International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-8, Issue-11S2, September 2019

Augmented Reality on Sudoku Puzzle using Computer Vision and Deep Learning

Azhar Talha Syed, Suresh Merugu

Abstract*— Many people try solving Sudoku puzzles everyday. These puzzles are usually found in newspapers, magazines and so on. Whenever a person is unable to solve a puzzle or is running short on time to solve the puzzle, it will be very convenient to show the solved puzzle as an augmented reality. Objectives: In this paper, we propose an optimal way of recognizing a Sudoku puzzle using computer vision and Deep Learning, and solve the puzzle using constraint programming and backtracking algorithm to display the solved puzzle as augmented reality. Also, a comparative performance analysis with the previous work is provided with this paper. Methods: In order to implement augmented reality on to the Sudoku puzzle, image classification itself won't be sufficient as the solved puzzle has to be shown on top of the area of the unsolved puzzle in the original image. So puzzle detection has to be performed and for doing so we used CNN and Object Localization algorithms. After the detection we stored the values detected in each 9x9 cells and ran a constraint programming and backtracking algorithm to solve the puzzle and finally filled the detected empty cells with correct values of the solved puzzle. Applications/Improvements: Usually the Sudoku puzzles that we find in newspapers and magazines are surrounded by a lot of noise such as text (characters) irrelevant to the puzzle and borders of the newspaper which could be similar to a Sudoku puzzle structure. In this paper we emphasise on how to handle such disturbances and improve the performance.

***Index Terms*—Object Localization, Sudoku, CNN, Augmented Reality, Computer Vision**

I. INTRODUCTION

Ever since the breakthrough of AlexNet in the imagenet challenge, CNN's have become one of the most prominent algorithms in image classification problems. However, just classifying whether or not the image contains a Sudoku puzzle is not enough. The bounding box of the Sudoku puzzle has to be identified and inorder to do so, we used Object Localisation [1]. The reason for not using other algorithms such as Selective Search [2], YOLO [3] or Region Proposals [4] for finding the bounding box is, all these algorithms are computation- ally expensive when compared to Object Localisation. This is another important performance criteria as the number of frames per second has to be taken into consideration for a better AR experience, as the computation increases the number of frames per second decreases. Moreover, these algorithms are very effective when detecting multiple objects in a frame and undergo additional computation, such as in the case of selective search a support vector machine is trained to find the regions of interest [2]. Where as in our problem we are only trying to detect one class which is the Sudoku board so Object Localisation is a good solution for this problem as the cost of finding the bounding box is very less and it works well for detecting a single object in a frame [1]. After finding the bounding box of the Sudoku board we iterate over each of the 9x9 cells, on each cell classification using CNN is performed to find the digits in the cell. For training this CNN we used the MNIST dataset [6] and achieved a 99.67% accuracy on the test data. The CNN used for performing Object Localization has an architecture similar to AlexNet [5]. The architectures and the training methodologies used for training Object Localization for detecting the Sudoku board and CNN for classifying digits will be further illustrated later in Sect. 3.

Fig. 1. A typical Sudoku Puzzle found in daily newspaper.

This paper is organised as follows. In Sect. 2, it describes the previous work done on solving Sudoku using augmented reality. In Sect. 3, an illustration of steps involved to solve the problem is provided along with a description of methodologies used in training Deep Neural Networks used in the solution. In Sect. 4, an analysis of solution is provided and results. Finally, in Sect. 5, a conclusion on the work done is provided with a suggested solution.

II. RELATED WORK

In the year 2012, Pramod J Simha, Suraj K v, and Tejas Ahobala, proposed Recognition of Numbers and Position using Image Processing Techniques for Solving Sudoku Puzzles [12]. For detecting the Sudoku Puzzle they made an assump- tion that the largest contour in the frame is the Sudoku Puzzle. However, this could not be true everytime as there could be larger contours in a frame other than the

Revised Manuscript Received on September 10, 2019.
Azhar Talha Syed, Department of Computer Science and Engineering, CMR College of Engineering & Technology, Hyderabad, Telangana, India
(E-mail: syedazhartalha@gmail.com)
Suresh Merugu, Research and Development Centre, CMR College of Engineering & Technology, Hyderabad, Telangana, India
(E-mail: msuresh@cmrcet.org)

Sudoku Puzzle. After finding the largest contour they proposed to split the contour into 81 parts, each part representing a cell as a Sudoku contains 9 x 9 cells. On each cell a template matching algorithm was used using template data to find the digits in each cell. After finding all the values in each the Sudoku was solved using backtracking.

