SUDOKU SOLVER USING CONVOLUTIONAL NEURAL NETWORKS AND BACKTRACKING

Pradyumna Vemuri (pvemuri),
Sriram Sudharsan (ssudhar),
Sudharsan Janardhanan (sjanard)

Abstract

Sudoku is a widely popular game, where the players are expected to have a high logical and pattern detection skills. In this project we explore CNNs and back-tracking models to efficiently solve these Sudoku puzzles. CNN is a fundamental Deep learning model that has been established as a go-to solution for classification tasks in the field of Computer Vision, and the backtracking algorithm solves the puzzle in ideal time, performing much better than any naive method. Using contour detection we isolate Sudoku grids from images of newspaper's pages for testing the model.

1 Introduction

The rise of the Artificial Intelligence (AI) field is due to the desire to build machines that can replicate human processes like logical reasoning, pattern detection, perception and problem solving, with the goal to create human-like machines.

- In 1996, IBM's Deep Blue program defeated the world chess champion Kasparov.
- In 2011, IBM's Watson won Jeopardy.
- In 2017, AlphaZero was build to master Chess, Go and Shogi, which it did.

Motivated by these projects, we decided to tackle the most commonly found, yet challenging puzzles of Sudoku.

The field of Computer Vision has been making strides in the modern world for the last few decades, because of its features and applications. Keras is a popular library used by millions to build Deep Learning Applications. We've also utilised other tools such as OpenCV for purposes of pre-processing.

Although the problem solving aspect of the project can be solved using a brute-force algorithm, it is very time consuming. We have therefore employed an algorithm that utilises backtracking, that provides a solution for this NP-hard problem.

To solve the puzzle, our model places a number in each empty cell without causing any violation in row, column or square. Since, the puzzle depends on the values and positions of the filled in elements, it is important that these dependencies are represented properly. We implement a CNN that has been trained on the MNIST data set to detect the numbers in a grid, then use back-tracking to solve the puzzle and provide a solution matrix as an output.

2 Background

2.1 Sudoku

Sudoku puzzles have numbers ranging from 1-9 placed all over a 9x9 square grid strategically, additionally the grid itself is further divided into subgrids of size 3x3. The goal is to fill the empty cells with respect to the already filled in numbers, without violating the rules. The rule is that the rows, columns and squares of the entire grid can be filled with numbers ranging from 1-9 without repeating any number. Or, with respect to each row, column and subgrid no duplicate numbers are allowed and no number should be left out from 1-9.

2.2 CNN

A CNN or a Convolutional Neural Network is a Deep learning algorithm which is quite often used for image recognition and classification. A CNN is capable of differentiating and learning different aspects of images easily and is capable of performing with high accuracy without much feature engineering. CNNs perform well on image-based tasks as they capture the Spatial dependencies of images using filters and can fit the data well due to reduction in unnecessary parameters and re usability of weights. A CNN starts with a convolutional layer, followed by filters and pooling layers to flatten the features of the classifier.

2.3 Backtracking

Backtracking is an algorithm building technique which works by solving sub-problems recursively one-by-one, until it arrives at the final optimal solution. Backtracking tends to remove partial solutions that reach a dead end, and reverts back to the recent point of decision to pursue another path for a solution. Backtracking generally is used for problems that have a set of clear and well-defined constraints for a solution, like chess or Sudoku. Sudoku has constraints where in each square, row or column each number should not repeated. Back tracking can be used to recursively traverse the puzzle grid and use hash maps to keep track of constraints.

3 Method

The pipeline we used for our project consists of 3 modules:

- **Training a CNN** with MNIST data set for digit recognition.
- **Pre-processing** the images in our Sudoku data set.
- **Backtracking** module for solving the puzzle matrix.

3.1 Training a CNN

Keras

There are many frameworks that are capable of building a Convolutional Neural Network such as Theano, PyTorch and Tensorflow. Out of these Theano is an older framework, while the others are constantly being researched about making them much powerful. Tensorflow is most sought after framework, but still is not specific and has many high and low level machine learning functionalities. Tensorflow is not very user friendly, so we decided to use Keras on Tensorflow to make our CNN's architecture more modular and easier to use.

MNIST Dataset

MNIST dataset or Modified National Institute of Standards and Technology dataset consists of 60,000 gray scale square images of size 28x28, where the images are divided equally between digits 0 to 9.

CNN training

First, the MNIST dataset is loaded and split into training and testing data sets. The model then reshapes images in both testing and training dataset, to convert them to data arrays of single color

channel. The pixel values of all images are converted from integer to float and normalized to the range [0,1].

The CNN after training acts as a digit recogniser due to two main aspects of its architecture: a convolutional later that with its activation function acts as a feature filter max pooling layers that flattens and simplifies the image data to provide features to classify.

We build a Sequential model using Keras, with multiple Convolutional layers and ReLu as the activation function. Since the output is a class label, the model needs to have a layer of 10 nodes to represent digits from 0-9. For this we use a softmax activation function with a Dense layer before the output. Once we compile and fit the model with data, the model is trained for 10 epochs while measuring loss and accuracy at each step.

3.2 Preprocessing the images

Since we have a digit recognition model, we can go ahead and start with the pre-processing of the images from the Sudoku data set we collected.

Dataset

There is no public dataset available for Sudoku puzzles, so we have gathered and made a dataset of our own with around 100 images of Sudoku puzzles acquired from various sources, through the internet. Our model uses backtracking to solve the puzzle part of the Sudoku grid. We use our secondary dataset with real life image data on sudoku puzzles found from various sources such as newspapers.

