An Efficient Machine Learning Technique to Classify and Recognize Handwritten and Printed Digits of Sudoku Puzzle

Sang C. Suh[*1], Aghalya Dharshni Manmatharaj[2]
Department of Computer Science
Texas A&M University-Commerce
Commerce, Texas

***Abstract*—In this paper, we propose a convolutional neural network model to recognize and classify handwritten and printed digits present in Sudoku puzzle, which is captured using smartphone camera from various magazines, and printed papers. Sudoku puzzle grid is detected using various image processing and filtering techniques such as adaptive threshold. The system described in the paper is thoroughly tested on a set of 100 Sudoku images captured with smartphone cameras under varying conditions. The system shows promising results with 98% accuracy. Our model can handle more complex conditions often present on images that were taken with phone cameras and the complexity of mixed printed and handwritten digits.**

Keywords*—Convolutional Neural Network (CNN); Artificial Neural Network (ANN); Deep Belief Network (DBN); Optical character recognition (OCR), Open source computer vision (OpenCV); Convolutional Deep Belief Network (CDBN)

I. Introduction

Profound Neural Network display arrangements have demonstrated exceptionally fruitful on digit acknowledgment. We thought of different explorations that tended to be the issues of perceiving Sudoku digits from paper pictures taken with an advanced camera, recognizing computerized numbers present in the Sudoku and distinguishing manually written numbers. The approach that we are attempting to address in this paper is whether such immense neural framework structures can manage the interpretation of both handwritten and computerized digits, present in Sudoku picture without separating them and classify into two characterizations.

The Sudoku baffle is a Japanese diversion. It is a rational, a number-based riddle. This paper centers around the standard number Sudoku played on a 9 x 9 network. Each cell should contain a digit between 0 and 10. The diversion starts with a halfway filled network and the objective is to fill each line, segment, and sub 3 x 3 square with numbers, so each number is available once [1][2][3][4][5]. In this work, we center around filled Sudoku, containing both transcribed and printed digits [6]. We employ different picture preparing procedures to acquire a commotion free Sudoku picture and use MNIST digits dataset to discover the digit's present in the picture [1][5][13]. The dataset is available and accessible online for everybody as a free of expenses. Fig. 1 shows one of the input images that we used as input.

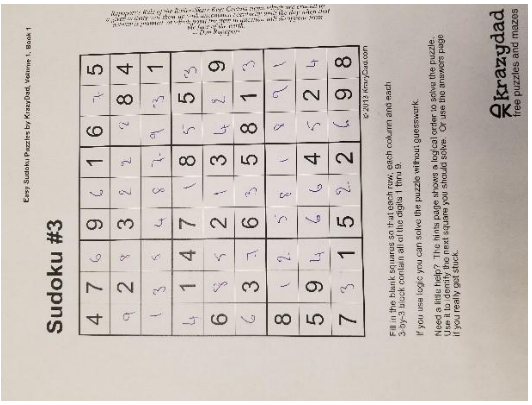

Sudoku #3
Easy Sudoku Puzzles by KrazyDad, Volume 1, Book 1
Fill in the blank squares so that each row, each column and each 3-by-3 block contain all of the digits 1 thru 9.
If you use logic you can solve the puzzle without guesswork.
Need a little help? The hints page shows a logical order to solve the puzzle. Use it to identify the next square you should solve. Or use the answers page if you really get stuck.
© 2013 KrazyDad.com
krazydad
free puzzles and mazes

Fig. 1. Image of a Sudoku Puzzle Planned to be used.

The remainder of this paper is organized as follows. Section II gives the background of the problem and describes previous work achieved related to our research reported in this paper. Section III presents the methodology used to validate the proposed solution followed by the design of the project briefly described in Section IV. Section V describes the methods used and provides results of the system. Section VI presents the significance and scope of the study. Finally, the paper ends with conclusion with some ideas and suggestion for future work related to the problem.

II. Literature Review

A. Sudoku

Sudoku is a number game played on 9*9 grid. The objective is to fill the blank squares with digits 1 to 9. When Sudoku is completed, each row and column must contain all digits from 1 to 9 exactly once [1][2][3][4][5]. Numbers from 1 to 9 must be affixed within each grid of 3*3, no row and column can contain repeat of the instance [6][7][28]. In real life, we come across many Sudoku puzzles of varying difficulty levels in newspapers and other books.

