the recurrent confidence principle to solve the problem of logical inference in calculus. The model is mainly composed of a progressive structure with more hidden layers and higher complexity, making the process of training inference networks more time-consuming. In addition, the accuracy of the model for solving Sudoku inference problems depends on the number of prompts. When the number of prompts is 17, the accuracy rate cannot reach 100%, which is true for the other models as well. The reason is that with a decrease in the number of prompts, the parameters needed to be calculated in the model increase exponentially. Even with sufficient training data, the experimental results are not better. However, this model can avoid training failure caused by overfitting when the number of prompts is small and the number of iterations is sufficient. At the same time, the inference model based on the SOR method and the recurrent confidence principle can provide more types of inference methods for people and help them complete inference tasks better.

IV. CONCLUSION

In the current study, a new deep learning model, RCI-Net, was built based on the linear extrapolation method, here by using the SOR method and recurrent confidence principle. The model adopts the progressive way, and the inference is repeated step by step until the solution is found. The integration of the SOR method makes the model reduce the deviation of each unknown value in the logic inference problem, that is, to reduce the difference between the unknown value and correct solution. Based on this, two successive replacement steps are used for linear extrapolation to improve the accuracy. We have also proposed the RCIconv layer and HMSE loss function and applied them to the RCI-Net model, which not only completely prevented training collapse, but also maintained high accuracy. This superiority was proved through theory and experiment. Compared with the latest inference

model, this model has improved the accuracy of the Sudoku solution.

In the follow-up research, we will further explore how to simplify the neural network structure, reduce the training time, and further improve the accuracy of solving Sudoku inference problems. In the next step, we will study how to combine the RCI-Net model with generative countermeasure neural network [31] or Autoencode neural network [32] to construct an inference problem generation neural network.

REFERENCES

[1] G. Charwat, W. Dvorák, S. A. Gaggl, J. P. Wallner, and S. Woltran, "Methods for solving reasoning problems in abstract argumentation—A survey," *Artif. Intell.*, vol. 220, pp. 28–63, Mar. 2015.
[2] X. Zhang, B. Gong, F. Yang, and S. Ang, "A stochastic multicriteria acceptability analysis–evidential reasoning method for uncertain multiattribute decision-making problems," *Expert Syst.*, vol. 36, no. 4, Aug. 2019, Art. no. e12426.
[3] H. Bauke, "Passing messages to lonely numbers," *Comput. Sci. Eng.*, vol. 10, no. 2, pp. 32–40, Mar./Apr. 2008.
[4] A. Pal, S. Chandra, V. Mongia, B. K. Behera, and P. K. Panigrahi, "Solving sudoku game using a hybrid classical-quantum algorithm," *Europhys. Lett.*, vol. 128, no. 4, Jan. 2020, Art. no. 40007.
[5] G. Singh and K. Deep, "A new membrane algorithm using the rules of particle swarm optimization incorporated within the framework of cell-like P-systems to solve sudoku," *Appl. Soft Comput.*, vol. 45, pp. 27–39, Aug. 2016.
[6] R. B. Palm, U. Paquet, and O. Winther, "Recurrent relational networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. CesaBianchi, and R. Garnett, Eds., 2018, pp. 1–22.
[7] P. W. Wang, P. L. Donti, B. Wilder, and Z. Kolter, "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6545–6554.
[8] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, Ed., 2017, pp. 1–10.
[9] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," presented at the 34th Int. Conf. Mach. Learn., vol. 70, Sydney, NSW, Australia, 2017.
[10] M. Fitzsimmons and H. Kunze, "Combining Hopfield neural networks, with applications to grid-based mathematics puzzles," *Neural Netw.*, vol. 118, pp. 81–89, Oct. 2019.
[11] C. A. Tavares, T. M. R. Santos, N. H. T. Lemes, J. P. C. D. Santos, J. C. Ferreira, and J. P. Braga, "Solving ill-posed problems faster using fractional-order Hopfield neural network," *J. Comput. Appl. Math.*, vol. 381, Jan. 2021, Art. no. 112984.
[12] G. A. F. Guerra and S. B. Furber, "Using stochastic spiking neural networks on SpiNNaker to solve constraint satisfaction problems," *Frontiers Neurosci.*, vol. 11, Dec. 2017, Art. no. 714.
[13] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, Aug. 2009.
[14] S. Sevgen, E. Arslan, and R. Samli, "Solving sudoku puzzle with numbers recognized by using artificial neural networks," *Istanbul Univ., J. Electr. Electron. Eng.*, vol. 17, no. 1, pp. 3205–3211, 2017.
[15] W. Chen, C.-C. Chang, S. Weng, and B. Ou, "Multi-layer mini-sudoku based high-capacity data hiding method," *IEEE Access*, vol. 8, pp. 69256–69267, 2020.
[16] T. Allahviranloo, "Successive over relaxation iterative method for fuzzy system of linear equations," *Appl. Math. Comput.*, vol. 162, no. 1, pp. 189–196, Mar. 2005.
[17] T. Shu, S. Todorovic, and S.-C. Zhu, "CERN: Confidence-energy recurrent network for group activity recognition," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4255–4263.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017.
[19] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2267–2273.
[20] H. Chen, Y. Zhang, M. K. Kalra, F. Lin, Y. Chen, P. Liao, J. Zhou, and G. Wang, "Low-dose CT with a residual encoder-decoder convolutional neural network," *IEEE Trans. Image Process.*, vol. 36, no. 12, pp. 2524–2535, Dec. 2017.
[21] W. Shen, M. Zhou, F. Yang, D. Yu, D. Dong, C. Yang, Y. Zang, and J. Tian, "Multi-crop convolutional neural networks for lung nodule malignancy suspiciousness classification," *Pattern Recognit.*, vol. 61, pp. 663–673, Jan. 2017.
[22] Q. Zhu, Z. He, T. Zhang, and W. Cui, "Improving classification performance of softmax loss function based on scalable batch-normalization," *Appl. Sci.*, vol. 10, no. 8, p. 2950, Apr. 2020.
[23] S. Okada, M. Ohzeki, and S. Taguchi, "Efficient partition of integer optimization problems with one-hot encoding," *Sci. Rep.*, vol. 9, no. 1, Sep. 2019, Art. no. 13036.
[24] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1510–1519.
[25] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.
[26] R. B. Palm, U. Paquet, and O. Winther. (2020). *Recurrent Relational Networks for Complex Relational Reasoning*. Accessed: Jan. 14, 2020. [Online]. Available: https://github.com/rasmusbergpalm/recurrent-relational-networks/blob/master/tasks/sudoku/generate_hard.py
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[28] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*. Accessed: Jan. 14, 2020. [Online]. Available: https://arxiv.org/abs/1412.6980
[30] K. Park. *Can Neural Networks Crack Sudoku?* Accessed: Jan. 14, 2020. [Online]. Available: https://github.com/Kyubyong/Sudoku
[31] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.
[32] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, Sep. 2014.