The Solution of The Sudoku Games: Exploration of The Potential of Cnn in Logical Reasoning Problems

Jiaye Li

School of Physics, Nanjing University, Nanjing, Jiangsu, China

Abstract. Logical reasoning problems, such as Sudoku, have long been considered a challenge for traditional artificial intelligence techniques. While these problems require a high level of pattern recognition and deductive reasoning, the application of deep learning, particularly Convolutional Neural Networks (CNNs), in solving such tasks has not been extensively explored. Recent advancements in neural networks have shown promise in handling complex reasoning tasks, yet their potential in logical problem-solving domains remains largely untapped. However, exploring the potential of CNN in logical reasoning problems, such as Sudoku, remains an under-explored area. The research uses a dataset of one million Sudoku games from Kaggle. Data is preprocessed and fed into a CNN model. The model is evaluated using loss and accuracy metrics, trained with a batch size of 64 for up to 100 epochs, and optimized with Adam optimizer. The model achieves a test accuracy of 0.9632, indicating some generalization ability. This study is significant as one of the early efforts to apply CNN to logical reasoning. It provides a basis for future research, highlighting both the potential of CNN in this area and the challenges that need to be addressed for further improvement.

1 Introduction

Sudoku is a captivating logic-based number-placement puzzle that can be represented as a 9 by 9 matrix. The aim of Sudoku is to fill the entire 9 by 9 grid with numbers 1 through 9. It requires that all the digits from 1 to 9 must appear precisely one time in every row, every column, and each of the 3 by 3 sub-grids. There are two existing methods of Sudoku solving: the back-tracking method and the unique candidate method.

The back-tracking method is a classic algorithm for solving Sudoku puzzles [1]. The algorithm starts by filling an empty cell with a number from 1 to 9. If the number is valid according to the Sudoku rules, the algorithm moves on to the next empty cell and repeats the process. However, if a number leads to an invalid state, the algorithm will retrace its steps to the preceding cell and proceed to attempt the subsequent available number. This process persists ceaselessly until a viable solution is successfully identified or every single possible combination of numbers has been thoroughly explored and exhausted. While the back-

231840047@smail.nju.edu.cn

tracking method is guaranteed to find a solution if one exists, it can be computationally expensive, especially for difficult puzzles with few initial clues.

The unique candidate method is a more straightforward logical deduction technique[2]. It relies on identifying cells that have only one possible number based on the numbers already present in the row, column and sub-grid. For example, if in a particular row, all numbers from 1 to 9 are present except for 5, and the column and sub-grid of an empty cell in that row also do not have 5 as a conflicting number, then the value of that cell must be 5. This method is relatively fast and can quickly solve easy to medium-difficulty Sudoku puzzles. However, for more complex puzzles, it may not be sufficient on its own, and additional techniques may need to be combined.

Convolutional Neural Network (CNN), as a type of deep neural network, has revolutionized the field of artificial intelligence, particularly in image processing and computer vision [3]. Some scholars have conducted excellent research on the applications of CNN in image processing [4] and computer vision [5]. However, there are few articles exploring the potential of CNN in dealing with logical reasoning problems [6]. Sudoku games have clear rules, quantifiable data, and adjustable levels. These characteristics make them an excellent starting point. Therefore, this research aims to explore the potential of CNN in handling logical reasoning problems by studying its performance in solving Sudoku puzzles. It is hoped that this research can provide some new inspiration to other scholars.

In this article, data is obtained and processed from the dataset first. Subsequently, the model is constructed, and parameters are set and adjusted according to the performance during the solving process. Then, the loss and accuracy curves are plotted to analyze the performance of CNN in solving Sudoku puzzles.

2 Dataset and Method

2.1 Dataset Description and Data Preprocessing

The dataset used in this experiment consists of one million Sudoku games and is sourced from Kaggle.

Data transformation begins with converting CSV data into numpy arrays for efficient numerical operations and TensorFlow compatibility [7]. Quiz data is reshaped into a 4D tensor (-1, 9, 9, 1), where the first dimension represents the number of samples, the second and third represent the grid of the Sudoku, and the fourth is for the single channel. Solution data is reshaped to a 3D tensor (-1, 9, 9) and its values are subtracted by 1 to start from 0, better suiting the sparse categorical cross-entropy loss function in model training.

2.2 Model Architecture

The CNN model architecture for solving Sudoku puzzles, as detailed in Table 1, starts off with a series of convolutional layers that are specifically engineered to extract features in a hierarchical manner from the input grid. The initial convolutional layer (conv2d) employs 128 filters with a 3 by 3 kernel size, utilizing ReLU activation and the same padding to maintain the 9 by 9 spatial dimensions of the Sudoku grid (output shape: None, 9, 9, 128). This architectural choice preserves the structural integrity of the puzzle while introducing non-linearity through ReLU activation, enabling the network to capture complex patterns.