B. Sudoku Image Recognition

In the year 2012, A. Van Horn proposed a framework to perceive and fathom Sudoku puzzles dependent on the Hough change. Four corners of the lattice are identified dependent on the crossing point of the recognized lines. Van Horn proposed a framework to perceive and fathom a Sudoku confuse

[*] Corresponding Author

dependent on the Hough change. Four corners of the network are identified dependent on the crossing point of the distinguished lines. Digits are centered in their cells and passed to an ANN [29][30]. The framework starts with a little arrangement of pictures. Versatile thresholding is connected, and segments associated with the fringes are expelled to lessen clamor and improve later advances. Utilizing another calculation, the biggest segment territory is distinguished as a network. At last, the digits are grouped according to straightforward format machine methodology. None of these techniques have been completely tried on characterized informational collections and no examination has been done on ordering Sudoku with combined printed and handwritten digits.

C. Camera-based OCR

Content identification and recognition in pictures picked up from the scanners have been concentrated eventually with powerful courses of action proposed. Another hand, camera-based PC vision issues stay trying for a few reasons. Center in such gadgets are seldom flawless and optical zoom is of low quality. Pictures are taken under different conditions and different lighting conditions either normal or fake which presents shadows and slope of brightening [19]. The text from the Sudoku images, which have been taken from the scanner, has the perfect images than the distorted captured images from the camera. While thinking about the paper, a few different wellsprings of fluctuation contemplated the pictures from the paper results in twisted pictures. Font sizes and styles in a newspaper can also differ. The standard steps of image processing (normalization, text localization, enhancement, and binarization) are analyzed and different solutions are compared.

D. Image Processing Techniques

By and large, picture preparation goes through such phases as picture import, examination, control and picture yield. Advanced (digital) and simple (analog) are the two strategies used for picture preparation [17]. Techniques which have been applied for image processing are image editing, image restoration, independent component analysis, anisotropic diffusion, linear filtering, and neural networks. In our research, the neural networks approach was investigated and used. The neural network techniques which we particularly focused on are Contour detection and Hough transform.

E. Hough Transform

It is a technique which can be used to isolate features of the shape of an image because it requires the desired features to be specified in the parametric form. Generalized Hough transform can be employed in applications where an analytic description of features is impossible. The Hough method is especially helpful for figuring a worldwide portrayal of features where the quantity of arrangement classes need not be known from the earlier given (potentially uproarious) nearby estimations [14][20][22][26].

The motivating idea behind the Hough technique for line detection is that each input measurement (e.g., coordinate point) indicates its contribution to a globally consistent solution such as the physical line which gives rise to an image point [1][5][8].

F. Deep Belief Network (DBN)

A DBN is an unsupervised deep learning algorithm. It is composed of a multilayer of latent variables. Latent variables are binary, also called as hidden units. It is considered a hybrid graphical model. Top two layers are undirected. Lower layers have directed connections, with the arrows pointed towards the layer that is closest to data. Lower layers have acyclic connections that convert associate memory to observed variables. There are no intralayer connections. DBN is pre-trained using Greedy learning algorithm. This algorithm is fast and learns one layer at a time. In the year 2009, Lee et al. presented a new type of DBN, a CDBN [1]. This arrangement permits to scale the system to bigger picture sizes, allowing a CDBN to have the capacity to group full-sized normal pictures. They exhibited superb execution on visual acknowledgment tasks. Additionally, their network accomplished cutting edge on the MNIST dataset. Since then, Deep Belief Networks and Deep Architectures, in general, have been used in several domains such as face recognition, reinforcement learning, and handwritten characters recognition. They have demonstrated effective, frequently accomplishing cutting edge results by selecting a pattern in image.

G. Digit Recognition

Digit acknowledgment utilizing OCR is done by identifying the length and broadness of the square matrix and isolating it into 9 a balance of yielding 81 littler squares. The corner to corner x and y co-ordinates of every little square are figured and re-allocated values from 1 to 9, determining the line and section number of each square. Therefore, the centroid estimations of every digit present in the riddle are resolved and these facilitate qualities are re-doled out qualities from 1 to 9 by over and again looking at the x and y directions of the centroid to the directions of each littler square. On the off chance that the centroid exists in the x and y directions of a square, it takes the estimation of the line and section number of that square. At that point the numbers present in the picture are perceived and arranged utilizing OCR [10][11]. The OCR yields exceedingly precise outcomes under the condition that the clamor present around the characters in the picture is insignificant. In this way, it is significant that the picture handling stages for commotion evacuation yield sharp and clear pictures for OCR location as appeared in Fig. 4. The recognized digits are put away in a 9x9 grid dependent on their new centroid esteems and empty spots are relegated zeros in the lattice.

