Sudoku Solver

Niraj Gupta, Ragul Venkataraman Ravisankar, Aravindakumar Vijayasri Mohan Kumar

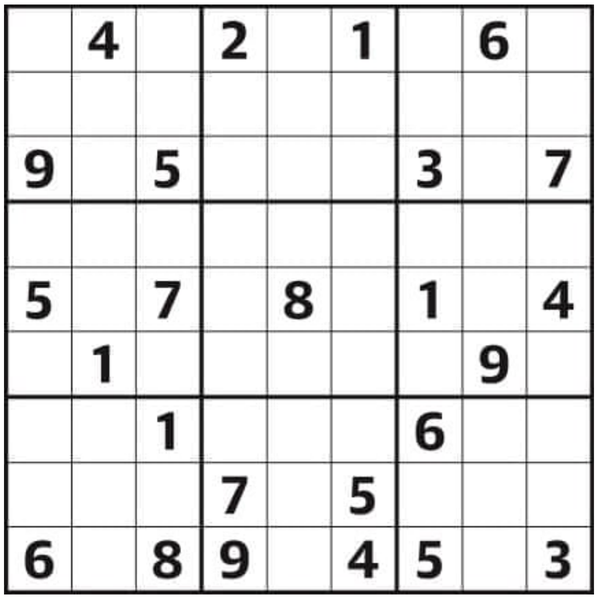
December 13, 2021

1 Problem Space

Sudoku is a popular number puzzle which has different variants with respect to grid size. For this project, we are considering the grid size of 9x9. Some cells are pre-filled with numbers from 1 to 9 and others are left blank. A player is supposed to fill the blank cells such that each row, each column and each sub-grid of size 3x3 have all the numbers from 1 to 9. This project is focused on building an intelligent system which, when given a Sudoku image, detects the digits and uses previously acquired knowledge and tries to solve it.

2 Dataset

2.1 Sudoku Image



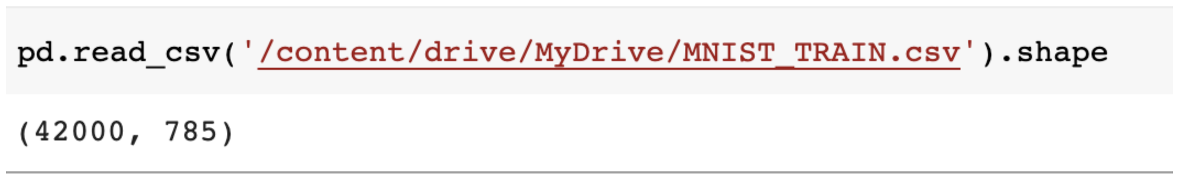
	4		2		1		6	
9		5				3		7
5		7		8		1		4
	1						9	
		1				6		
			7		5			
6		8	9		4	5		3

Figure 1: Sample image for Sudoku puzzle

This data consists of 9x9 Sudoku image extracted from the source mentioned above. This input is used to test our application.

2.2 MNIST data

This dataset contains handwritten digits of all the numbers from (0 to 9). The dataset consists of 42000 images and each image is 28x28 pixels (=784) and one column for class labels. So the dimension is 785 (784+1). We will use this dataset to recognize the digits of the Sudoku cell. It will replace the pre-filled Sudoku cell with the corresponding digits and the blank cell will be replaced as zeros.



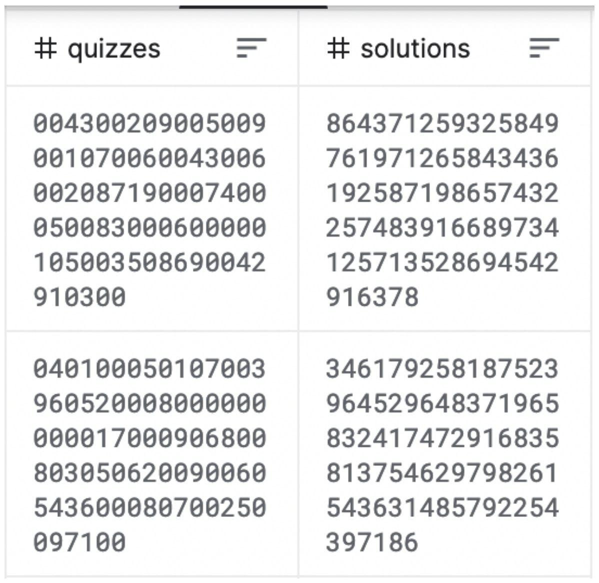
```
pd.read_csv('/content/drive/MyDrive/MNIST_TRAIN.csv').shape

(42000, 785)
```

