CSE 676 - Deep Learning
Final Project Report

Solving Sudoku using Image Detection, Digit Recognition, and Deep Learning

Prajvala Rajendra Sonawane
Department of Computer Science
prajvala@buffalo.edu
50418778

Nikita Satyajit Desai
Department of Computer Science
ndesai2@buffalo.edu
50425060

Mrunmayee Vijay Rane
Department of Computer Science
mrane@buffalo.edu
50417094

Abstract

Solving the *sudoku* that is printed in the daily newspaper is something that we have all grown up with. However, the problem with this is that we cannot confirm the correctness of the solution until the next day since the solution is printed in the following day's newspaper. From the reference links provided to us by the professor, we have chosen the topic of *Solving Sudoku using Deep Learning*. Multiple machine learning algorithms already exist that solve the *sudoku*, but development is still in progress in the field of taking *sudoku* image as input. For a layman, entering the *sudoku* in array format would be a tedious task as compared to clicking a picture of the *sudoku* and uploading it. We thus propose a technique where in only the *sudoku* image will be required as the input and the algorithm will output the solved sudoku.

1. GitHub Link for project code

Solving-Sudoku-DL (https://github.com/desainikita/Solving-Sudoku-Using-DL)
The instructions to execute the code are explained in the README.md file in the GitHub repository.

2. Introduction

The *SUDOKU* is a 9x9 grid logical puzzle game. Each of the 9x9 grid is further divided into 9 cells thus making a total of 81 cells in the grid. Few of these cells are initially populated as clues that are used to solve the puzzle by filling the left-over ones. The puzzle is considered solved when each 3x3 block, each vertical line of cells and each horizontal line of cells contains exactly one of the digits from 1 to 9 as shown below in the highlighted cells.

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

Our strategy to solve the *sudoku* includes taking the *sudoku* image as input and returning the solved *sudoku* in the output. For this we have trained our network on the digits part of Chars74K dataset and then used this model to read and predict digits from the input image of *sudoku*. This will then be fed as input to the second deep learning model to produce the solution using another trained model.

3. Background

Previous work

We went through some existing work that has been done in this area and came across the following approaches that have certain loopholes that can be worked upon.

Approach 1

A traditional approach [10] that includes solving the sudoku using recursive backtracking approach.

A. First check if the current number is not already present in the current row, column or 3x3 sub-grid. Once confirmed, assign the number.
B. If the number is not present in the respective row, column, or sub-grid, it assigns the number, and recursively checks if the current assignment leads to a solution. If the assignment does not result into a solution, it tries the next number for the current empty cell. If none of the numbers lead to a solution, then it returns false.
C. The loop continues until it finds the correct solution or until it reaches a blocking stage from where no solution can be deduced.

This neural network approach uses Convolutional Neural Network model for digit recognition from the images but solves sudoku using a recursive backtracking problem. The drawback of this approach is that it uses brute force for solving the sudoku instead of using artificial intelligence which is much faster and smarter than traditional methods.

Approach 2

This paper [9] that we studied talks about solving the sudoku using reinforcement learning. However, they could produce an accuracy of only 7% using this approach as the Sudoku puzzle consists of a single correct solution and rewarding the model based on it will require a lot of training and still produce incorrect output during evaluation.

Approach 3

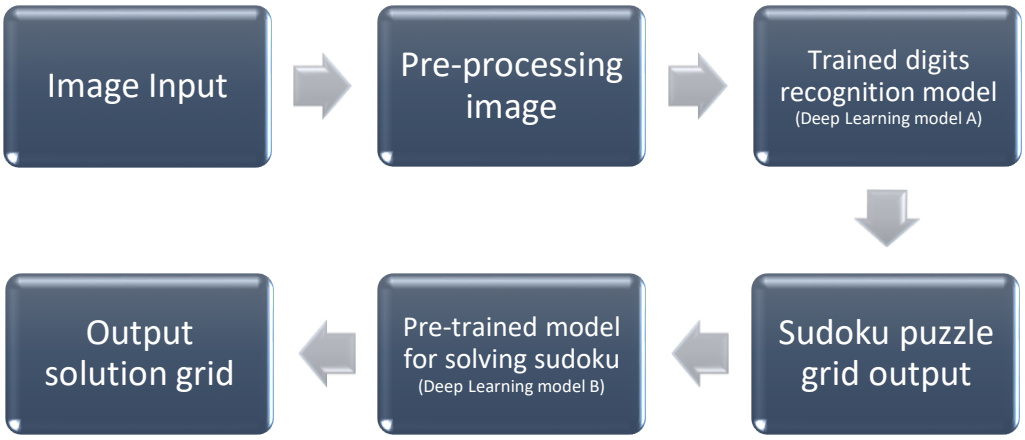
In this approach [3] the sudoku is solved using a Convolutional Neural Network. However, the input that this method takes is in the form of a string. The user must manually enter the entire sudoku in the form of a single string which is not user friendly.

