Sudoku generation and rating algorithm

Qiming Huang and Feilong Tang

April 16, 2021

1 Introduction

Sudoku is a popular game for people of all ages. It originated in Japan and became very popular in the United States. There are many ways to solve array, but it is not the focus of this paper. Generating and rating a Sudoku is much more interesting than solving the Sudoku puzzle. In this paper, we discuss an algorithm to generate new Sudoku puzzles and score at four difficulty levels. we generate a solved Sudoku by backtracking algorithm, and then randomly remove the cells to generate Sudoku. After that we use backtracking algorithm to detect whether this Sudoku is the unique solution. Then, we try to analyze the difficulty of Sudoku by using one of the methods of explaining neural network, Taylor decomposition [1], and calculate the complexity score, and divide the difficulty into four levels. Auto encoder for classification is used to prove our method is workable. The arrangement of this paper is showed below

Section two is the literature review about previous study on sudoku. Section three is the introduction about our model, generating method and difficulty analysis are included. Then, section four is complexity analysis. Finally, section five is conclusion. (All codes in this paper can be accessed at: `https://github.com/Thomaszz4/Project/tree/main/sudoku_challenge`

2 Related works

In the last ten years, many researchers have used different algorithms to solve and generating Sudoku problems. For example, Sitorus [2] used backtracking algorithms to solve Sudoku problems. It tries filling digits one by one. Whenever it finds that the current digit cannot lead to a solution, we remove it (backtrack) and try the next digit. But the complexity of the backtracking algorithms method is relatively high. Mantere et al. [3] used genetic algorithms (GA) to solve and generate Sudoku puzzles. It is randomly chosen various rows from two different parents, in order to create the child by crossover. And then tried a mutation to solve and generate new Sudoku puzzles. Pelánek [4] used a constraints propagation approach to evaluate the difficulty of Sudoku puzzles, i.e., the use of "logic techniques". It simulates humans to use the logic techniques (Hidden single, Naked Single, Direct Pointing, i.e.) solving this problem. It is more objective to use logical skills of different levels of difficulty to define the difficulty of Sudoku.

3 Method

3.1 Genetating

The method of generating the sudoku is backtracking algorithm and random removing.Firstly, Generate a (9×9 grid) sudoku with answers. The three constraint conditions: 1. The digit $(1-9)$ can only appear once in each row, each column and its (3×3)square. 1.generate all zero the 9×9 grid sudoku . And solve all the cell through the backtracking algorithm which is a depth-first search. It can explore one of the branches to find a possible solution. For instance, firstly, the algorithm selecting one digit $(1-9)$ to place the cell which is zero, and checks (row, column, and square constraints) if it is allowed to be there. Secondly, If there are no violations then the algorithm advances the next zero cell and selecting one digit $(1-9)$ to place and check. If nine numbers $(1-9)$ are not allowed in a cell, the algorithm leaves the cell blank and moves it back to the previous cell. When it returns to the previous step, it reselects the number and is not the same as before. finally, repeat until you find the allowed value in the last (81st) cell.

Remove some digits randomly from the generated $9*9$ array. Effective removal can improve the possibility of generating Sudoku, that is, generating Sudoku with unique solution. Generally, the more times you remove it, the higher the difficulty level of Sudoku.

Finally, the detail of our model is showed at figure 1 we use the backtracking method to judge whether the removed Sudoku is the only solution. If two or more answers are obtained by the

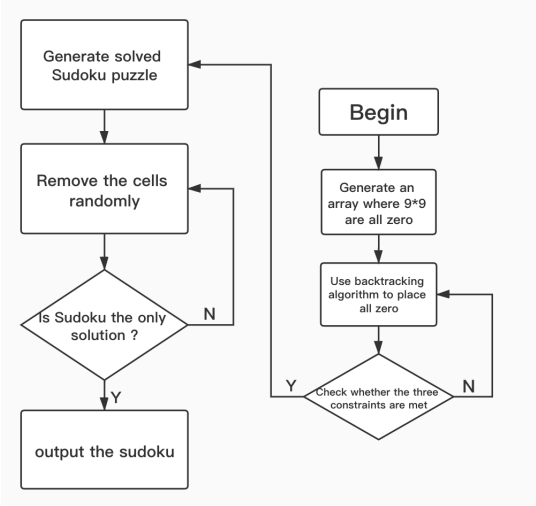


Figure 1: The steps of Sudoku generating

backtracking method, it will be removed again until the unique solution is obtained. One of the ways to judge the difficulty of Sudoku is to determine its difficulty by its backtracking times. It is easy to calculate the backtracking times.

