Capstone Project Proposal

By - Pratyush Bhatnagar

Solving Sudoku with Convolution Neural Network | Keras

1.) Domain Background

When I was in my school days, I used to solve sudoku puzzles which I find very interesting now also, but I always had in my mind that there would be some technique to solve it using AI. 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.

CNNs are an extension of multilayer perceptrons, which can learn filters that need to be computed by the machine learning models. Since the training process involves learning about patterns from smaller patterns, computations usually are time-consuming and may require GPU-based implementation

A **convolutional neural network** consists of distinct hidden layers in addition to the input and output layers. These distinct layers usually consist of convolutional layer with filters that can be learned, rectified linear unit layer for application of activation function, pooling layer for down-sampling and loss layer for specification of penalization for incorrect output. We use Keras with TensorFlow as backend to process the CNN model and provide a comparison between the distinct CNN implementations formed by choosing different hyperparameters associated with each layer.

```
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]]
```

0's means black positions to be filled.

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

This is a **Supervised Learning** technique where we have a pair of array of string as input and the corresponding output. The model gets trained on the dataset we provide.

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.) <u>Datasets and Inputs</u>

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	004300209005009001070060043006002087190007400...	864371259325849761971265843436192587198657432...
1	040100050107003960520008000000000017000906800...	346179258187523964529648371965832417472916835...
2	600120384008459072000006005000264030070080006...	695127384138459672724836915851264739273981546...
3	497200000100400005000016098620300040300900000...	497258316186439725253716498629381547375964182...
4	005910308009403060027500100030000201000820007...	465912378189473562327568149738645291954821637...

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

As humans when we solve Sudoku, we fill numbers one by one. We do not simply look at the sudoku once and fill all the numbers. The advantage of filling the numbers one by one is that each time we fill a number we keep getting a better idea about the next move.

I implemented the same approach while solving the sudoku now. Instead of predicting all 81 numbers at once, I am picking just one number among all blank position that has the highest probability value, and filling that number in the sudoku. After filling one number we again feed this puzzle to the network and make a prediction. We keep repeating this and filling the blank positions one by one with the highest probability number until we are left with no blank positions.

This approach boosted the performance and the network was able to solve almost all the games in this dataset. Test accuracy on 1000 games was 0.99.

The solution path includes-

- Importing necessary libraries such as pandas, numpy, sci-kit learn etc.
- Loading and getting data for the model to get trained.
- 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

Most of the sudoku solvers use **Recursion** or **Backtracking** as a way to solve sudoku problems. One such example can be using MATLAB-

Solving a Sudoku puzzle in MATLAB may not be a challenge, but to solve it quickly is not so easy. More than 20 Sudoku solvers have been submitted in MATLAB File Exchange. Most of them use recursive approaches, which, without non-recursive algorithm support, could be very inefficient.

It maintains a Candidate Map (C, a 81 x 9 logical array) to discover the solution. Initially, without any number on the Board (B, 9 x 9 array), all elements of C is true, indicating that at any position, there are 9 candidates. Each of 9 x 9 grids belongs to a row, a column and a block, marked as (r,c,b). Let 81 x 1 vectors I, J and K represent row, column and block indices of all grids respectively. When a specific number, n is put into the grid (r,c,b), i.e. B(r,c)=n, then all grids in L=(I==r | J==c | K==b) should not have candidate n, hence C(L,n)=false. If a row of C has only one candidate, i.e. L=sum(C,2)==1, then, the corresponding grid is solved, i.e. B(L(k))=find(C(L(k),:),'first'). Many other rules have been implemented to update C.

If a puzzle cannot be completely solved by the non-recursive solver, a recursive solver will be called. The recursive solver will always try the grid, which has the minimum number of candidates indicated by C. For example, a grid m has two candidates, n1 and n2. The recursive solver will set B(m)=n1 and call the non-recursive solver to find the rest solutions. If an error is returned by the non-recursive solver, then the recursive solver will set B(m)=n2 to call the non-recursive solver again. If both branches are failed, then the puzzle is not valid.

Due to Recursion and Backtracking, the Time Complexity of the model gets increased and gives delayed outputs. So, for the sake of time and other factors, I chose Deep Learning to be the best approach to solve sudoku puzzles. This model would be the best benchmark for other sudoku solvers.

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.

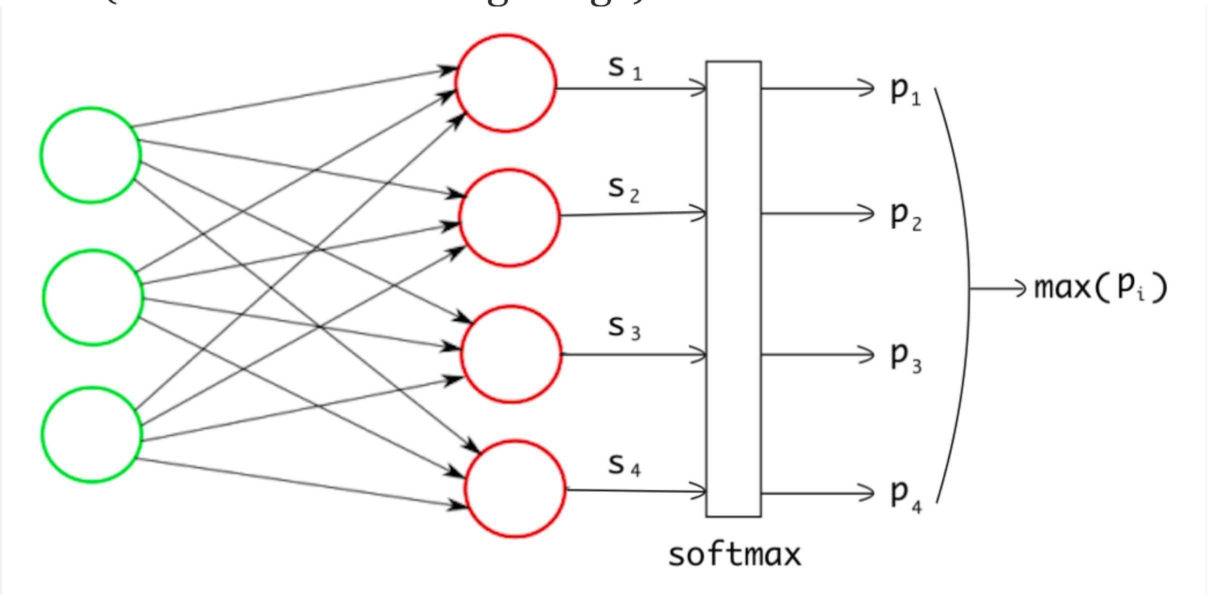
Network was able to solve the puzzles with 99% accuracy.

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.