Novelty of the approach

From the approaches discussed above, we see that there is scope for improvement in solving the 9x9 sudoku using deep learning and image processing. Instead of taking string input, we are taking the image of the sudoku as input. We are then performing digit extraction from this image using a deep learning model and then solving the sudoku using another deep learning model. We are processing images of formats *png, jpg, jpeg,* etc.

4. Architecture

We have used two deep learning models that have been trained and are being used in two different phases of our project. The main architecture of implementation follows the below depicted flow.



A. Image Input

This is the image of the sudoku that will be clicked by any smart handheld device and fed as input to the model.

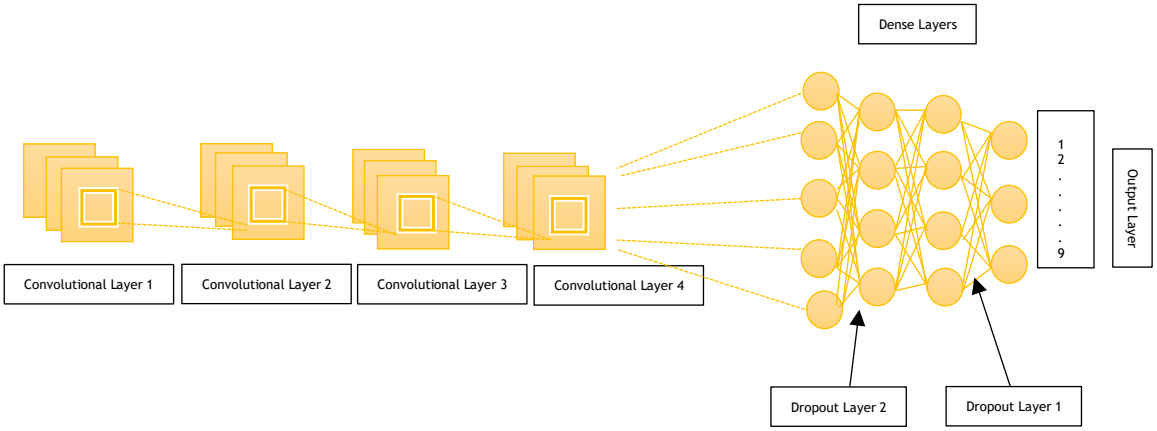
B. Pre-processing image

Pre-processing input image - Preprocessing input image includes application of gaussian filter and adaptive thresholding for removal of noise. This is then followed by finding contours with the largest area for the 9x9 sudoku grid. Post this, the image is straightened using a homography matrix for obtaining it's wrap perspective which removes distortions that result from optical aberrations on some of the images.

Pre-processing cropped image - For obtaining each cell in the 9X9 grid we split the rows and columns of the pre-processed image followed by cropping each cell in each row and column. The resulting cell image is then saved in a folder named *cropnum*. Each of these images in *cropnum* is then cropped to avoid grid lines of the sudoku and further pre-processed with binarization, normalization and reshaping.

C. Pre-Trained digits recognition model

This model has been trained on the Chars74K dataset [1]. This dataset contains images of types of digits in different fonts. It contains four convolutional layers followed by alternating dropout and dense layers. The Rectifier Linear activation function (RELU) is used between the dense layers and the Softmax function is used in the output layer. Nine classes are present in the output layer depicting the



