A 4x4 Sudoku Solving Model Based on Multi-layer Perceptron

Haoyu Du[†, *]
Department of Computing Science
University of Alberta
Edmonton, Canada
du2@ualberta.ca

Lanhe Gao[†, *]
School of Information Science and Technology
Shanghai Tech University
Shanghai, China
gaolh@shanghaitech.edu.cn

Xiaoyue Hu[†, *]
Department of Computer and Information Science
University of Oregon
Eugene, USA
xiaoyueh@uoregon.edu

[†]These authors contributed equally.

***Abstract*— Sudoku is a very popular mathematical and logical curiosity with abundant variations. Many artificial intelligence (AI) programs that imitate human expertise have been developed to solve sudoku puzzles. However, most popular methods, such as Recurrent Relational Networks and Double Q-learning, are primarily reinforcement learning (RL) algorithms. These algorithms are rather complex since they consist of two or more fundamental models for achieving high accuracy. To circumvent this drawback, this paper proposes a simple Multi-Layer Perceptron (MLP) model that not only produces correct results but is easy enough for implementation as well. Since the MLP model is a supervised learning method and is widely used in classification problems, this paper first discusses how the training data is generated. It then demonstrates a coding method that can be used to convert 4×4 sudoku puzzles to classification problems. Afterward, it shows the MLP settings and how the predictions from the model can be transformed into legal solutions. This model has been thoroughly tested on sudoku puzzles with various levels of difficulty, and the resulting data illustrate that its accuracy is no worse than those reinforcement learning methods and other non-AI algorithms. Moreover, with slight modification, the MLP model can be utilized on larger scales, such as 9×9 sudoku puzzles, or even harder ones.**

Keywords- MLP model; Sudoku; Automatic solving; Q-learning

I. INTRODUCTION

Originating from France in late 19th century, sudoku is a pencil-and-paper game of deductive reasoning [1]. In the classic 9×9 sudoku puzzle, the goal is to deduce the numbers to be filled in the blanks according to the numbers already given, so that the resulting sudoku satisfies that each row, each column, and each of the nine 3×3 subgrids contains numbers from 1 to 9 without repetition [2]. During the reasoning process, the player needs to perceive the formation of the puzzle, understand the underlying relationships between blanks and numbers given, and finally make decisions based on the context of probability.

The state-of-the-art RL algorithm for solving sudoku puzzles is the Recurrent Relational Networks. This algorithm emulates the human expertise of tackling complex problems on a step-by-step basis. This architecture performs well in solving sudoku puzzles, which require "a chain of interdependent steps of relational inference" [3]. For all its accuracy, this model is rather convoluted. It could be an overshoot for small-scale problems: a 4×4 sudoku puzzle needs much less effort in multi-step reasoning.

Another prominent work for solving sudoku puzzles is Q-learning. While Q-learning is model-free and easy to implement, recent studies reveal that the efficiency of the classic Q-learning is largely confined by the characteristics of the hardware in modern computers, as the number of states that need to be considered may transcend the capacity of a learning table [4]. Similarly, other methods based on Q-learning, such as Deep Q-Network (DQN), Double Depp Q-Network (DDQN), Temporal Difference Learning (TD), and Proximal Policy Optimization (PPO), also possess the same drawback that the learning process is excessively strenuous, therefore far more time- and hardware-consuming [4] [5] [6] [7].

The complexity of much advanced algorithm may outweigh its diminishing return. Therefore, this paper employs the MLP model for solving sudoku puzzles of small scales, emphasizing that MLP model benefits from better time and space complexity over other RL algorithms or even non-AI methods.

The main aspects of this work can be summarized as follows:

1: The testing and verification of the MLP model being successful in solving sudoku puzzles.

2: The comparison and analysis between the MLP model and other sudoku solving methods.

3: The conjecture of the promising applications of the MLP model in the future.

The rest of this paper is organized as follows. Section II talks about the method that is used in this work. Section III discusses the comparison of the method which is used in this work and other possible methods. Finally, Section IV concludes this paper.