3.2 difficulty analysis

Sudoku is an iteration problem which computer can find the unique solution once search for all its possible numbers for each sub-grid. However, brute force is never the first choice for people to play a sudoku puzzle. In this section, we propose a method to analysis the difficulty level of a given sudoku puzzle. Also a proof of availability of models is given.

Main steps for difficulty level analysis are showed below:

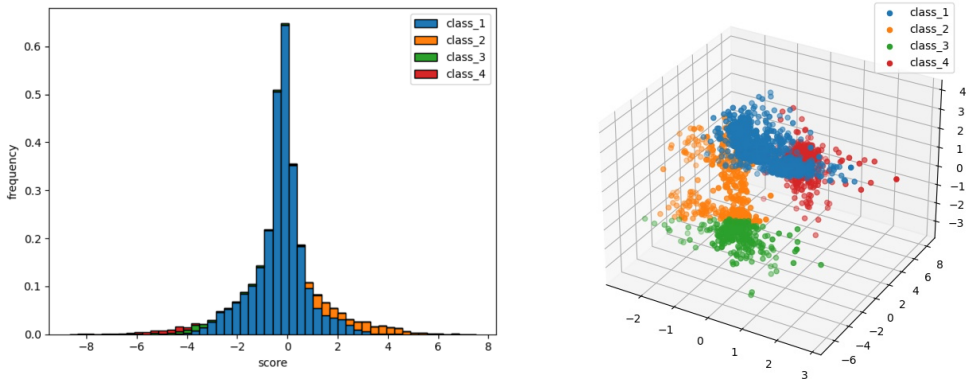
1. tain a CNN model to solve sudoku puzzle based on the data set collected from `https://www.kaggle.com/bryanpark/sudoku`
2. Calculate the gradients of input data and multiple of input data (based on simple Taylor decomposition) and feature map values separately.
3. Classify sudoku with k-means (k=4) based on output of last convolutional layers and 3-dimensional data compressed by auto encoder. (Result shows that auto encoder based classier surpass that of CNN)
4. Divide sudokus into 4 difficulty levels based on the distribution of difficulty scores.

3.2.1 Train slover

we treat the sudoku as a 9x9 size image. As for the unassigned grid, we put 0 in it. The model structure is simple, three convolutional layers and one fully connected layer which is used to classification. Specially, in this task, we consider sudoku as a classification task. Given an unsolved sudoku, we want our model to classify the number from 1 to 9 for each grid.In addition, the solution of sudoku is unique integers instead of a probability which is usually used in classification task. Hence, we convert our labels (solved sudoku puzzle) into one hot coding which means the original labels is 81x1 (reshape 9x9 into one dimension), after one hot coding, the labels become 81x9. In this case, for each row we can output the probability and 9 columns represent the numbers from 1 to 9. After training with 200000 samples in 20 epochs, the accuracy can reach 97.6%. Then we want to perform unsupervised leaning algorithm, k-means with k = 4 for instance, on output feature map from CNN to classify which is done by [5]. Their research show that applying unsupervised learning algorithm on feature map from CNN

surpasses that of directly applying algorithm on input data (in this case 9x9 original sudoku data). However, after experiments, the result is not desirable. Showed on figure 2a. We will discussed this later. Hence, we try a new method to classify which is auto encoder [6] .