As illustrated in Table 1, the network progressively increases its filter capacity through successive convolutional layers (conv2d_1 to conv2d_6), expanding from 128 to 256, 512, and ultimately 1024 filters. This hierarchical expansion facilitates the extraction of increasingly abstract features, transitioning from local pattern recognition to global

relationship understanding within the Sudoku grid. Each convolutional operation is immediately followed by batch normalization (batch_normalization to batch_normalization_6), a critical component that stabilizes the learning process by normalizing layer inputs. The batch normalization layers, with their respective parameter counts (512 to 4096 parameters), effectively reduce internal covariate shifts, accelerating convergence and enhancing generalization capabilities.

The final convolutional layer (conv2d_7) strategically reduces the feature space to 9 filters with a 1 by 1 kernel, producing an output shape of (None, 9, 9, 9) with 9,225 parameters. This transformation prepares the feature maps for subsequent processing through fully-connected layers. The network then employs a flatten operation to convert the multi-dimensional output into a one-dimensional vector of 729 elements.

As shown in Table 1, the model incorporates two dense layers (dense and dense_1) with 512 and 729 neurons respectively, containing 373,760 and 373,977 parameters. These fully-connected layers perform the final feature transformation and mapping to the output space. The architecture concludes with layer normalization (1,458 parameters), reshaping to a 9 by 9 by 9 tensor, and softmax activation, which generates probability distributions for each cell across the 9 possible digits. This comprehensive architecture, with its carefully designed progression of layers and normalization techniques, effectively captures the complex constraints and relationships inherent in Sudoku puzzles.

Table 1 Sequential Model Architecture

Layer (type)	Parameter	Output Shape
conv2d (Conv2D)	1280	(None, 9, 9, 128)
batch_normalization (Batch Normalization)	512	(None, 9, 9, 128)
conv2d_1 (Conv2D)	147584	(None, 9, 9, 128)
batch_normalization_1 (Batch Normalization)	512	(None, 9, 9, 128)
conv2d_2 (Conv2D)	295168	(None, 9, 9, 256)
batch_normalization_2 (Batch Normalization)	1024	(None, 9, 9, 256)
conv2d_3 (Conv2D)	590080	(None, 9, 9, 256)
batch_normalization_3 (Batch Normalization)	1024	(None, 9, 9, 256)
conv2d_4 (Conv2D)	1180160	(None, 9, 9, 512)
batch_normalization_4 (Batch Normalization)	2048	(None, 9, 9, 512)
conv2d_5 (Conv2D)	2359808	(None, 9, 9, 512)
batch_normalization_5 (Batch Normalization)	2048	(None, 9, 9, 512)
conv2d_6 (Conv2D)	4719616	(None, 9, 9, 1024)
batch_normalization_6 (Batch Normalization)	4096	(None, 9, 9, 1024)
conv2d_7 (Conv2D)	9225	(None, 9, 9, 9)
flatten (Flatten)	0	(None, 729)
dense (Dense)	373760	(None, 512)
dense_1 (Dense)	373977	(None, 729)
layer_normalization (Layer Normalization)	1458	(None, 729)
reshape (Reshape)	0	(None, 9, 9, 9)
activation (Activation)	0	(None, 9, 9, 9)

3 Experiments and Results

3.1 Settings

The experimental setup for model training and evaluation is systematically configured as detailed in Table 2. The dataset is divided into subsets by making use of the "train_test_split" function sourced from the "sklearn" library, with 80% allocated for training and 20% reserved for evaluation. This strategic division ensures robust model generalization by maintaining a substantial portion of unseen data for performance assessment.

As specified in Table 2, the model is compiled with the “sparse_categorical_crossentropy” loss function, which is particularly suitable for integer-encoded target labels. The optimization process employs the Adam optimizer with a learning rate of 0.001, providing efficient gradient-based optimization with adaptive moment estimation. Model performance is quantitatively assessed using accuracy as the primary evaluation metric, measuring the proportion of correctly predicted cells in the Sudoku grid.

The training configuration, as outlined in Table 2, implements a batch size of 64 and a maximum of 100 epochs to balance computational efficiency and model convergence. To mitigate overfitting, an EarlyStopping callback is implemented with specific parameters: it monitors the validation loss (val_loss) and terminates training if no improvement is observed for 3 consecutive epochs. This configuration ensures optimal model performance while preventing unnecessary computational expenditure. The comprehensive settings, as systematically presented in Table 2, establish a robust framework for model training and evaluation, facilitating effective learning while maintaining generalization capabilities.