D. Image to grid conversion of sudoku puzzle

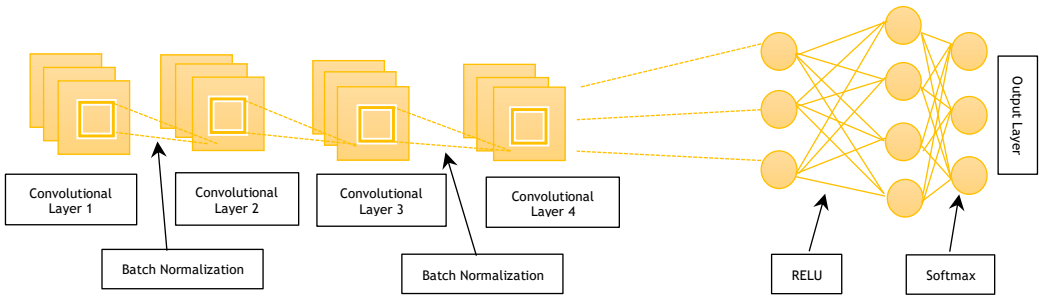
Once the image pre-processing of the image and cropped image is completed, we give that as input to the pre-trained digits recognition model. The model predicts the probability of each digit from 1 to 9. To differentiate between blank and populated cells in the sudoku image, we first calculate the number of white pixels for each of the cells containing digits. The maximum *max* of these values is used as a threshold to find blank cells using the condition

if (number of while cells in a cell) > max
current cell = blank cell

We then find the maximum probability of a digit in a cropped cell image and store these numbers predicted for each cell in a list and create a *list of list* data structure for every row in the 9x9 sudoku. This list is fed as input to the sudoku solving model.

E. Pre-trained model for solving sudoku

This model has been trained on the Sudoku dataset [2]. This dataset contains over one million sudoku games and their solutions. It contains four convolution layers followed by one dense layer for classification with an additional softmax layer.



F. Solution Output

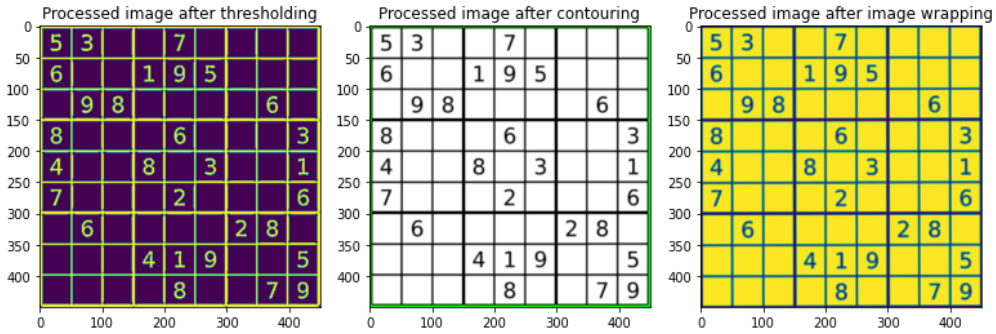
This is the final output of the project that produces the solved sudoku.

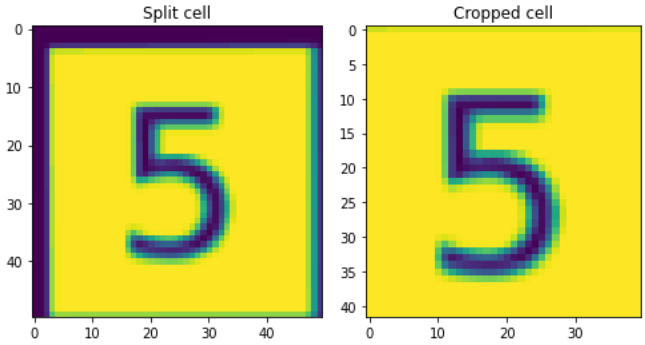
5. Implementation

The implementation has been divided into three major parts.

Image processing and detection

A 9x9 sudoku grid image is given as input. Pre-processing of the input as discussed above is performed. The implementation can be seen from the images below.





Digit Recognition

All 81 cell images including populated and blank cells are tested to recognize the digits in them as discussed above. After training the model on multiple datasets like MNIST and Char74K, we observed that the model gives maximum accuracy after training it on Char74K and thus used trained our model with it eventually. This model recognizes each digit in each cell image, calculates the probability of each number and outputs the most favorable digit.