Figure 2: MNIST data dimension

2.3 Sudoku dataset

Solved Sudoku data set consists of 1 million entries and it has two columns: one column represents the posed Sudoku question of existing numbers where blank cells are represented as 0 and the other column will be the solved Sudoku which denotes the solution to the solved problem. The dimension of this data set is (1000000 * 2). The model will be trained on this data to determine the solution to our input Sudoku image.



# quizzes	# solutions
004300209005009001070060043006002087190007400050083000600000105003508690042910300	864371259325849761971265843436192587198657432257483916689734125713528694542916378
040100050107003960520008000000000017000906800803050620090060543600080700250097100	346179258187523964529648371965832417472916835813754629798261543631485792254397186

Figure 3: Sudoku Dataset

3 Approach

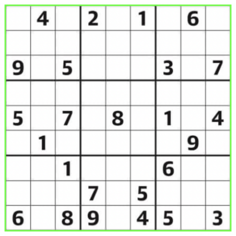
3.1 Image processing

The first step in solving a Sudoku puzzle is to extract the Sudoku board from a given image. Image processing involves the following steps:

3.1.1 Pre-Processing the image

For pre-processing the Sudoku image we use the Open-CV library. Pre-processing of the image involves three steps: Gaussian blur, threshold and dilation or inversion. The color has no significance as the model will be learning from the geometry present in the image. Converting to gray scale will help in sharpening and reducing the dimensions. To smoothen the image further, Gaussian blur is applied in which the image is convoluted with a Gaussian kernel. This also reduces noise obtained in threshold algorithm.

To detect the Sudoku in the image, we need to segment the image into some boundaries or regions and then detect the contours. For this, we apply adaptive threshold on the image and detect the contours.



(a) Contour detection

(b) cell extraction

3.1.2 Cell Extraction

Cell extraction involves detecting the Sudoku and creating single image for each cell. We can extract the Sudoku using the largest contour from the threshold image. For this, we first detect all the contours in the image and then filter the contour with largest area and 4 corners. We also apply perspective transformation to get the top view of the Sudoku image. Finally, we split the image into 81 cells which gives us a separate image for every cell in Sudoku.

3.1.3 Cell Prediction

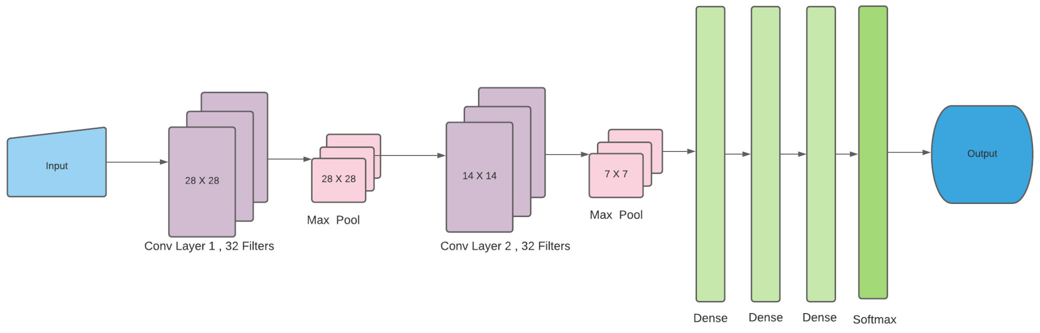


Figure 5: Digit prediction network

We use CNN and MNIST dataset for recognizing the digits. The MNIST dataset contains 60,000 small square 28×28 pixel gray scale training images of handwritten digits from 0 to 9 and 10,000 images for testing. The MNIST dataset has 10 different classes. Before training the network, we pre process the image by converting it 4 dimensions(1, 28, 28, 1) and also normalize the pixels. Three different models are trained on 10, 15 and 30 epochs respectively with test accuracy of approximately around 99%. All the cells that are extracted using Image processing are predicted using this trained network and stored as 1x81 matrix.