Snigdha Kamal, Simarpreet Singh Chawla, Nidhi Goel pro- posed, "Detection of Sudoku Puzzle using Image Processing and Solving by Backtracking, Simulated Annealing and Ge- netic Algorithms: A Comparative Analysis " [13] in 2015. For detecting the Sudoku Puzzle these authors used an algorithm similar to the one proposed by Pramod J Simha, Suraj K v, and Tejas Ahobala, which is assuming that the largest contour in the frame is the sudoku puzzle. After finding the puzzle, OCR was performed on each cell and the values corresponding to each cell were stored. Three algorithms were used to solve the Sudoku which were Backtracking, Simulated Annealing and Genetic Algorithm. The results of these algorithms were compared with different difficulties of the Sudoku problems and their performance analysis was made based upon the execution time that each problem took to be solved.

In 2005, Helmut Simonis, proposed Sudoku as a Constraint Problem [9]. A common way of solving Sudoku is by using Backtracking algorithm, which is basically a blind search algorithm similar to Depth First Search Algorithm (DFS). As these blind searches can take exponentially large time in some cases, Helmut proposed a better algorithm which doesn't involve search. He viewed Sudoku as a constraint problem which means choosing an answer based on some set of constraints. This will really boost the speed of solving the problem. But however, the solution will not work for all the problems and in such cases Backtracking has to be used again. For detecting the Sudoku puzzle, A. Van Horn proposed the use of Hough transformation to detect the puzzle in his literature, ""Extraction of sudoku puzzles using the Hough transform"" [14]. The same algorithm was used by Baptiste Wicht and Jean Hennebert in their work, "Mixed handwritten and printed digit recognition in Sudoku with Convolutional Deep Belief Network" [15]. After detecting the Sudoku Puz- zle, Baptiste Wicht and Jean Hennebert used Convolutional

Deep Belief Network to classify the digits Keeping the previous works in mind we have worked on improvising the existing techniques especially in the case of Sudoku Puzzle detection. And also provided a full solution for the problem.

III. METHODOLOGY

For solving this problem we have divided it into four subproblems. They are:

- Detecting the Sudoku Puzzle.
- Classifying and storing the values in each cell.
- Detecting the Sudoku Puzzle.
- Classifying and storing the values in each cell.

A. Detection of Sudoku Puzzle

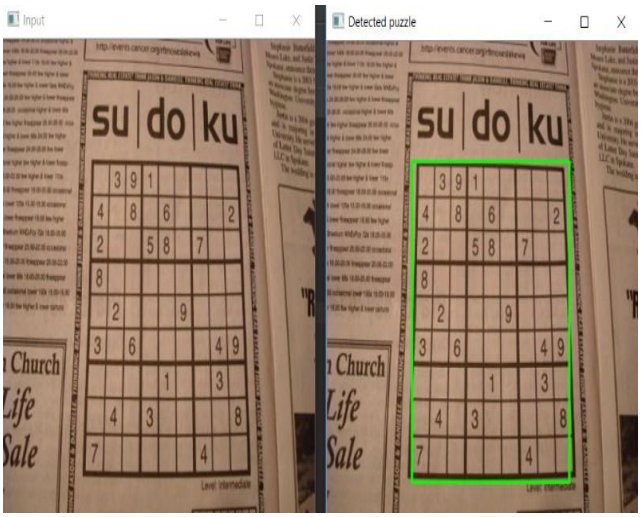


Fig. 2. Objective of the problem, (left) input frame, (right) detected Sudoku Puzzle.