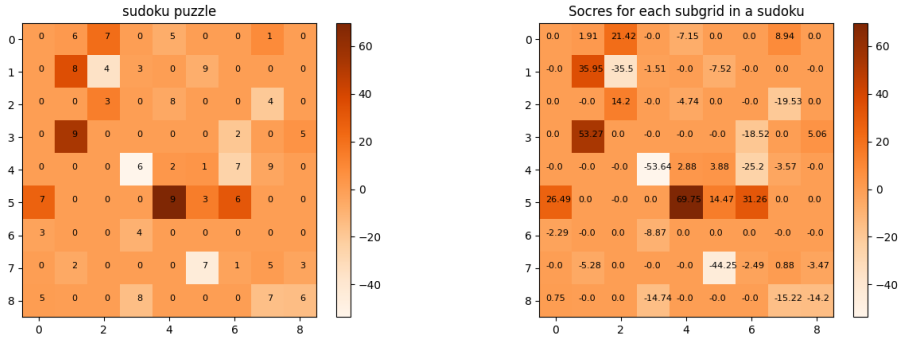
Auto encoder firstly subsamples the input image into a low dimensional data with a sub-model called encoder and then upsamples this low dimensional data back to original data with a sub-model called decoder (training data is same as the target data). During training, network is forced to used least data to store the most valuable information to represent input data. That is what we want. Hence, we train an auto encoder model which the low dimensional data with 3 default. Then, we only save encoder to compress our sudoku data into 3-dimensional representations. After that, we again perform k-means with k = 4 on those 3-dimensional data. The results show that the classification effect is better than that of applying on feature of CNN. Showed on fogure 2b .



(a) The classification of directly applying k-means on feature map outputed from CNN

(b) The classification of applying k-means on 3-dimensional feature outputed from encoder

Figure 2: Comparison of the effect of two kinds of classification



(a) The unsolved sudoku puzzle

(b) significant score for each subgrid of puzzle

Figure 3: Apply taylor decomposition on the model of sovling sidoku puzzles

3.2.2 Simple Taylor Decomposition

Convolution neural network can automatically learn image features through convolution kernel, which has excellent performance in the field of image classification and target detection. Although we can not understand what features convolutional neural network learned in the process of training, we can use some tools to visualize the weights learned by neural network, For example, Taylor decomposition [1], Layer-wise relevance propagation [7] and Grad-cam[8].We

Table 1: result of PCA

Dimension	First	Second	Third
PCA	0.69148695	0.21081918	0.09769384

can regard Sudoku as a small 9x9 image, and we can also let it automatically learn a potential feature relationship of Sudoku through convolution neural network. In this paper, we choose taylor decomposition to explain our model and calculate the impact of each small area on the final result as the difficulty of sudoku

Taylor expansion can construct a polynomial to approximate a function with the derivative value of each order at a certain point. Given a function $f(x)$, choosing a root point $\tilde{x}$ which satisfies $f(\tilde{x}) = 0$ and the first-order taylor decomposition of $f(x)$ is[1]:

$$f(x) = f(\tilde{x}) + \left(\frac{\partial f}{\partial x}|_{x=\tilde{x}}\right)^{\top} \cdot (x - \tilde{x}) + \varepsilon = 0 + \sum_{p} \frac{\partial f}{\partial x_p}|_{x=\tilde{x}} \cdot (x_p - \tilde{x_p}) + \varepsilon$$

the $\sum_p$ cover all the pixels in a image, in terms of sudoku, each subgrid in 9×9 grid. Then, simplify this equation, we got:

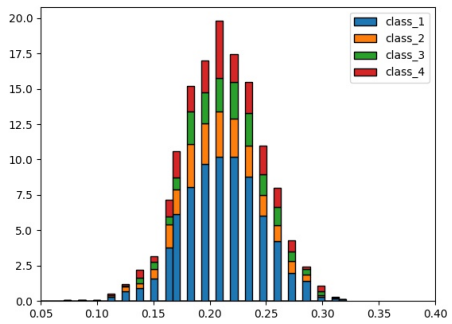
$$R(x) = \frac{\partial f}{\partial x}|_{x=\tilde{x}} \odot (x - \tilde{x})$$

for each $R(x)$ represents the contribution to solve the puzzle. Showed on figure 3b The dark red pixel represents the positive contribution to the final prediction result, and the light colored pixel represents the negative contribution to the final prediction result.This show us what sub-grids in the unsolved sudoku puzzle 3a are important for solving.