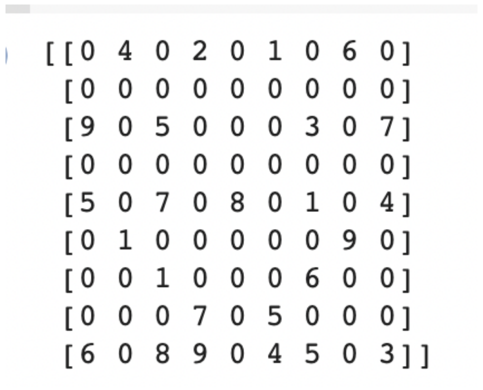
3.2 Solving the Sudoku

3.2.1 Backtracking

The first milestone in our project is successfully extracting digits from a given input image and then solve the Sudoku using a back-tracking algorithm. The image will be processed using the techniques mentioned above. Then, the processed image will be given as input to our model trained on MNIST data for digit recognition. The output of this model will be a 1x81 matrix. This matrix will be given to our Backtracking algorithm which solves the Sudoku.

3.2.2 Solving using Neural Network

The second milestone will be to solve the Sudoku puzzle using the output from the Digit Recognizer model. The output from the Digit Recognizer model will be given as input to Sudoku Solver model. The Sudoku solver model is expected to give a solved Sudoku as the output.



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

Figure 6: Predicted Digits by the network

We trained three Neural Network architectures to solve the Sudoku puzzle. One common issue that was observed with the three architectures was that it was replacing the pre-filled Sudoku cells with a different value and therefore the accuracy took a massive hit.

To improve the accuracy, instead of solving the whole puzzle at once we tried to predict each cell at a time. Because in Sudoku problem, after filling each empty cell, the probability of filling the numbers in the subsequent cells get influenced. Therefore, we tried to solve one cell at a time and this improved the accuracy significantly for the test data. Since we are solving each cell at a time, the time taken to solve the entire Sudoku puzzle by the Neural network is significantly higher than the backtracking approach.

3.2.3 Model 1: 3-Layer Architecture

This model has three convolution layers. The first two layers had a kernel size of (3x3) with a channel size of 64. The third convolution layer has a kernel size of (1x1) with 128 channels. The batch normalization was done after every convolution layer. This model was trained on one million solved Sudoku dataset.

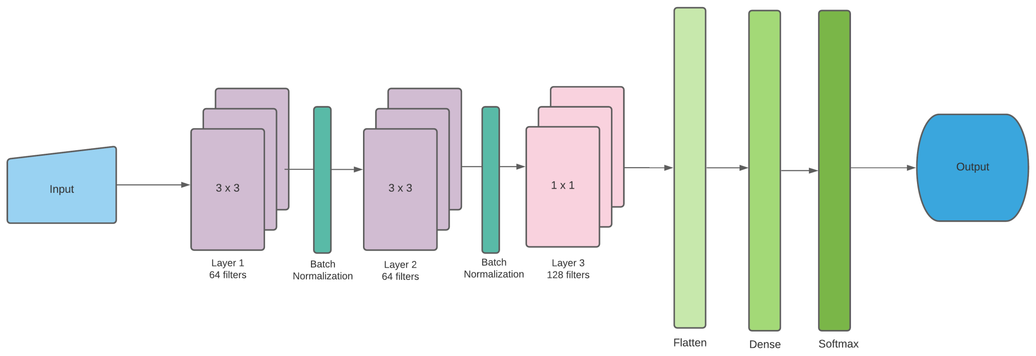


Figure 7: 3 Layer Architecture

3.2.4 Model 2: 5-Layer Architecture

This model has five convolution layers. The first four layers had a kernel size of (3x3) with a channel size of 64. The last convolution layer has a kernel size of (1x1) with 128 channels. The batch normalization was done after every convolution layer. This model was trained on two million solved Sudoku dataset.