The objective here is to find the bounding box of the Sudoku Puzzle in a frame. One way of doing this is finding the largest contour in the frame and assuming it to be the puzzle, calculating the areas of all the contours and treating the contour with the largest area as the puzzle. However, in real life there are a lot of disturbances in a frame, it is also possible that Sudoku puzzle is not even present in the frame. For example in Fig. 1. the puzzle is surrounded by another contour, which is the border of the paper, and if we use the same detection algorithm the border of the paper will be detected as the puzzle because it has a larger area compared to the puzzle. So this particular approach is not very effective and a better algorithm is needed for detecting the puzzle.

Sliding Window, in this algorithm a fixed set of sizes of windows are chosen and starting from the top left corner of the frame, part of the frame is chosen along with the size equal to the size of the window. This part of the frame is given as input to a classifier to know whether or not it is a Sudoku. Same is done for the whole frame with a fixed stride. Different window sizes are applied and the process is done again [11]. However, the algorithm is not very effective for this problem as it is hard to predict the exact window size which is equal to the size of the Sudoku Puzzle because precision is very important for implementing Augmented Reality on the same detected area.

Other detection algorithms such as Selective Search com- bined with CNN, YOLO, Region Proposals combined with

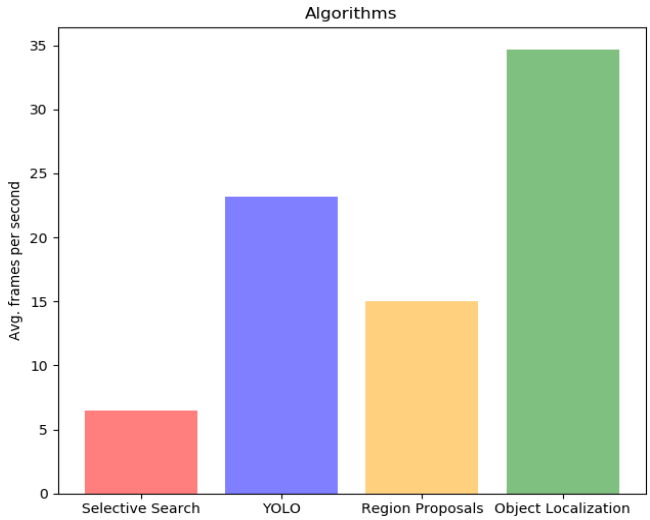


Fig. 3. Average frames per second for different detection algorithms.

CNN are good detection algorithms but involve heavy com- putation which will in turn lead to reduction in number of frames per second. In Fig. 3. We have plotted the average frames per second in a one minute window for detecting the puzzle (not solving). Object localization has outperformed the other algorithms as it has only one CNN which is the same for classifying and finding the bounding box as well [1]. Selective Search, YOLO, Region Proposals were proposed for detecting multiple objects in a frame [2, 3, 4], but in our problem we are only trying to detect one object. So, there is no need of extra computation which is required to detect multiple objects. We will further describe the architecture and the training methodologies used for Object Localization as we notice it performing well over other algorithms in our empirical data for this problem[16, 17, 20].

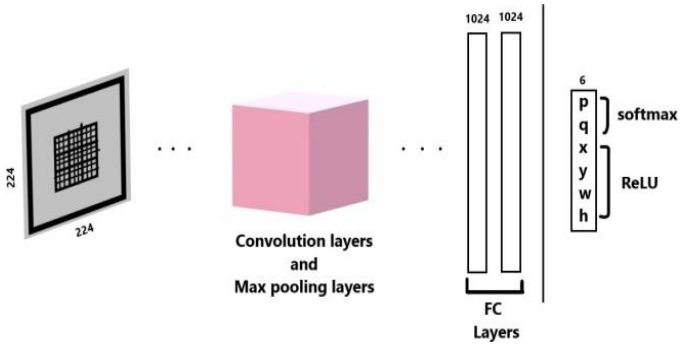


Fig. 4. Architecture of CNN used for Object Localization

First the frame is resized to 224 x 224 pixels.

Note that a pixel represents a real value $x_{ij}$where x_{ij} s [0, 1]

As the image is converted to grayscale and the values between [0, 255] are normalized to real values between [0, 1]. This resized frame is passed as input to a CNN with following architecture.