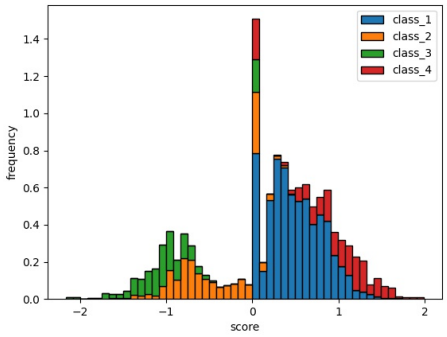
3.2.3 Computation of difficulty level

Then we add up all the scores to form the final Sudoku difficulty score and its distribution is showed on 6. However, when we apply the classes (given by k-means) on these score, the distribution is showed on 4a. It indicates that, for each class, the data distribution is not separable which means for each score, there may be different classes. This can not be used for distinguishing different difficult levels.To solve this issue, we multiplied 3-dimensional features instead input X to make data distribution a clearer distinction. In this case, PCA (Principal Component Analysis) is applied to reduce dimension for the purpose of multiplication. The variance ratio given by PCA is showed on table 1.Hence, we choose first element to perform multiplication and get the final scores. The scores distribution is showed on figure 4b. It successfully separate different classes although there some overlap which proves our method is able to work. In addition, we also try to add low dimensional data with gradients, the result is showed on figure 4c. It looks better than multiplication. For multiplication, when the score is 0, it contains all classes which is not expected. Maybe it's because the gradients contain many zeros after multiplication, the data is mapped to zero. Also in the addition, this issue does not happen which proves that. As for the difficulty level, we assume the score closer to 0, the simpler it is, and vice versa. Compared with solving steps (or time) showed on figure 5, they are similar which shows that the simple sudoku distribution is more, and the difficult sudoku distribution is less. In this case, according to the figure, we can divided sudoku into four difficult levels, class 1 and class 2 are simple, class 3 and class 4 are hard. Moreover, we also extent classes in k-meas, showed on figure 4d. But the overlap of different classes is high which means the performence is not great.

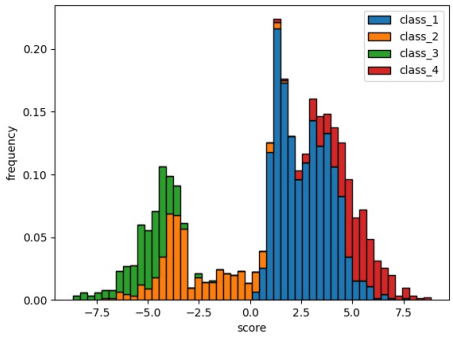
When your gradient is large which means you still left many effect to do to reach the final goal (minimize the loss function). Consider during a correct training processing, at the vary begining training epoch, your gradients are alway large and loss is reducing rapidly cause weights are



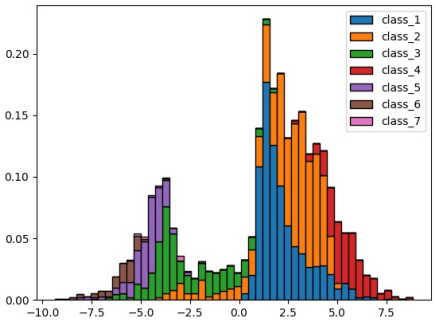
(a) The distribution from CNN feature map



(b) The distribution from auto encoder (multiplication)



(c) The distribution from auto encoder (addition)



(d) The distribution from auto encoder (extent for 7 classes)