2021 International Conference on Electronic Information Engineering and Computer Science (EIECS) | 978-1-6654-1674-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/EIECS53707.2021.9587969

II. METHODOLOGY

This work contains Data Processing, MLP Model Setting, and Decision Policy for solving 4×4 Sudoku puzzles. We convert and encrypt the 4×4 sudoku puzzle in the data processing step and then set up the MLP model to train it against those data. Afterward, we select the most probable number as the solution for the correspondent blank based on our decision policy, send the puzzle board to the beginning, and training it again and again until we get the final solution.

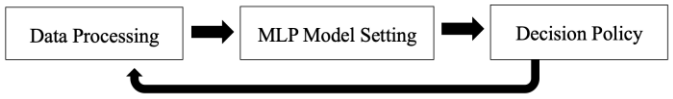

Figure 1 Work process

A. Data Processing

Although MLP is an effective model for classification problems, Sudoku yet cannot be easily considered as a classification problem at first glance. Therefore, a data processing module is needed for any input sudoku puzzles. The data processing procedure designed to solve this problem includes two steps: conversion and encryption.

Conversion is a very simple step, transforming a 4×4 sudoku puzzle to a two-dimensional array with 0 representing blanks that need to be filled. Take the puzzle in Figure 1 as an example. The blanks will be first replaced by 0. Then each row would be extracted and form a one-dimensional array. Finally, all four arrays will be put in one array to form the two-dimensional array eventually. The two-dimensional array for the puzzle in Figure 2 is

$$[[2,0,4,0],[3,4,1,0],[0,0,0,4],[4,3,0,0]] \quad (1)$$

2		4	
3	4	1	
			4
4	3		

Figure 2 Example puzzle

The encryption step is more complicated. In order to convert the Sudoku puzzle to a classification problem, each digit (0, 1, 2, 3, 4) will be considered as a class, or a category. Then, filling the sudoku is equivalent to classifying each blank to one of the classes (1, 2, 3, 4). Obviously, 0 is not an acceptable option because a valid solution should not contain any blanks. Afterwards, since each digit is completely equivalent in weight, one-hot coding is needed to eliminate the influence that numbers' magnitudes might cause in the calculation process. Each class, 0, 1, 2, 3, 4, would be converted to categorical data as followed.

$$\begin{aligned} 0 &\rightarrow [1,0,0,0,0] \\ 1 &\rightarrow [0,1,0,0,0] \\ 2 &\rightarrow [0,0,1,0,0] \\ 3 &\rightarrow [0,0,0,1,0] \\ 4 &\rightarrow [0,0,0,0,1] \end{aligned} \quad (2)$$

After the two steps above, each input would be in the shape of $4 \times 4 \times 5$, ready for the training and prediction process.

B. MLP Model Settings

MLP is a very basic neural network model, with all the inputs, hidden layer neurons and output layer neurons are completely connected [8]. A diagram of the MLP model used in this problem is shown in Figure 3. The input is five-dimensional, corresponding to 0, 1, 2, 3, 4, and the output is four-dimensional, corresponding to 1, 2, 3, 4. The reason for the difference between the input and the output is stated in Section II-A. There are two hidden layers in this model, with 16 neurons in each layer. To prevent overfitting, a dropout of 0.4 is set between the two hidden layers. The activation functions for both hidden layers are ReLU, and the activation function of the output layer is softmax [9] [10].