- Convolution Layer: Input - a tensor of dimensions (224 x 224 x 1); number of filters - 96; kernel dimensions - (11 x 11 x 1); Padding - 0; stride - 4;
- Max Pooling Layer: Input - a tensor of dimensions (54 x 54 x 96); Pool size - (2 x 2 x 1); strides - (2 x 2 x 1);
- Convolution Layer: Input - a tensor of dimensions (27 x 27 x 96); number of filters-256; kernel dimensions - (5 x 5 x 96); Padding - 0; stride - 1;
- Max Pooling Layer: Input - a tensor of dimensions (23 x 23 x 256); Pool size - (3 x 3 x 1); strides - (2 x 2 x 1);
- Convolution Layer: Input - a tensor of dimensions (11 x 11 x 256); number of filters - 384; kernel dimensions - (3 x 3 x 256); Padding - 0; stride - 1;
- Convolution Layer: Input - a tensor of dimensions (9 x 9 x 384); number of filters - 384; kernel dimensions - (3 x 3 x 384); Padding - 0; stride - 1;
- Convolution Layer: Input - a tensor of dimensions (7 x 7 x 384); number of filters - 256; kernel dimensions - (3 x 3 x 384); Padding - 0; stride - 1;
- Max Pooling Layer: Input - a tensor of dimensions (5 x 5 x 256); Pool size - (2 x 2 x 1); strides - (2 x 2 x 1);
- Fully Connected Layer: Input - a vector of length 1024; output neurons - 1024;
- Fully Connected Layer: Input - a vector of length 1024; output neurons - 1024;
- Output Layer: input - a vector of length 1024; softmax neurons - 2; ReLU neurons - 4; output neurons - 6;

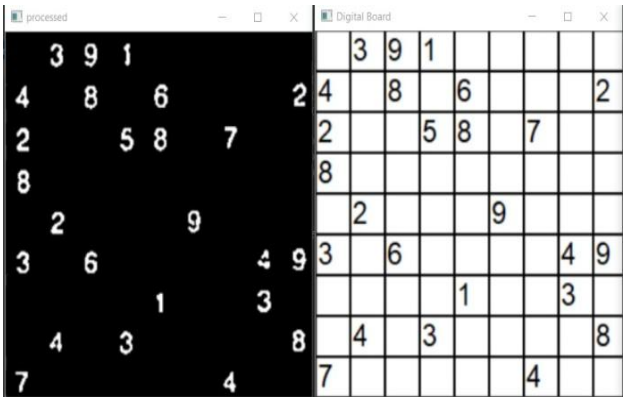
The architecture is similar to AlexNet [5]. However, there are some major changes in the fully connected layers and output layer. AlexNet contains 4096 neurons in fully connected layers, whereas, we used 1024 neurons in the fully connected layers and by doing so there is a total reduction of 18 million parameters, which is a great performance boost with respect to computation. And also AlexNet was designed for 1000 class classification problem but we are concerned with only a single class which is the Sudoku Puzzle along with four other regression outputs. We noticed an accuracy of 98.3 on the test data when using 4096 neurons in the fully connected layers and an accuracy of 98.2 with 1024 neurons in the output layer. The reason for having such a less difference in the test data accuracy could be overfitting . As the number of parameters increase there is a chance that the model is overfitting [7], however, weight decay (L2) regularization was used in both cases. The difference in the accuracies is very less but the difference in the computation are extremely large so we decided to use 1024 neurons in the fully connected layers. Speaking of output layer, it outputs 6 values. p, q, x, y, w, h. Where p is the probability that the Sudoku Puzzle is in the frame and q is the probability that the sudoku puzzle is not in the frame. So, in an ideal case p should be 1 and q should be 0 if the puzzle is in the frame, if not p should be 0 and q should be 1. The values x, y, w, h represent the bounding box. Where (x, y) are coordinates of the top left corner of the bounding box, values w and h represent width and height of it[18, 19].

For all the convolution layers and fully connected layers ReLU is used as the activation function. In the output layer, values p and q have Softmax activation and values x, y, w, h have ReLU activation. This is because for values p

and q we expect a probability where the value of p + q is always equal to 1, it can be achieved using Softmax activation over p and q Values x, y, w, h are always non negative integers. ReLU always outputs a non negative real number, so ReLU is a good activation function for predicting these values.

