VUDOKU - A VISUAL SUDOKU SOLVER

PREPRINT, COMPILED JUNE 7, 2021

Jovial Joe Jayarson [+*] **and Ebin P M** [+]

[+]Department of Computer Science & Engineering, IESCE
[+]APJ Kalam Technological University

ABSTRACT

It is no secret that AI is an upcoming titan. Even though people are stunned to hear that AI has been here for around a century, due to the advancement in computational methods and resources, today AI peaks like never before. As a tiny glimpse into the field of Digit Recognition, this project aims to understand the underlying cogs and wheels on which the neural networks spin. This paper tries to elucidate a project which solves the Sudoku puzzle drawn and written by hand. The paraphernalia for that project includes programming language: Python3; libraries: OpenCV, Numpy, Keras/Tensorflow; datasets: MNIST handwritten digit database. Digit recognition is a classical problem which will introduce neurons, neural networks, connections hidden layers, weights, biases, activation functions like sigmoid, back-propagation and other related topics as well. Algorithm(s) in the project employed to solve Sudoku is also explored in this paper.

Keywords Machine Learning · Artificial Intelligence · Computer Vision · Sudoku Puzzle · Neural Networks · Back Tracking

1 INTRODUCTION

Machines are inherently incapable to understand, text, images and audio. These formats which carry information, needs to be converted into numbers, matrices or vectors which is a significant step, in any procedure, to solve problems using computers. Another vital stage for digit recognition is *machine learning*. With classical problem solving techniques, humans provide computers with both data and rules as input. In contrast, we provides data and presumed result, as input to a machine learning system. And then we let it find out some *rules*. A quick example would be to give three numbers x, y and z and tell a machine that - certain operation was performed on x and y to obtain z. Once a machine is fed with lots and lots of similar examples, it will be equipped find out some pattern behind it. At a later stage, it can very well predict, what operations might have been performed on x and y to result in z. Sudoku is a mathematical problem and by default has 9×9 grids. There are other variant to it but to keep things simple, the project moves along with the default one. Sudoku is classified as Exact Cover or Hitting Set problem [1]. Various algorithms has been developed over the years to solve it, like Backtracking, Stochastic algorithms, Constraint programming and so on. Given that the output of the digit recognition system is as *value*, *position* pairs or and *encoded string*, this project uses simple backtracking algorithm to solve Sudoku.

2 CONTEMPORARY TECHNOLOGIES

Complex problems exists all around us. Some of them are stumbled upon and other have be thoughtfully crafted. Solutions to these problems at times are intuitive and yet difficult to explain. Structured mathematical approach to problems has engineered some of the finest solutions to some of the toughest problem. In the era of computing, appetite to solve problems using machines has become a hobby. With technology at fingertips, it's no-brainer that upcoming generations have a choice to be way smarter. It's of no surprise that the field of artificial intelligence has come a long way. This section deals with some of the famous models and how they are related to digit recognition. Starting from the ones released in 80's like *Recurrent Neural Network (RNN)* to the latest 2017 model *The Transformer*, much study has been conducted.

2.1 *Recurrent Neural Network*

Basic RNNs are a network of neuron-like nodes organized into successive layers. Each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer. The following figure (1) gives the internal schematics of Recurrent Neural Networks.

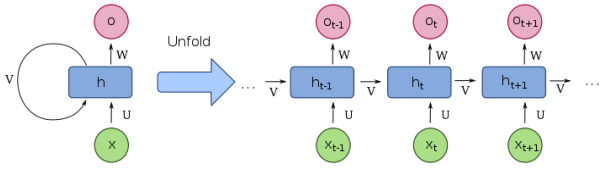

Figure 1: Recurrent Neural Networks Unfolded [2]

Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. Nodes are either input nodes (receiving data from outside of the network), output nodes (yielding results), or hidden nodes (that modify the data en route from input to output). Each sequence produces an error as the sum of the deviations of all target signals from the corresponding activations computed by the network. For a training set of numerous sequences, the

*correspondence: jovial7joe@hotmail.com

total error is the sum of the errors of all individual sequences. The following equation (1) was proposed as part of research on *Simple Recurrent Neural Network* [2]:

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y) \quad (1)$$

Where, x_t is input vector, h_t is hidden layer vector, y_t is output vector, W, U and b are parameter matrices and vector, respectively σ_h and σ_y are activation functions.

Recurrent neural networks models contain a self-connected hidden layer. One benefit of the recurrent connection is that a 'memory' of previous inputs remains in the network's internal state, allowing it to make use of past context. This is significant for handwriting recognition as mentioned in [3]. A major problem with gradient descent for standard RNN architectures is that error gradients vanish exponentially quickly with the size of the time lag between important events [2].

2.2 *Long Short Term Memory*

Long short-term memory (LSTM) is a deep learning system that avoids the vanishing gradient problem. LSTM is normally augmented by recurrent gates called "forget gates". LSTM prevents backpropagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier [2].

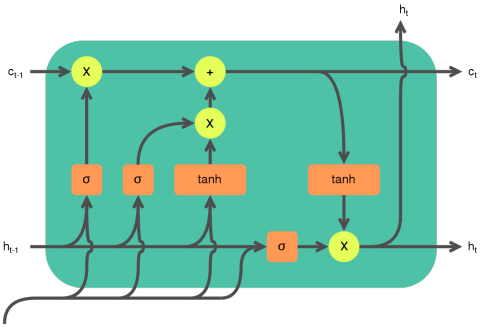

Figure 2: Long Short Term Memory Cell [4]

A common LSTM unit, as shown in figure (2), is composed of a cell, an *input gate*, an *output gate* and a *forget gate*. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. Some variations of the LSTM unit do not have one or more of these gates or maybe have other gates. For example, *Gated Recurrent Units* (GRUs) do not have an output gate. LSTM networks are hence well-suited for classifying time series data.

