Sudoku Solver with TensorFlow

Abstract

Sudoku puzzles have long been a popular pastime, challenging players to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub grids contain all of the digits from 1 to 9. In this project, we explore the application of deep learning techniques to solve Sudoku puzzles. Specifically, we train a convolutional neural network (CNN) model to predict the missing digits in incomplete Sudoku puzzles. The model learns to recognize patterns in the provided Sudoku puzzles and generates solutions for them. We demonstrate the effectiveness of the proposed approach by solving Sudoku puzzles with high accuracy and efficiency.

Introduction

Sudoku is a logic-based combinatorial number-placement puzzle that has gained widespread popularity worldwide. The objective of the puzzle is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub grids contain all of the digits from 1 to 9. Although Sudoku puzzles can be solved manually using logical deduction and elimination, solving them algorithmically presents an interesting challenge in the field of artificial intelligence and machine learning.

In this project, we propose a novel approach to solving Sudoku puzzles using deep learning techniques. We leverage the power of convolutional neural networks (CNNs) to train a model that can predict the missing digits in incomplete Sudoku puzzles. By learning from a large dataset of Sudoku puzzles and their solutions, the model learns to recognize patterns and relationships within the puzzles and generate accurate solutions.

Problem Statement

The problem at hand entails the development of a deep learning-based solution for automated Sudoku puzzle solving. Given partially completed Sudoku grids, the objective is to predict the missing digits accurately and efficiently, ensuring that the completed grids adhere to the

Sudoku rules of having each digit from 1 to 9 appearing exactly once in each row, column, and 3x3 subgrid. This project aims to leverage convolutional neural networks (CNNs) to learn patterns and relationships within Sudoku puzzles, enabling the model to generate accurate solutions, thereby offering a novel approach to solving Sudoku puzzles algorithmically.

Dataset Description

The dataset consists of Sudoku puzzles along with their solutions. Each sample in the dataset includes a puzzle grid with some cells filled in with digits (representing the initial state of the puzzle) and the corresponding solution grid with all cells filled in (representing the completed puzzle). Both the puzzles and solutions are represented as 81-digit strings, where each digit corresponds to a cell in the 9x9 grid, with '0' representing an empty cell. The dataset is structured as a CSV file with two columns: 'quizzes' containing the puzzle strings and 'solutions' containing the solution strings. This dataset serves as the basis for training and evaluating the deep learning model for Sudoku puzzle solving.

Methodology:

1. Import Libraries:

Import the necessary libraries including NumPy, pandas, Keras, and Matplotlib for data handling, deep learning model building, and visualization.

2. Load Data:

Load the Sudoku dataset from a CSV file, which contains puzzles and their corresponding solutions.

3. Data Preprocessing:

Preprocess the loaded data by converting the puzzle and solution strings into arrays, reshaping them into 9x9 grids, and normalizing the values to be within the range [-0.5, 0.5].

4. Define Data Generator Class:

Define a custom data generator class DataGenerator to generate batches of data for training the neural network model. This class inherits from the Keras Sequence class and implements the __getitem__ and __len__ methods required for data generation.

5. Define CNN Model:

Define a convolutional neural network (CNN) model using Keras Sequential API. The model consists of convolutional layers, batch normalization, flattening layer, dense layer, reshape layer, and softmax activation function.

6. Compile Model:

Compile the CNN model using the sparse categorical cross-entropy loss function and Adam optimizer with a specified learning rate. The accuracy metric is also specified for model evaluation during training.

7. Split Data:

Split the dataset into training and validation sets, typically using a 95-5 split.

8. Define Callbacks:

Define callbacks including ModelCheckpoint and ReduceLROnPlateau for saving the best model weights based on validation accuracy and adjusting the learning rate during training, respectively.

9. Train Model:

Train the CNN model using the fit_generator method, passing the training and validation data generators, number of epochs, verbose mode, and callbacks list.

10. Evaluate Model:

Load the best model weights saved during training and evaluate the model's performance using a sample Sudoku puzzle. The solve_sudoku_with_nn function is used to solve the puzzle using the trained model, and the print_sudoku_grid function is used to print the solved puzzle grid.

Results

Input ---

```
game = '''
        0 5 2 0 0 6 0 0 0
        1 6 0 9 0 0 0 0 4
        0 4 9 8 0 3 6 2 0
        4 0 0 0 0 0 8 0 0
        0 8 3 2 0 1 5 9 0
        0 0 1 0 0 0 0 0 2
        0 9 7 3 0 5 2 4 0
        2 0 0 0 0 9 0 5 6
        0 0 0 1 0 0 9 7 0
    '''
```

Output ---

```
Sudoku Solution (NN):
3 5 2 | 4 7 6 | 1 8 9
1 6 8 | 9 5 2 | 7 3 4
7 4 9 | 8 1 3 | 6 2 5
---------------------
4 2 5 | 6 9 7 | 8 1 3
6 8 3 | 2 4 1 | 5 9 7
9 7 1 | 5 3 8 | 4 6 2
---------------------
8 9 7 | 3 6 5 | 2 4 1
2 1 4 | 7 8 9 | 3 5 6
5 3 6 | 1 2 4 | 9 7 8
```

Conclusion

In conclusion, this project successfully demonstrated the application of deep learning techniques for solving Sudoku puzzles. By leveraging convolutional neural networks (CNNs), we trained a model capable of predicting missing digits in incomplete Sudoku puzzles, ultimately completing them with high accuracy and efficiency. Through extensive experimentation and training, we achieved a model capable of accurately solving Sudoku puzzles from a variety of initial states.

The developed CNN model showcased robustness and generalization ability, effectively handling different puzzle configurations and providing accurate solutions. Furthermore, the model's performance was evaluated using both training and validation datasets, ensuring its reliability and effectiveness.Overall, this project contributes to the advancement of AI-powered puzzle-solving systems and underscores the potential of deep learning in tackling complex logical challenges.