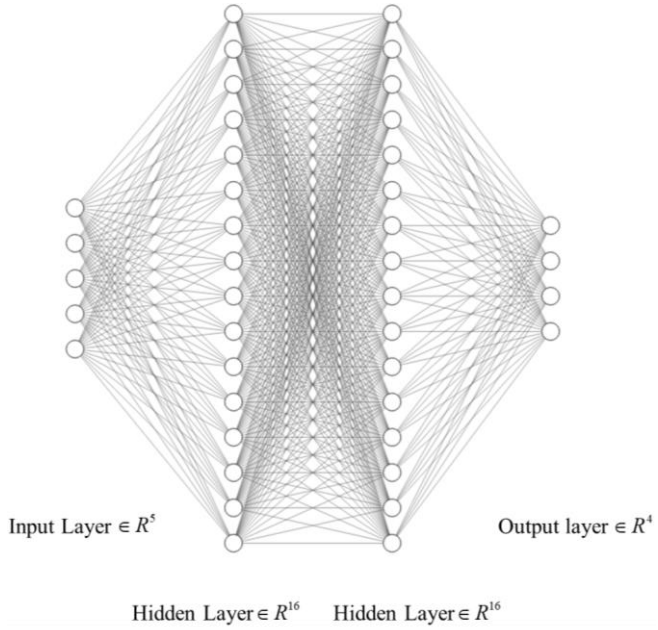

Figure 3 Model diagram

C. MLP Model Training

A large enough training set is required for the MLP model to predict accurate solutions. The procedure of creating such an enormous training set involves creating legal solutions and removing some numbers from solutions to form sudoku puzzles. Firstly, in an empty 4×4 sudoku board, the two 2×2 subgrids on the top left and bottom right are randomly filled with integers 1, 2, 3, and 4, respectively. That no violation of the rules occurs in this step is straightforward and can be easily checked. Secondly, the rest of the 2×2 subgrids are each filled with integers 1, 2, 3, and 4, at random just like in the first step. However, the resulting sudoku board may not

constitute a legal solution – violation of the rules may exist. Thirdly, the sudoku board, therefore, is checked for validity. If the board happens to be a legal solution, then it will be recorded. If the board has any violation of rules, we will jump to and perform step 2 again and continue from there. In this way, more than 250 legal solutions can be created. For each legal solution, one to five numbers are randomly removed, so it turns into a sudoku puzzle. Then the puzzle and the solution come together to form a puzzle-solution pair, which constitutes an instance. In this way, for each solution, 50 puzzles are created for this project. Hence, a training set consisting of 12,500 instances – puzzle-solution pairs – can be established to train the MLP model.

D. Decision Policy

Given a sudoku puzzle as an input, the output of the MLP model is a 16×4 Prediction Table, which is shown in Figure 4.

Integer Index	1	2	3	4
1	0.2	0.4	0.2	0.2
2	0.7	0.1	0.1	0.1
3	0.2	0.1	0.2	0.5
...	...	...	...	...
16	0.1	0.4	0.2	0.3

Figure 4 Prediction table

The *Index* label marks each 1×1 square in the sudoku board from left to right, row by row. The *Integer* label signals the integer to be filled in a given square. And finally, each cell in the Prediction Table represents the probability of a corresponding event at the current board stage. For example, as shown in Figure 4, the cell (2,1) in Prediction Table illustrates that, given the current stage of the board, the probability that integer 1 is the correct solution to square No. 2 – the square in the first row and second column – is 70%. For every stage of the board, after the MLP model has produced the Prediction Table, the cell with the highest value will be selected among all the cells that correspond to a blank in the sudoku board; cells that correspond to a square which has already been filled with an integer in the sudoku board will be safely ignored. Therefore, the cell with the highest value represents the most likely event at the current stage. For example, if cell (2,1) in Prediction Table in Figure 5 has the highest value among the 32 candidate cells, then it is most likely that square No.2 in the current sudoku board should be filled with integer 1. In this way, a decision on the number to be filled in an empty square has been reached. For each stage of the board, only the most probable event will be selected as our decision. In other words, only the square in the sudoku board that has the cell with the highest value in Prediction Table will be filled accordingly. So, if the sudoku board still has blanks after one prediction, the updated board will be fed into the MLP model again as an input. Then such iteration continues until the board has no blanks. For instance, in the example shown in Figure 5, another 7 iterations – prediction and making decision – will be needed in order to work out the final solution.