Table 2 Model Training and Evaluation Settings

Item	Content	Setting / Parameter
Data Splitting	train_test_split	80% for training 20% for evaluating
Loss Function	sparse_categorical_crossentropy	
Optimizer	the Adam optimizer	A learning rate of 0.001
Evaluation Metric	accuracy	
Model Training	batch_size & epochs	The batch size is 64 the maximum epoch is 100.
Early Stopping Callback	EarlyStopping	Monitored Metric: val_loss Patience: 3 consecutive epochs

3.2 Analysis

3.2.1 Loss Curves

The training loss starts at 0.9590 in the first epoch and drops rapidly to 0.0442 by the 13th epoch. The steep decline in the training loss evidences efficient model learning from the training data. Gradient-based parameter updates iteratively align model predictions with ground-truth targets, systematically reducing loss through error minimization.

The validation loss begins at 0.2737 and also decreases initially, reaching 0.1119 by the 13th epoch. However, after the 7th epoch, the rate of decrease in the validation loss slows down, and there is a slight increase after the 11th epoch. This behavior is a strong indication of overfitting.

3.2.2 Accuracy Curves

The training accuracy climbs steadily from 0.6471 in the first epoch to 0.9841 by the 13th epoch. This illustrates that the model is becoming more accurate in predicting the solutions for the training data. Learning from the training data, the model is able to make better predictions, resulting in higher accuracy.

The validation accuracy increases from 0.8998 in the first epoch to 0.9632 by the 13th epoch. However, similar to the validation loss, the growth in validation accuracy slows down around the 9th epoch. This also suggests overfitting, as the model's performance on the validation data is not improving as rapidly as on the training data.

3.2.3 Evaluation of the Test Set

After training, the evaluation results show a loss of 0.1119 and an accuracy of 0.9632. The relatively high accuracy indicates that the model is able to generalize reasonably well to new, unseen Sudoku puzzles. However, the non-perfect accuracy and the signs of overfitting suggest that there is still room for improvement.

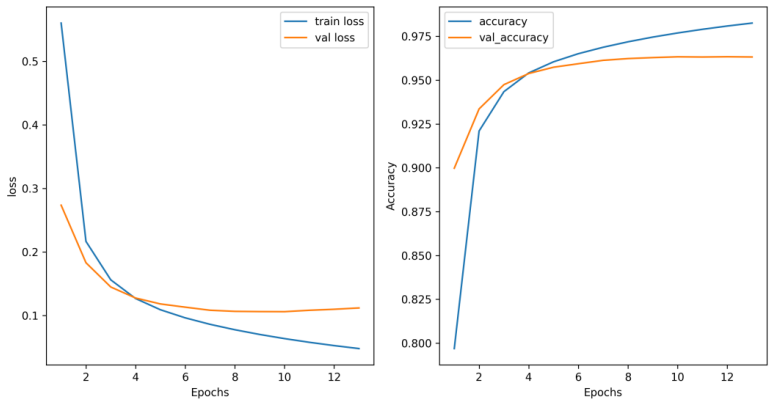

Figure 1 Evolution of loss and accuracy on training and validation sets (Picture credit: Original).

4. Discussion

4.1 Limitations

4.1.1 Overfitting

One of the most significant limitations of the current model is overfitting. As observed from the training and validation curves, the model's performance on the validation data begins to deteriorate, even though it continues to show improvement on the training data after a specific number of epochs. Overfitting reduces the model's generalization ability, meaning that it may not perform well on new Sudoku puzzles that are different from the ones in the training set. This can be a major problem, especially in real-world applications where the model needs to handle a wide variety of puzzles.

4.1.2 Dataset Limitations

The dataset used for training may not be comprehensive enough to cover all possible Sudoku patterns. Sudoku puzzles can have an extremely large number of possible initial states and solution patterns. The absence of certain complex or rare patterns in the training data can lead to the model's inability to accurately predict solutions for such puzzles. Additionally, the data preprocessing step of subtracting 1 from the solution values, while necessary for the loss function, may have introduced some biases or challenges that affect the model's performance.

4.1.3 Model Architecture Constraints

The current CNN architecture may not be the most optimal for Sudoku solving [8]. The architectural decisions regarding the depth of convolutional layers, the number of filters per layer, and the configuration of fully-connected layers all exert significant impacts on model performance [9]. For example, the consecutive use of convolutional layers with a fixed filter size of 3 by 3 may not be sufficient to capture all the necessary local and global features of the Sudoku grid. The fully-connected layers at the end of the network may also introduce overfitting due to a large number of parameters, especially if not properly regularized.

