Capstone Project Proposal

By - Pratyush Bhatnagar

Solving Sudoku with Convolution Neural Network | Keras

1.) Domain Background

I used to solve **sudoku** a long time ago. A few days back I was wondering if I can solve it with Convolution Neural Network(**CNN**). I knew Sudoku has spatial features since it has a particular arrangement of numbers and CNNs are good at extracting spatial features.

2.) Problem Statement

A project to create a Sudoku solver from inputs for educational purposes to explore themes of:

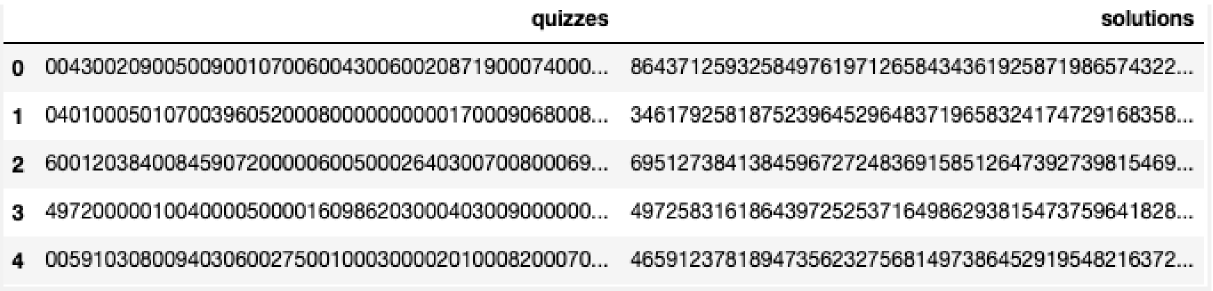
- Puzzle Solving Algorithms
- Machine Learning
- Deep Learning

The Sudoku class can take an input as an array of natural numbers and produce solved puzzle as an output. Output can be as a formatted string, dictionary or an array of string.

3.) Datasets and Inputs

Data Collection

- Download the **dataset** for this project. I found the following data on Kaggle, which contains 1 million unsolved and solved Sudoku games. Please take a look at the data below.



	quizzes	solutions
0	0043002090050090010700600430060020871900074000...	8643712593258497619712658434361925871986574322...
1	0401000501070039605200080000000000170009068008...	3461792581875239645296483719658324174729168358...
2	6001203840084590720000060050002640300700800069...	6951273841384596727248369158512647392739815469...
3	4972000001004000050000160986203000403009000000...	4972583161864397252537164986293815473759641828...
4	0059103080094030600275001000300002010008200070...	4659123781894735623275681497386452919548216372...

Sudoku Dataset

The dataset contains 2 columns. The column quizzes has the unsolved games and the column solutions has respective solved games. Each game is represented by a string of 81 numbers. Following is a 9x9 sudoku converted from the string. The number 0 represents the blank position in unsolved games.

4.) Solution Statement

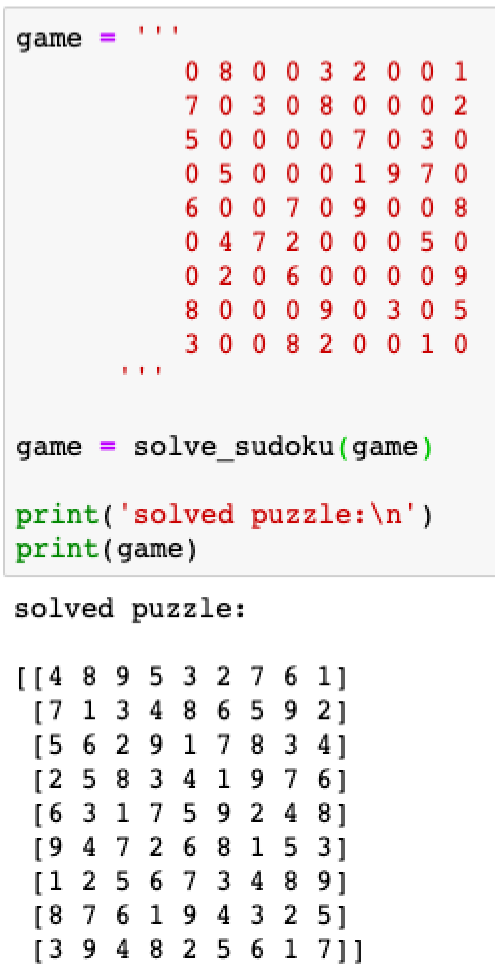
I trained the network for 2 epochs, with batch size 64. The learning rate for the first epoch was 0.001 and for second epochs I reduced it to 0.0001. The final training loss settled down to 0.34. I tried a few different network architecture and strategies but could not reduce the loss further so I went ahead with this network. Its time to test the network.