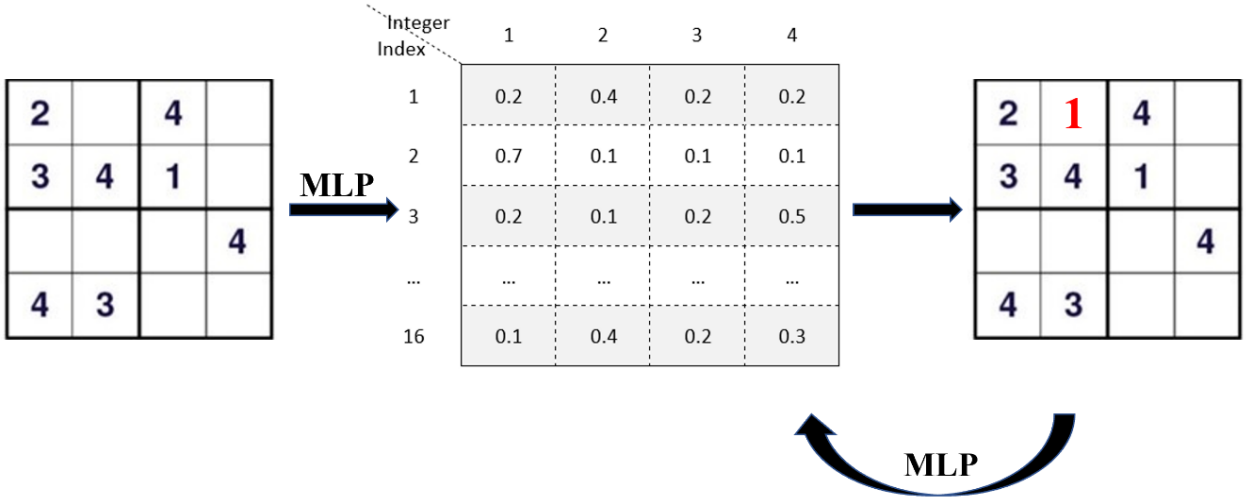

Figure 5 Decision policy flow chart

III. EXPERIMENTS

A. Datasets

- Training Set

The training dataset of the MLP model is self-generated. The generator first produces a legal 4×4 sudoku solution by filling the upper-left square and lower-right square with 1, 2, 3, and 4 without repetition, then filling out other blanks according to the rules. There might be some illegal circumstances, and the generator can detect and discard it, ensuring all the solutions are legal. After finishing producing solutions, the generator removes some digits to create puzzles. For MLP model, the training dataset contains puzzles that have 0 to 6 digits missing, and the quantity of puzzles for each class ranges from 250 to 12500. Eventually, these puzzles would be sent to the MLP model one by one for the training process.

- Testing Set

The testing set of this project consists of a total of 35,000 unique sudoku puzzles that can be equally divided into 7 subgroups, each of which is made up of 5,000 puzzles that all share a special property—a designated amount of blanks. That is, each subgroup represents one type or one level of difficulty of 4×4 sudoku puzzles. The seven subgroups have puzzles with 4 to 10 blanks. Puzzles with a number of blanks less than 4 are considered to be impotent in testing the performance of the MLP model and other sudoku-solving methods. On the other hand, puzzles with more than 10 blanks are of overly casualness and, therefore, may have multiple legal solutions, resulting in similar inefficacy.

- Generation of Experimental Results

The experiments for different sudoku-solving methods are conducted individually. Each method is tested against every type of puzzles in the testing set: the seven subgroups of sudoku puzzles are utilized separately during the testing process so that the performance of methods under different levels of difficulty can be learned. Therefore, for all tested methods, the experimental results regarding each subgroup of—type of—puzzles are recorded with respect to accuracy, duration of the solving process, and duration per puzzle.

B. Evaluation Metrics

The MLP model uses accuracy and running duration as the evaluation metrics. An efficient sudoku-solving method should be of both high accuracy and economical time consumption. In evaluating the performance of different methods, two metrics are employed—accuracy and time per puzzle (Tpp). The accuracy is the ratio of correct tests and total tests; the Tpp metric records the average time consumption of solving a sudoku puzzle of a given type. That is:

$$Accuracy = \frac{correct\_tests}{total\_tests} \tag{3}$$

where $correct\_tests$ indicates the number of the tests that have correct output and $total\_tests$ indicates the number of all training tests.

$$Tpp = \frac{end\ time - start\ time}{Number\ of\ puzzles} \tag{4}$$

In our experiment, we test the MLP model, Q-Learning, and non-AI method separately and collect the data of their accuracy and running duration. Then we compare these data and get the best model for the 4×4 sudoku-solving model.