Data processing and Contour detection

We use OpenCV and its tools to perform suitable pre-processing tasks. We first convert the images from RGB color scale to gray scale, in order to conveniently detect lines of the puzzle grid from in the image. Gaussian blur is used to reduce noise and reduce unnecessary detailing at the pixel level of the image. Subsequently, we perform adaptive thresholding in order to separate the background from the foreground, which helps us in detecting digits effectively. Finally, we identify each and every square in the puzzle's grid. We then iterate through each grid detected from the earlier steps and we evaluate whether it is empty or not, based on the pixel density in that grid. Finally, for each non-empty grid we use the model trained earlier to classify the digit.

This process generates a 9x9 array, with digits from 1 through 9 and '0', where 0 is a placeholder for an empty cell. Backtracking is employed to solve the Sudoku puzzle with this array as an input.

3.3 Backtracking

An easier solution to deploy would be using brute force, where at each cell every digit from 1-9 is place and checked if there is a violation, But this solution in computationally too complex, hence we use backtracking.

Approach

We iterate through the empty grids one by one, and we apply backtracking to each grid. We start by placing the least number, say '1'. If the number that was placed doesn't violate the constraints of the Sudoku game, i.e the same number doesn't repeat in both the row and the column, then the number is placed at that spot. This process continues until we reach a grid where substituting any number leads to an invalid grid. When this happens, we backtrack to the previous solution and try the next possible solution.

The time complexity of this algorithm is $O(9^m)$ and its space complexity O(m), where m is the amount of empty grids.

4 Experimental Results

After training the CNN model on the MNIST dataset on a training set of 60000 images and a test set of 10000 images, the results obtained are as follows.

```
Epoch 1/10
1875/1875 [==============================] - 176s 93ms/step - loss: 0.1835 - accuracy: 0.9443
Epoch 2/10
1875/1875 [==============================] - 177s 94ms/step - loss: 0.0774 - accuracy: 0.9767
Epoch 3/10
1875/1875 [==============================] - 175s 93ms/step - loss: 0.0581 - accuracy: 0.9823
Epoch 4/10
1875/1875 [==============================] - 173s 92ms/step - loss: 0.0477 - accuracy: 0.9856
Epoch 5/10
1875/1875 [==============================] - 174s 93ms/step - loss: 0.0404 - accuracy: 0.9871
Epoch 6/10
1875/1875 [==============================] - 172s 92ms/step - loss: 0.0351 - accuracy: 0.9888
Epoch 7/10
1875/1875 [==============================] - 172s 92ms/step - loss: 0.0300 - accuracy: 0.9910
Epoch 8/10
1875/1875 [==============================] - 173s 92ms/step - loss: 0.0265 - accuracy: 0.9912
Epoch 9/10
1875/1875 [==============================] - 172s 92ms/step - loss: 0.0251 - accuracy: 0.9922
Epoch 10/10
1875/1875 [==============================] - 174s 93ms/step - loss: 0.0240 - accuracy: 0.9926
```

Figure 1: Loss and accuracy of CNN

We record an accuracy of 0.992 on the train dataset and 0.99 on the test dataset. We then run pre-processing on an image we obtain from our secondary dataset and use the CNN model on each grid. The grid generated is as follows.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2: Sample Sudoku Puzzle

After pre-processing is performed, we receive a result as follows on Figure 3.

We then perform backtracking to get the final solution as shown on Figure 4.

Our future plans for this project is to create a front-end so that the user can upload an Sudoku image and our application returns a solved Sudoku array. We also intend to implement a model with using a Regional Based Convolutional Neural Network (R-CNN) and compare the results between the two. R-CNN's perform selective search on an image and extract high quality regions within the image. The extracted features are then resized (in this case only our Sudoku Grid is selected). They perform selective search on the image by generating sub segments within the image and using greedy algorithms to recursively combine similar regions into larger ones. Using R-CNN's however makes the use of our pre-processing module obsolete since we train our neural network to extract features within the image. Our motive is to explore other object detection algorithms such as Faster R-CNN's

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

Figure 3: Matrix generated from CNN

```
5  3  4  | 6  7  8  | 9  1  2
6  7  2  | 1  9  5  | 3  4  8
1  9  8  | 3  4  2  | 5  6  7
- - - - - - - - - - - - - - -
8  5  9  | 7  6  1  | 4  2  3
4  2  6  | 8  5  3  | 7  9  1
7  1  3  | 9  2  4  | 8  5  6
- - - - - - - - - - - - - - -
9  6  1  | 5  3  7  | 2  8  4
2  8  7  | 4  1  9  | 6  3  5
3  4  5  | 2  8  6  | 1  7  9
```

Figure 4: Sudoku puzzle solved with backtracking

and YOLO(You Only Look Once) and perform a comparative study between them all.

5 Conclusion

The CNN we have trained with the MNIST dataset has an 99.2 % accuracy, and has been showing promising results for digit recognition. We further plan to utilise more image detection approaches such as Regional-CNNs and YOLO and perform a thorough comparative study between them as well as create a front end for uploading Sudoku images.

6 References

[1] Wicht, Baptiste & Hennebert, Jean. (2015). Camera-based Sudoku recognition with deep belief network. 6th International Conference on Soft Computing and Pattern Recognition, SoCPaR 2014. 83-88. 10.1109/SOCPAR.2014.7007986.

[2] K. S. Vamsi, S. Gangadharabhotla and V. S. Harsha Sai, "A Deep Learning approach to solve sudoku puzzle," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 1175-1179, doi: 10.1109/ICICCS51141.2021.9432326.