2.3 *Convolutional Neural Networks*

A convolutional neural network (CNN), is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or *space invariant artificial neural networks* (SIANN). CNNs have a wide variety of applications including, image and video recognition, recommender system, brain-computer interface etc. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. CNNs use relatively less pre-processing compared to other image classification algorithms. Independence from prior knowledge and human effort in feature design is a major advantage [5]

3 Project Walk-through

The goal of the project as stated in the abstract and introduction section (1), is to solve a Sudoku puzzle which is provided as an visual input to the computer. Being a multidisciplinary problem, it contains various modules laid out neatly. Below is the directory structure of the project:

```
|-- src/
    |-- samples/
    |-- classifier.py
    |-- extractor.py
    |-- main.py
    |-- scavenger.py
    |-- solver.py
|-- README.md
|-- LICENSE
```

As obvious the `main.py` script plays the pivotal role of managing the control flow. It's a Streamlit application. The UI layer looks like as shown in figure (3) and (4).

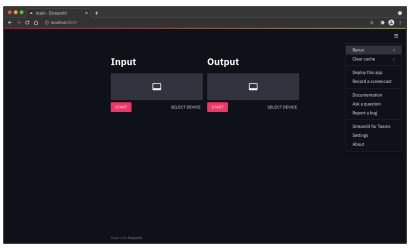

Figure 3: Vudoku UI

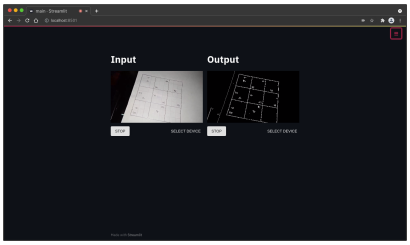

Figure 4: Edge Detection

3.1 *Scavenger*

This module is what takes the input and in human terms tries to understand it. It expects a similar input, where Sudoku is prominent in the input image as shown in (3).

Figure 5: Sample Input

Figure 6: Contours Points

It finds contours points that encapsulate the maximum area which is shown in (4). Finally performs a perspective transform that takes to contours points, extracts that area and superimposes it on a blank image to obtain the resultant image as in (5)

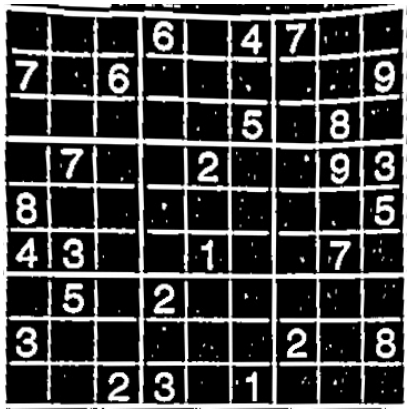

Figure 7: Transformed Sudoku

3.2 *Extractor*

The extractor module grabs each cell from the transformed Sudoku board and arranges them in a list. Then it loads a classifier (3.4) and classifies the image data in the cell into a number. Each of these number are string concatenated to obtain a 81 bit long string after the process.

3.3 *Solver*

Solver is an implementation of backtracking algorithm employed to solve Sudoku in specific. It verifies uniqueness in each row column and sub box while recursively filling out the puzzle. Finally it spits out an 81 bit long solution string. This string can be later manipulated to one's liking.

3.4 *Classifier*

The digit classifier module does not actively participate in the control flow, the reason being the existence of a saved model. If need arises for a more granular control, then a new model must be generate tuning the hyper-parameters. Speaking of which this model is a sandwich of

```
2D Convolution Network with ReLu activation
↪ + another 2D Convolution Network with ReLu
↪  activation + a 2D Max-Pooling layer +
↪ Flattened Layer + Dense Network with ReLu
↪  activation + Dense Network with SoftMax
↪ activation
```

The CNN architecture is best depicted by figure (8)

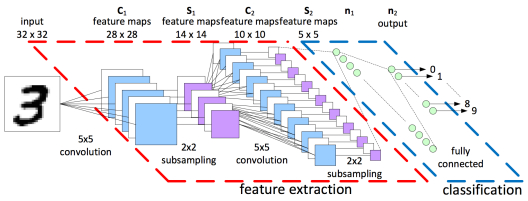

Figure 8: Digit cells

4 Conclusion

Digit recognition is an interesting classical problem to embark the journey in the field of artificial intelligence. Sudoku, a mathematical puzzle is captivating beauty of the mystery that lies in various patterns and combinations in itself. Visualizing Sudoku will give an insight on many aspects, including accuracy, performance, complexity, precision, recall etc. This project could be an inspiring stepping stone for futures enthusiasts to work upon.

References

[1] Wikipedia contributors. Exact cover — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Exact_cover#Sudoku, 2020. [Online; accessed 5-December-2020].

[2] Wikipedia contributors. Recurrent neural network — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=991832590, 2020. [Online; accessed 5-December-2020].

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009. doi: 10.1109/TPAMI.2008.137.

[4] Wikipedia contributors. Long short-term memory — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=991684445, 2020. [Online; accessed 5-December-2020].

[5] Wikipedia contributors. Convolutional neural network — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=992073762, 2020. [Online; accessed 5-December-2020].