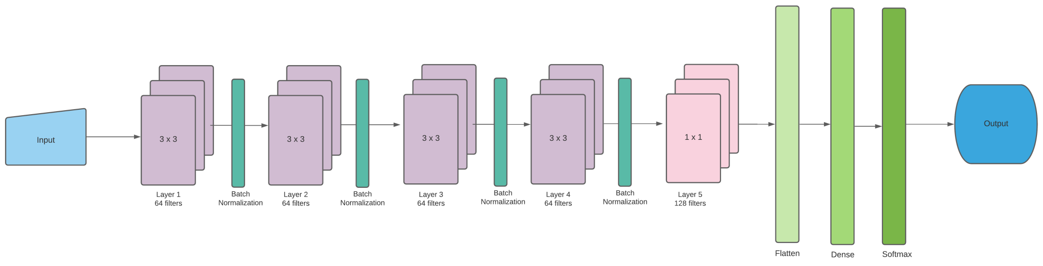


Figure 8: 5 Layer Architecture

3.2.5 Model 3: 15-Layer Architecture

This model has fifteen convolution layers. The first fourteen layers had a kernel size of (3x3) with a channel size of 512. The last convolution layer has a kernel size of (1x1) with 512 channels. The batch normalization was done after every 5 convolution layers. This model was trained on one million solved Sudoku dataset.

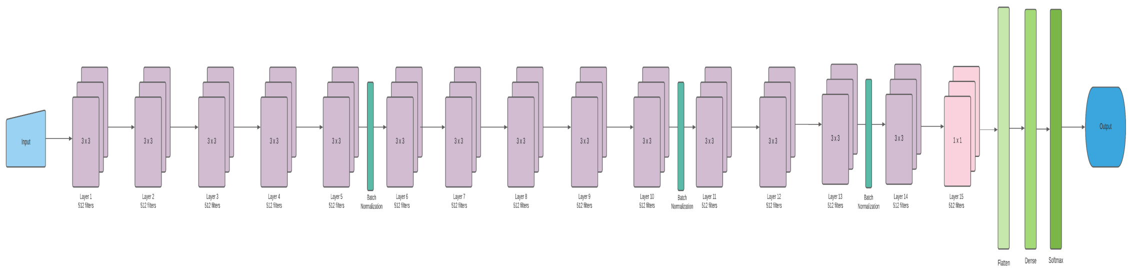


Figure 9: 15 Layer Architecture

4 Results

4.1 Comparison of Time taken for each approach

From the below graph, we can observe that the model with 15 layer architecture takes the most time for predicting 200 Sudoku samples and the backtrack approach is the fastest. The time taken by the neural network models are greater because they are solving one cell at a time rather than solving entire Sudoku puzzle.

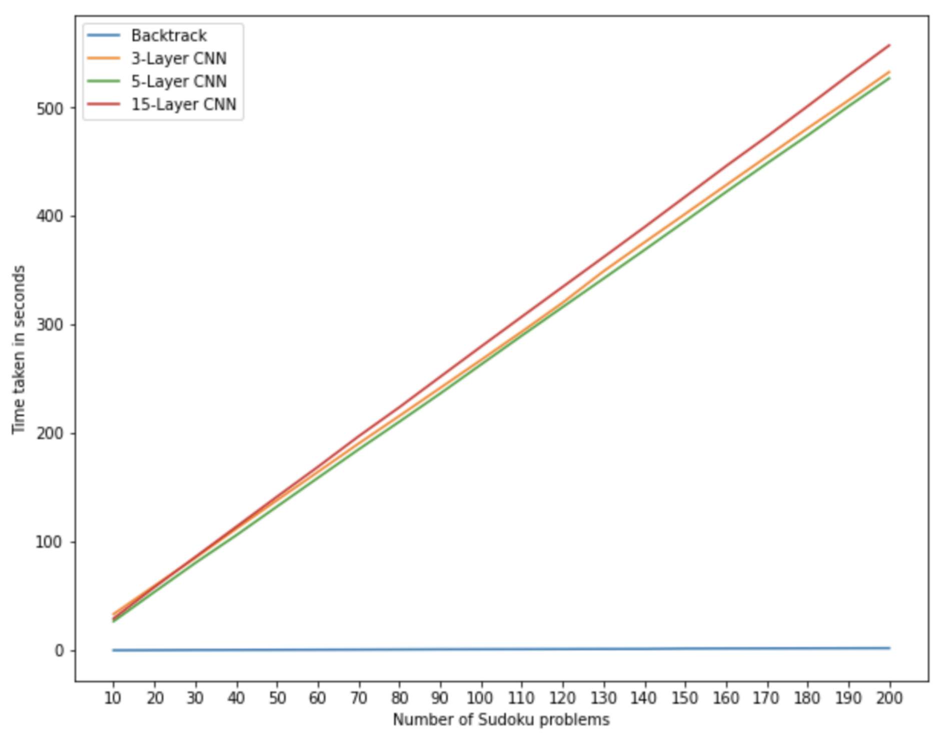


Figure 10: Time Comparison

4.2 Comparison of Accuracy Metric

From the graph below, we can observe that the 5-layer model trained on two million dataset gives 100% accuracy whereas 3-layer model trained on one million dataset gives an accuracy of about 94% and 15-layer model trained on one million dataset yields an accuracy of 91%.