Sudoku Solver

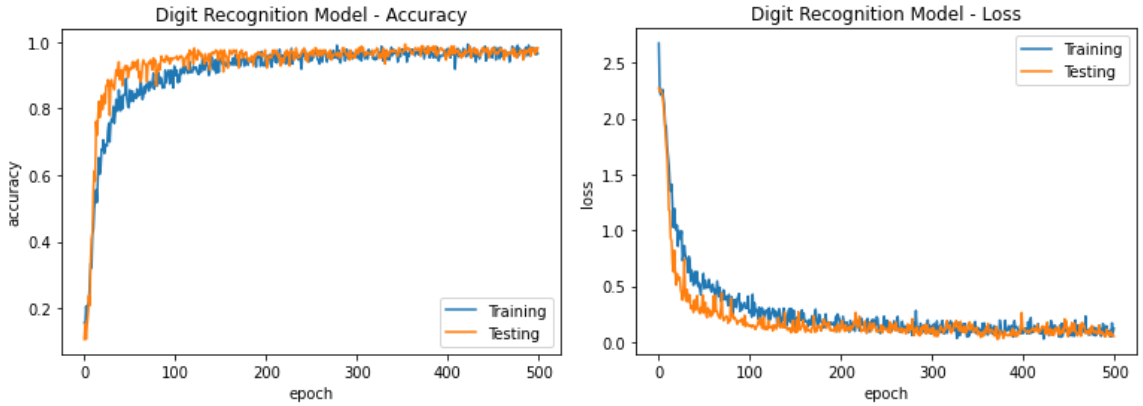
Sudoku solver model is trained using the Sudoku dataset [2], which consists of two columns –

i. Sudoku puzzle in string format, where 0s indicate the blank cells
   060720908084003001700100065900008000071060000002010034000200706030049800215000090
ii. Sudoku solution in string format
   163725948584693271729184365946358127371462589852917634498231756637549812215876493

We add a third axis to this data and normalize it by dividing and multiplying it by 9 and 0.5 respectively before feeding it to the defined Convolutional Neural Network model. We have made use of *Sparse Categorical Crossentropy* loss which computes the cross-entropy loss between the labels and respective predictions. We have used the Adam optimizer for faster conversion. This model has been trained for 2 epochs with 25000 steps each and a batch size of 64. The initial learning rate is set to 0.001for first epoch and this decays to 0.0001 by the completion of second epoch. The model predicts the digit to be filled in each of the blank cells one by one, just as any human would do and hence produces astounding accuracy. The output of this model is the solved sudoku solution.

6. Evaluation

On testing the digits recognition model on the Char74K dataset, we get a testing accuracy of 0.8888888955116272 and testing loss of 0.463793009519577



The model for solving sudoku gives an accuracy 0.99 and solves 99 out of 100 sudokus correctly.

7. Observations

As part of digit classification, we first tried training our model using the MNIST dataset but that gave very low accuracy of 40% and high loss of 60% thus not detecting the digits in the input correctly. Thus, we switched to the Char74K dataset later which gave better results.
For the second part of solving the sudoku, we tried to implement the Deep Q-Learning algorithm to solve it but that gave very low accuracy and required very high training time. Hence, we switched to a deep learning model later which gave better accuracy of 0.99.
We also tried implementing *pytesseract,* Google's optical character recognition engine, with OCR engine mode of 1 and 3 that makes use of neural nets, LSTM and Google's legacy engine, but this raised characterization issues while identifying blank cells in the digit recognition part and hence we could not go ahead with it.

8. Conclusion

Thus, using the convolutional neural network models and architecture as discussed above, along with *opencv*, *keras* and *tensorflow*, we have been able to come up with an efficient process for solving the sudoku puzzle. The process starts by taking the image as an input, then detecting the contour with the largest area, followed by detecting the digits from the image with the help of the first convolutional neural network model, and ultimately solving the sudoku using the second convolutional neural network model.

9. Future Scope

The accuracy of the digit recognition model is still not the best and can be further improved by training the models on more blurry and distorted images. There is also scope in exploring more deep reinforcement learning techniques like proximal policy optimization and prioritized experience replay for solving sudoku which will increase the training speed once the correct action space and reward set is defined for each action.

10. References

[1] http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/
[2] https://www.kaggle.com/datasets/bryanpark/sudoku
[3] https://towardsdatascience.com/solving-sudoku-with-convolution-neural-network-keras-655ba4be3b11
[4] https://www.kaggle.com/code/karnikakapoor/sudoku-solutions-from-image-computer-vision/
[5] https://thispointer.com/python-check-if-all-values-are-same-in-a-numpy-array-both-1d-and-2d/
[6] https://www.geeksforgeeks.org/opencv-counting-the-number-of-black-and-white-pixels-in-the-image/
[7] https://www.includehelp.com/python/cropping-an-image-using-opencv.aspx
[8] https://www.sitepoint.com/keras-digit-recognition-tutorial/
[9] https://arxiv.org/pdf/2102.06019.pdf
[10] https://analyticsindiamag.com/solve-sudoku-puzzle-using-deep-learning-opencv-and-backtracking/