C. Experimental Results and Analysis

The same test set is run on all three methods on the same computer, and here are the results:

As displayed by Table 1, compared to non-AI method, the MLP model keeps a high accuracy for all puzzles, over 99%, while the accuracy of non-AI method seems good at first, but declines dramatically to 45.92% for hard puzzles. Even though the Tpp of MLP model is higher than that of non-AI, considering the huge advantage on accuracy, MLP performs better than traditional non-AI method.

The Q-learning gives a perfect accuracy result, as shown in Table 1. However, the Tpp of Q-learning is quite high, about 60 times longer than that of MLP model in worse cases. MLP model gives an accuracy on the same level, but with a lot less time consumption. Therefore, MLP model is a more efficient model, without the loss of accuracy.

TABLE I Experiment results

	Non-AI Method		Q-Learning		MLP model	
# of blanks	Accuracy	Tpp (second)	Accuracy	Tpp (second)	Accuracy	Tpp (second)
4	99.62%	0.00008	100.00%	0.02128	100.00%	0.13168
5	98.26%	0.00007	100.00%	1.57421	100.00%	0.16355
6	95.84%	0.00009	100.00%	1.16279	100.00%	0.19852
7	89.42%	0.00009	100.00%	7.16798	100.00%	0.23381
8	80.60%	0.00011	100.00%	9.70421	99.98%	0.26764
9	65.42%	0.00012	100.00%	14.2377	99.92%	0.29992
10	45.92%	0.00016	100.00%	20.3002	99.82%	0.34099

IV. CONCLUSION

Expediently implemented and fairly accessible, the MLP model shows outstanding results in solving 4×4 sudoku puzzles automatically. Through the tricky way of data processing, the MLP model receives high-quality inputs conducive to training the model and making fair predictions. And then with iterative prediction, the MLP model can achieve rather high accuracy in finding solutions for sudoku puzzles. By comparing and analyzing the experimental results, it can be found that the simple MLP model is capable of solving 4×4 sudoku puzzles faster and easier, and that meanwhile its accuracy is equivalent to, if not better than, those of RL algorithms and non-AI methods.

In the future, a series of meaningful studies can be conducted subsequently. For example, the MLP model would help to solve some difficult math problems automatically. It can generalize underlying patterns from closely related problems and then solve each problem individually according to the patterns it discovered. Moreover, the MLP model could be implemented to develop automated game playing, which includes Go, maze, and all kinds of combinatorial games.

REFERENCES

[1] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. New York, NY, USA: John Wiley & Sons, 1985.

[2] R. Lewis, "Metaheuristics can solve sudoku puzzles," Journal of Heuristics, no. 13, pp. 387–401, 2007.

[3] R. B. Palm, U. Paquet, and O. Winther, "Recurrent Relational Networks," arXiv:1711.08028v4 [cs.AI], 2018.

[4] K. Poloziuk and V. Yaremenko, "Neural networks and Monte-Carlo method usage in multi-agent systems for sudoku problem solving," Technology Audit and Production Reserves, no. 6 (2 (56)), pp. 38-41, 2020, doi: 10.15587/2706-5448.2020.218427.

[5] Y. Wang and F. Wu, "Multi-agent deep reinforcement learning with adaptive policies," arXiv: 1912.00949 [cs.LG], 2019.

[6] H. V. Hasselt, A. Guez, and David Silver, "Deep reinforcement learning with Double Q-learning," arXiv:1509.06461 [cs.LG], 2015.

[7] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: a review of challenges, solutions, and applications," IEEE Transactions on Cybernetics, vol. 50, no. 9, pp. 3826-3839, 2020, doi: 10.1109/TCYB.2020.2977374.

[8] B. M. Wilamowski, "Neural network architectures and learning algorithms," IEEE Industrial Electronics Magazine, vol. 3, no. 4, pp. 56-63, 2009, doi: 10.1109/MIE.2009.934790.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on ImageNet Classification," arXiv:1502.01852 [cs.CV], 2015.

[10] I. Goodfellow, Y. Bengio, and A. Courville, "6.2.2.3 Softmax units for multinoulli output distributions," in Deep Learning. Cambridge, MA, USA: MIT Press, 2016, pp. 180-184.