Some other hyperparameters used while training:

- He initialization [8] was used for all the kernels.
- Weight Decay (L2 regularization) with lambda value 1e- 5.
- Adam [9] was used as the optimizer with a learning rate of 0.001.
- Batch Normalization after every convolution layer.



B. Digit recognition on each cell

Fig. 5. (left) Processed frame, (right) classified cells.

Now that we have the bounding box of the Sudoku Puzzle, we cropped the puzzle from the original frame and resized it to 252 x 252 image. Before we went further some image processing had to be performed. Which is removing the margins of the of the cells, as the margins of the cells are straight lines we used Hough Line detection to detect the margins and remove them. By doing so the frame will be similar to left image in Fig. 5.

There are a total of 9 x 9 cells in Sudoku, as we are solving 9 x 9 standard size Sudoku puzzles. So after resizing the image each cell will be of 28 x 28 pixels. These 28 x 28 pixels will be given as an input to a CNN which will classify if the cell is empty or which of the 9 digits (sudoku doesn't contain 0) does the cell contain. The classified values are stored for further solving the problem.

The CNN used for this subproblem has the following structure.

- Convolution Layer: Input - a tensor of dimensions (28 x 28 x 1); number of filters - 64; kernel dimensions - (3 x 3 x 1); Padding - 1; stride - 1;
- Max Pooling Layer: Input - a tensor of dimensions (28 x 28 x 64); Pool size - (2 x 2 x 1); strides - (2 x 2 x 1);
- Convolution Layer: Input - a tensor of dimensions (14 x 14 x 64); number of filters-128; kernel dimensions - (3 x 3 x 64); Padding - 1; stride - 1;
- Max Pooling Layer: Input - a tensor of dimensions (14 x 14 x 128); Pool size - (2 x 2 x 1); strides - (2 x 2 x 1);
- Convolution Layer: Input - a tensor of dimensions (7 x 7 x 128); number of filters - 256; kernel dimensions - (3 x 3 x 128); Padding - 1; stride - 1;
- Convolution Layer: Input - a tensor of dimensions (7 x 7 x 256); number of filters - 256; kernel dimensions - (3 x 3 x 256); Padding - 1; stride - 1;
- Max Pooling Layer: Input - a tensor of dimensions (7 x 7 x 256); Pool size - (2 x 2 x 1); strides - (2 x 2 x 1);
- Fully Connected Layer: Input - a vector of length 4096; output neurons - 1024;
- Fully Connected Layer: Input - a vector of length 1024; output neurons - 1024;
- Softmax Layer: input - a vector of length 1024; output neurons - 10;

Training strategies used:

- ReLU activation for all the layers except pooling and Softmax Layer.
- Dropout with rate=0.5.
- Batch normalization after last fully connected layer.
- He initialization for all the kernels.
- Adam optimizer with learning rate 0.001.

C. Solving the puzzle

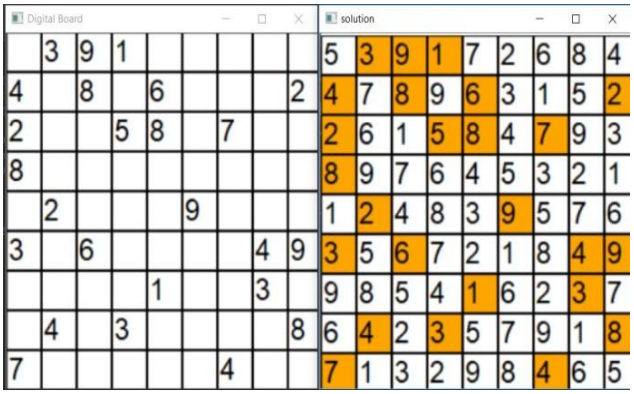


Fig. 6. (left) classified cells, (right) solved puzzle

By the time we reach this step, the complete Sudoku puzzle is detected and values in each cell are loaded into the memory. Many solutions have been proposed to solve Sudoku the simplest and the most popular one being the backtracking. In backtracking all the possible values are tested in each empty cell until there are no other possibilities left, if the solution is not yet reached then the next possibility of the previous empty cell is checked.