4.2 Future Works

4.2.1 Using Deep Q-Network (DQN)

DQN is a type of deep reinforcement learning algorithm that can be used to improve the accuracy of the Sudoku-solving model. In the context of Sudoku, the environment can be the Sudoku grid, and the actions can be the choices of numbers to fill in the empty cells. By training a DQN model, it may be possible to learn a more intelligent strategy for solving Sudoku puzzles, which could lead to a more accurate model.

4.2.2 Generating New Sudoku Games with Deep Reinforcement Learning

Deep reinforcement learning algorithms can also be used to generate new Sudoku games. By training an agent to generate valid Sudoku puzzles, it is possible to create a more diverse and comprehensive dataset for training the solving model. This can help in reducing the

overfitting problem and improving the model's generalization ability. The agent can be rewarded based on the difficulty level and the uniqueness of the generated puzzles.

4.2.3 Classifying Sudoku Games by Difficulty

Another potential area of future work is to classify different Sudoku games based on their difficulty levels. This can be achieved by training a separate model to predict the difficulty of a given Sudoku puzzle [10]. The initial state of the puzzle can serve as the input to the model, with the output being a difficulty rating. This classification can be useful for users who want to choose puzzles based on their skill level and for researchers who want to study the characteristics of different difficulty levels.

5 Conclusion

This research aimed to explore the potential of CNN in handling logical reasoning problems, using Sudoku games as a case study. The CNN model was trained on a dataset from Kaggle. Through data preprocessing, model construction, and training with specific settings, the model achieved a test accuracy of 0.9632, indicating its ability to generalize to new Sudoku puzzles to a certain extent.

However, several issues were identified. The model showed signs of overfitting, as seen from the divergence between loss and accuracy curves on training and validation sets. This overfitting could be due to the model architecture not being optimal for Sudoku, with fixed filter sizes in convolutional layers and large parameter numbers in fully-connected layers. Also, the dataset might not cover all possible Sudoku patterns, and the data preprocessing method might have introduced biases.

The significance of this study lies in being one of the early attempts to apply CNN to logical reasoning problems. It provides a foundation for future research in this area. Additionally, classifying Sudoku games by difficulty can be explored. Overall, this research not only reveals the potential of CNN in logical reasoning but also points out the challenges and opportunities for further improvement in this novel application domain.

References

1. Li, X.: 'Research and Implementation of Solving Algorithms for Sudoku Problems', Comput. Telecommun., 2017, (09), pp. 77–79, DOI: 10.15966/j.cnki.dnydx.2017.09.025
2. Qu, H., Yue, J., Wang, F.: 'Research on Generation and Solution Algorithms of Sudoku Problems', Bull. Sci. Technol., 2017, (06), pp. 14–17, DOI: 10.13774/j.cnki.kjtb.2017.06.004
3. Li, Z., Liu, F., Yang, W., Peng, S.: 'A survey of convolutional neural networks: analysis, applications, and prospects', IEEE Trans. Neural Netw. Learn. Syst., 2021, 33, (12), pp. 6999–7019
4. Liu, H., Sun, X., Li, Y., Han, L., Zhang, Y.: 'A Review of Deep Learning Models for Image Classification Based on Convolutional Neural Networks', Comput. Eng. Appl., pp. 1–29
5. Zhang, S., Gong, Y., Wang, J.: 'The Development of Deep Convolutional Neural Networks and Their Applications in Computer Vision', Acta Electron. Sin., 2019, (03), pp. 453–482

6. Gu, J., Wang, Z., Kuen, J., et al.: 'Recent advances in convolutional neural networks', Pattern Recognit., 2018, 77, pp. 354–377
7. Vamsi, K. S., Gangadharabhotla, S., Sai, V. S. H.: 'A deep learning approach to solve sudoku puzzle', Proc. Int. Conf. Intelligent Computing and Control Systems (ICICCS), May 2021, pp. 1175–1179
8. Narayanaswamy, A., Ma, Y. P., Shrivastava, P.: 'Image Detection and Digit Recognition to solve Sudoku as a Constraint Satisfaction Problem', arXiv preprint, arXiv:1905.10701, 2019
9. Mulamba, M., Mandi, J., Mahmutoğulları, A. İ., Guns, T.: 'Perception-based constraint solving for sudoku images', Constraints, 2024, 29, (1), pp. 112–151
10. Wei, X.: 'Difficulty Level Classification of Sudoku Puzzles Based on Convolutional Neural Network', Acad. J. Comput. Inf. Sci., 2023, 6, (11), pp. 35–39