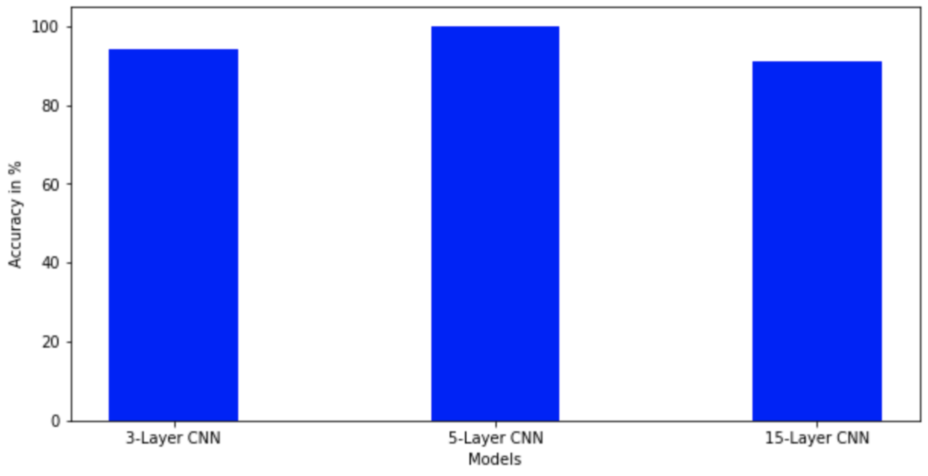


Figure 11: Accuracy metric

5 Discussion

One of the assumptions made while considering the image processing of Sudoku was that the puzzle is biggest element in the image. If the image has elements bigger than the Sudoku then detection of the contour would fail. Some of the other issues when doing this were:

- The inner grid lines are very fine and were tough to recognize.
- The lines are not perfect. if the lines have gaps then we would have to close it to detect contour.

The resolution of the image can also change the cell prediction accuracy. From the results, we noticed that higher resolution images required network trained on more epochs whereas lower resolution images required less. (431x431 image had 100% accuracy on model trained on 10 epochs and 1525x1525 had 100% on model trained on 15 epochs).

One of the first approaches was to solve the entire Sudoku problem using Neural Network models at once. The input to the network was 1x81 matrix of the entire Sudoku. The results were unexpected as the model changed the pre-filled cells in Sudoku and accuracy took a massive hit. To improve the accuracy, we are currently using neural network models to solve cell by cell at a time rather than solving the whole puzzle at once. This improved the accuracy significantly but the time taken to solve the puzzle is also high. In future, we would like to explore how to reduce the time taken for solving a puzzle entirely at once without impacting the accuracy.

6 Code

Implementation details can be found here : **Sudoku Solver**

References

[1] Akin-David, C. and Mantey, R., 2021. Solving Sudoku With Neural Networks. [online] Cs230.stanford.edu.

[2] Saha, S. (2018, December 15). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. Towards Data Science; Towards Data Science.

[3] Handwritten Digits Classification : An OpenCV (C++ / Python) Tutorial — LearnOpenCV . (2017, January 30)..

[4] Jason Brownlee. (2019, April 16). How Do Convolutional Layers Work in Deep Learning Neural Networks? Machine Learning Mastery.

[5] HENRIK, V. (n.d.). Performance and Scalability of Sudoku Solvers (M. VIKTOR, Ed.) Review of Performance and Scalability of Sudoku Solvers. NADA..

[6] Ercsey-Ravasz, M., Toroczkai, Z. (2012). The Chaos Within Sudoku. Scientific Reports. .

[7] Digit Recognizer. (n.d.). Kaggle.com..

[8] One million Sudoku games. (n.d.). Kaggle.com..

[9] Nine Million Solved Sudoku Puzzles. (n.d.). Kaggle.com..

[10] Sudoku images in PDF or HTML formats.

[11] A Beginner's Guide to Neural Networks and Deep Learning.