Another algorithm for solving Sudoku is Constraint programming. This method was proposed by Helmut Simonis where he proposed solving Sudoku as a constraint problem [9]. However, the algorithm might not work for some difficult problems so backtracking has to be used in such cases. So a combination of both algorithms will result in a faster solution. An implementation of such algorithm is provided by Rhollor [10]. Which is the same algorithm we used for solving this problem.

D. Drawing the solution onto the original frame

At this point we have the solution for the puzzle and inorder to fill the solution in the empty cells and give an AR experience. The x, y, w, h values obtained during Object

Localization will be used again. If a digit belonging to cell cij is to be drawn onto the frame, then the pixel on which we need to start drawing is

$(x + ((x + w)/9) * i + 1, y + ((y + h)/9) * j + 1)$

The same is done for all the empty cells and finally our goal is achieved. By the end of all the four steps the frame will look as the output in Fig. 7.

E. Training Data

The data set used for training CNN for object localization was created using 200 images of Sudoku Puzzle from news-papers and magazines and 1000 images which do not contain the Puzzle. For each image 6 labels are attached p,q, x, y, w, h which will be used by loss function during the training process.

Generating this data was very expensive. We have written a program which will provide the x, y, w, h values when a box is drawn around the puzzle using the mouse. The same had to be done for every image. And also we wanted to use the data for training, so we opted for Data augmentation with the following techniques.

- Rotation - we rotated the y - axis between [-15, 15] degrees arbitari and generated 100 more images.
- Translation - By translation of x and y axis another 400 images were generated.
- Gaussian noise - By adding Gaussian noise with a stan- dard deviation 1 another 100 images were added.

The CNN used for classification of digits is trained using MNIST dataset, as it does not contain empty cells all the zero labeled images are made as blank images. Because the digit '0' is not used in Sudoku but in there are very high chances that the cell could be empty. So these '0' digit images were replaced with blank images in the MNIST dataset for training. No data augmentation was performed here as the MNIST dataset is a very rich dataset [6] and we achieved 99.67% accuracy on the test data set.

IV. RESULTS

The CNN used for Object Localization has 99.33 accuracy in correctly classifying the test dataset and an average of 0.87 Intersection over Union (IoU) for finding the bounding box. Whereas, CNN used for classifying digits has an accuracy of 99.67 in correctly classifying the digits.

Overall the performance of our solution in terms of frames per second is as following. For an easy level of difficulty it was approximately 26 frames per second, for intermediate level of difficulty it was 20-22 frames per second and for hard level of difficulty 13-16 frames per second were noticed.

Also an analysis of each step involved in solving the problem was provided along with exploring the other possible solutions and discussing their drawbacks.

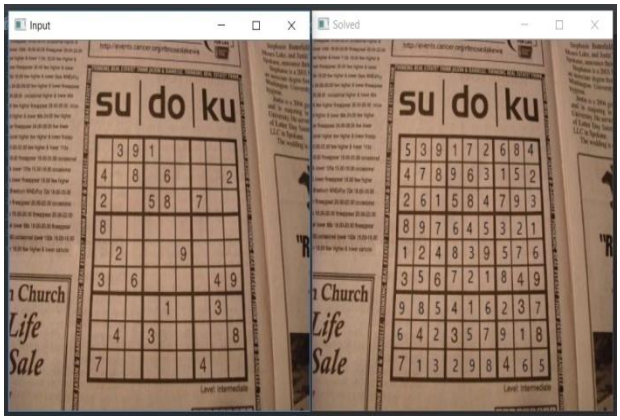


Fig. 7. (left) original frame, (right) solved puzzle as Augmented Reality

V. CONCLUSION

A full fledged system for showing the solution of a Sudoku puzzle as Augmented Reality was proposed in the paper. Using image processing, object localization, image classification, constraint programming and backtracking. Fig. 7. shows how the solution is shown on top of the original frame by the end of all the steps.

We also addressed some substantial improvements to the previous methodologies used for the problem set. Such as, using Object Localization for puzzle detection and the Con-volutional Neural Network to be used for implementing it, classifying digits from the puzzle, solving Sudoku and finally filling the solution in the empty cells precisely.