Now, I tried to solve the game using our trained network. I saw that the network always predict few values wrong. Following is a game predicted by the network. You can see a few numbers repeating in rows and columns.

```
> Input (Unsolved)
[[0 1 6 9 0 4 0 0 7]
 [0 0 4 0 3 0 0 8 0]
 [0 0 3 0 6 1 9 2 0]
 [5 0 9 1 4 0 8 0 0]
 [1 7 0 0 0 0 0 0 0]
 [0 0 8 7 0 0 0 6 5]
 [6 0 0 0 0 2 0 4 0]
 [0 2 0 8 0 5 3 1 0]
 [0 3 0 0 0 0 0 0 9]]> Output
[[2 1 6 9 8 4 5 3 7]
 [2 9 4 2 3 7 6 8 1]
 [7 8 3 5 6 1 9 2 4]
 [5 6 9 1 4 6 8 7 3]
 [1 7 2 5 5 8 4 9 4]
 [4 4 8 7 2 9 1 6 5]
 [6 9 1 3 1 2 5 4 8]
 [9 2 7 8 9 5 3 1 6]
 [4 3 1 4 1 6 2 5 9]]
```

I had to try something else instead of changing network architecture to solve the game since training loss was not going below a certain number.

5.) Benchmark Model



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

game = solve_sudoku(game)

print('solved puzzle:\n')
print(game)
```

```
solved puzzle:

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

Network was able to solve the puzzles with 99% accuracy.

6.) Evaluation Metrics

For evaluating the performance and accuracy, we want to output the correct solved sudoku as an array of string. So,

1.) Accuracy %age = (total no. of correct outputs / total no. of inputs)   x 100

2.) The sum of every row will be (1+2+3+4+5+6+7+8+9) = 45.

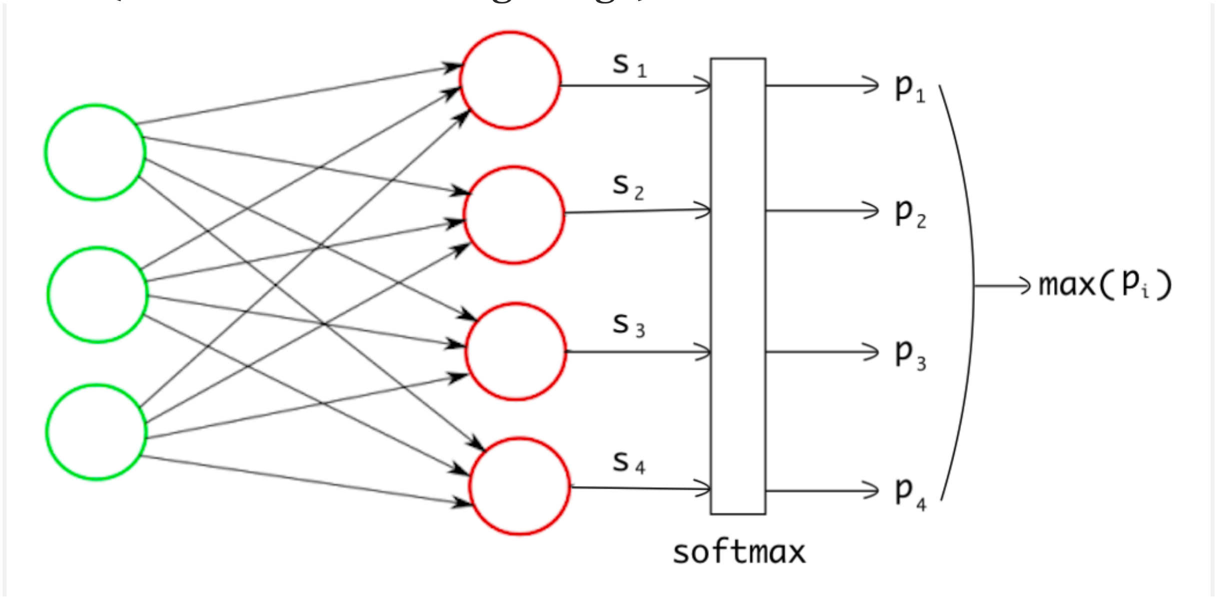
Therefore, we can check the sum of every row as 45.

7.) Project Design

Our task is to feed the unsolved sudoku to a neural network and get the solved sudoku out of it. This means we have to feed 81 numbers to the network and need to have 81 output numbers from it.

We have to convert the input data(unsolved games) into a 3D array since we have to feed it to the CNN. I have converted each string of 81 numbers in a shape of (9,9,1). Then I normalized the input data by dividing it with 9 and subtracting 0.5. By doing so data becomes zero mean-centred and in the range of (-0.5 - 0.5). Neural networks generally perform better with zero centred normalized data.

In a typical multi-class classification, the neural network outputs scores for each class. Then we apply softmax function on the final scores to convert them into probabilities. And the data is classified into a class that has the highest probability value(refer to the following image).



But in sudoku, the scenario is different. We have to get **81** numbers for each position in the sudoku game, not just one. And we have a total of **9** classes for each number because a number can fall in a range of 1 to 9.

To comply with this design, our network should output 81x9 numbers. Where each row represents one of the 81 numbers, and each column represents one of 9 classes. Then we can apply softmax and take the maximum along with each row so that we have 81 numbers classified into one of the 9 classes.

I created the following simple network for this task. The network consists of 3 **Convolution** layers and one **Dense** layer on top for classification.

Note I am reshaping the output of the Dense layers in a shape of (81, 9) then adding a softmax layer on it. I compiled the model with sparse categorical crossentropy loss and adam optimizer.

Since we are using scc loss, we don't need to provide a one-hot encoded target vector. Our target vectors shape is (81, 1) where the vector elements represent the true class of 81 numbers.