Figure 4: Two kinds of score distribution with classes information

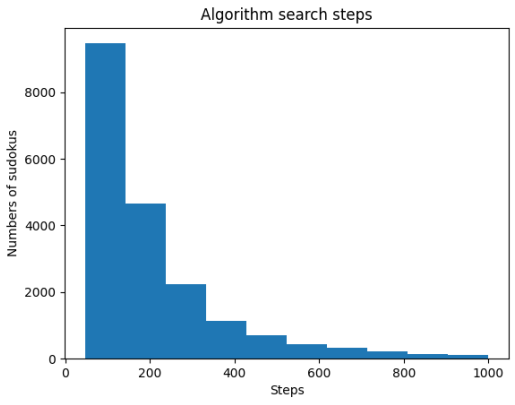


Figure 5: The steps needed to solve arrays using backtracking

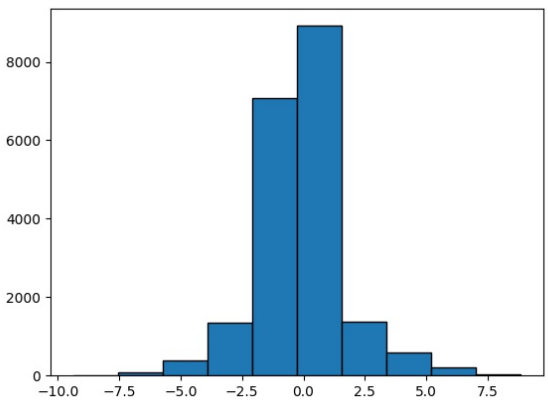


Figure 6: The distribution of solving sudoku puzzles

update via back propagation algorithm. But after some epochs, the loss will reduce slightly. In this case, the gradients are small. In addition, in the local minimal, the gradient of that function is zero. Hence, we assume, the score of sudoku (multiplication of gradient and feature representations) we computed with a small distance to zero is a simple one and vice versa. For explanation, feature representations are used to add information of classes to make different types of data can be better distinguished.

4 Complexity analysis of Generating method

Listing 1: Code of complexity analysis

```
void dfs(int step) {
    if (step == 81) {
        solution++;
    }else{
        int x = step/9;
        int y = step%9;
        if(show_sudoku[x][y]== 0){
            for(int i = 0;i<9;i++){
                // Check whether three constraints are satisfied
                if(check(x,y,i)){
                    show_sudoku[x][y] = i;
                    dfs(step+1);
                    show_sudoku[x][y] = 0;
                }
            }
        }else{
            dfs(step+1);
        }
    }
}
```

In the process of recursion, each time a number is placed in a legal position, and then enter the next empty grid to place the number. And so on, until N spaces are all placed, output a solution. Hence, N is the number of spaces to be filled in Sudoku.According to Hamilton period, the time complexity is given by the following formula:

$$T(N) = N * (T(N-1) + O(1))T(N) = N * (N-1) * (N-2).. = O(N!)$$

When backtracking is used to solve and generate Sudoku, Its time complexity is O(N!).Space complexity is O(N).

5 Conclusion

In this paper,we discover a mothod using backtracking and randomly remove subgrid to generate new sudoku puzzle. Then, we applied taylor decompotion to divide sudoku puzzles into four levels of difficulty.And verify that using classified data distribution to check if difficulty scores are easily distinguished. After that, we divided sudoku into four difficulty levels based on distribution of solving steps utilizing back tracking to solve sudoku. For future study, more experiments can be focused on extent the classes of sudoku with a low overlap range.

References

[1] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[2] P. Sitorus, E. Zamzami, *et al.*, "An implementation of backtracking algorithm for solving a sudoku-puzzle based on android," in *Journal of Physics: Conference Series*, vol. 1566, p. 012038, IOP Publishing, 2020.

[3] T. Mantere and J. Koljonen, "Solving, rating and generating sudoku puzzles with ga," in *2007 IEEE congress on evolutionary computation*, pp. 1382–1389, IEEE, 2007.

[4] R. Pelánek, "Difficulty rating of sudoku puzzles: An overview and evaluation," *arXiv preprint arXiv:1403.7373*, 2014.

[5] J. Guérin, O. Gibaru, S. Thiery, and E. Nyiri, "Cnn features are also great at unsupervised classification," *arXiv preprint arXiv:1707.01700*, 2017.

[6] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[7] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: an overview," *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 193–209, 2019.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.