VI. COMPLIANCE WITH ETHICAL STANDARDS

Both the authors declares that he/she has no conflict of interest and have written this after reviewing various research papers and then given a easy solution to solve the Puzzles.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

VII. REFERENCES

1. Andrew Ng. Lecture on "Object Localization" in the course "Convolutional Neural Networks". [Online]. Avail-able:https://www.youtube.com/watch?v=GSwYGkTfOKk.
2. J. R. R. Uijlings, K. E. A. van de Sande,T. Gevers, A. W. M. Smeul- ders."Selective Search for Object Recognition" in international journal of Computer Vision, September 2013, Volume 104, Issue 2, pp 154–171.
3. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection" in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.
4. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" in Advances in Neural Information Processing Systems 28 (NIPS 2015).

5. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks" in Advances in neural information processing systems 25(2). [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep- convolutional-neural-networks.pdf.
6. The MNIST Dataset. By Yann LeCun, Corinna Cortes and Christopher J.C. Burges. [Online]. Available: http://yann.lecun.com/exdb/mnist/.
7. Michael Nielsen, in his book Neural Networks and Deep learning -Chapter 3 - Overfitting and Regularization
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. in "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification" The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034.
9. Helmut Simonis. "Sudoku as constraint problem" paper presented at the Eleventh International Conference on Principles and Practice of Constraint Programming.
10. Rhollor. "Sudokusolver". GitHub. Rhollor. Retrieved 8 December 2016.
11. Andrew Ng. Lecture on "Object Detection" in the course "Convolutional Neural Networks". [Online]. Available: https://www.youtube.com/watch?v=5e5pjeojznk
12. Pramod J Simha, Suraj K v, and Tejas Ahobala, "Recognition of numbers and position using image processing techniques for solving Sudoku Puzzles" in IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM - 2012)
13. Snigdha Kamal, Simarpreet Singh Chawla, Nidhi Goel proposed, "De- tection of Sudoku Puzzle using Image Processing and Solving by Back- tracking, Simulated Annealing and Genetic Algorithms: A Comparative Analysis" in 2015 Third International Conference on Image Information Processing (ICIIP).
14. A. Van Horn, "Extraction of sudoku puzzles using the hough trans- form", University of Kansas, Department of Electrical Engineering and Computer Science, Tech. Rep., 2012.
15. Baptiste Wicht and Jean Hennebert, "Mixed handwritten and printed digit recognition in Sudoku with Convolutional Deep Belief Network" in 2015 13th International Conference on Document Analysis and Recognition (ICDAR).
16. Merugu S., Reddy M.C.S., Goyal E., Piplani L. (2019) Text Message Classification Using Supervised Machine Learning Algorithms. In: Kumar A., Mozar S. (eds) ICCCE 2018. Lecture Notes in Electrical Engineering, ISSN 1876-1100, Vol. 500. Springer, Singapore.
17. Yadav B.C., Merugu S., Jain K. (2019) Error Assessment of Fundamental Matrix Parameters. In: Kumar A., Mozar S. (eds) ICCCE 2018. Lecture Notes in Electrical Engineering, ISSN 1876-1100, Vol. 500. Springer, Singapore.
18. Sharma S.K., Jain K., Suresh M. (2019) Quantitative Evaluation of Panorama Softwares. In: Kumar A., Mozar S. (eds) ICCCE 2018. Lecture Notes in Electrical Engineering ISSN 1876-1100, Vol. 500. Springer, Singapore.
19. Suresh Merugu and Kamal Jain, "Colorimetry-based edge preservation approach for color image enhancement," Journal of Applied Remote Sensing, (SPIE)
20. Suresh Merugu, Kamal Jain, 2015, "Semantic Driven Automated Image Processing using the Concept of Colorimetry", Second International Symposium on Computer Vision and the Internet (VisionNet'15), Procedia Computer Science 58 (2015) 453 – 460

VIII. APPENDIX

This Paper was implemented by Azhar Talha Syed under the supervision and guidance of Dr. Suresh Merugu. The implementation of the project was done in Python using Tensorflow and OpenCV libraries.