III. Methodology

We will be following the Agile Environment over our project. Agile is a theoretical structure for undertaking programming structure projects. Agile methods attempt to minimize risk by developing software in short time boxes, called iterations, which typically last one to four weeks. Every cycle resembles a smaller than usual programming venture of its own and incorporates every one of the errands important to discharge the scaled down addition of new usefulness: arranging, necessities investigation, plan, coding, testing, and documentation. While iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. At the end of each iteration, the team reevaluates project priorities. Agile techniques underline continuous correspondence, ideally up close and personal, and overwritten archives. Most light-footed groups are situated in a warmup area and incorporate every one of the general population important to complete the product. At the very least, this incorporates developers and the general population who characterize the item such as item chiefs, business examiners, or real clients. The warmup area may likewise incorporate analyzers, interface planners, specialized scholars, and the board. Light-footed techniques additionally underscore working programming as the essential proportion of advancement. Joined with the inclination for eye to eye correspondence, coordinated techniques produce next to no composed documentation in respect to different strategies.

The reason for selecting this environmental model is because there are three unique phases. Each phase will provide an individual set of output.

IV. Design and Approach

A. Requirements and Planning

We used Python of version 3 and various libraries such as TensorFlow [12], Matplotlib, Keras, Convolutional [16], OpenCV 2, Numpy, Pylab. We used MNIST and Char74K dataset for training.

B. Design

Fig. 2 describes the design of the project.

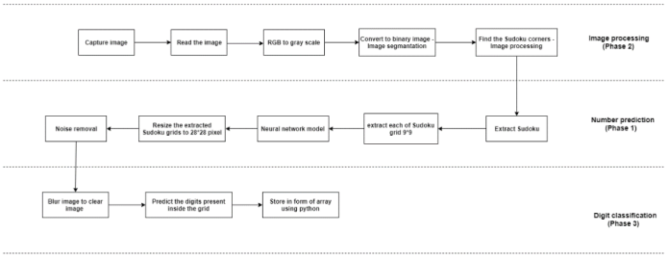

Capture image
Read the image
RGB to gray scale
Convert to binary image - Image segmantation
Find the Sudoku corners - image processing
Image processing (Phase 2)
Extract Sudoku
extract each of Sudoku grid 9*9
Neural network model
Resize the extracted Sudoku grids to 28*28 pixel
Noise removal
Number prediction (Phase 1)
Blur image to clear image
Predict the digits present inside the grid
Store in form of array using python
Digit classification (Phase 3)

Fig. 2. Design of Mixed Handwritten and Printed Digit Recognition and Classification in Sudoku With Convolutional Deep Belief Network.

V. Development and Results

A. Phase 1: Develop Handwritten Number Prediction Model

A Neural Network model is trained to predict numbers using MNIST dataset [13]. The training dataset is structured as a 3-dimensional array of the instance, image width, and image height. For a multi-layer perceptron model, we must reduce the images down into a vector of pixels. For this situation, the 28×28 estimated pictures will be 784-pixel input esteems. We can do this change effectively utilizing the reshape work on the NumPy exhibit. We can also reduce our memory requirements by forcing the precision of the pixel values to be 32 bits, the default precision used by Keras anyway [27]. The pixel values are grayscale between 0 and 255. It is quite often a smart thought to play out some scaling of info which esteems when neural system models can be utilized. Since the scale is notable, we can rapidly standardize the pixel esteems to the range 0 and 1 by separating each incentive by the limit of 255. Fig. 3 shows the grayscale version of the image present in the MNIST database.

The yield variable is a whole number from 0 to 9. This is a multi-class classification problem. As such, it is good practice to use one hot encoding of the class values, transforming the vector of class integers into a binary matrix. We can undoubtedly do this by utilizing the inherent np_utils.to_categorical() method to support the work in Keras. We are presently prepared to make our straightforward neural system model. We will characterize our model in a capacity to training and testing. This is convenient in the event that you need to expand the precedent later and attempt and show signs of improvement score. The model is a basic neural system with one concealed layer which contains indistinguishable number of neurons. There are 784 inputs neurons present in the hidden layer. A rectifier actuation work is utilized for the neurons in the concealed layer.

A softmax activation function is used on the output layer to turn the outputs into probability-like values and allow one class of the 10 to be selected as the model's output prediction [15]. The logarithmic loss is used as the loss function called categorical_crossentropy in Keras and the efficient ADAM gradient descent algorithm is used to learn the weights [1][2][5]. We can now fit and evaluate the model. The model fits more than 60,000 training images in 10 epocs with 99 percent accuracy. The test data is used as the validation dataset, allowing you to see the skill of the model as it trains. A verbose value of 2 is used to reduce the output to one line for each training epoch. Fig. 4 shows the output of the epochs which finally gives a training accuracy of 99%.

The overall model follows Convolutional Neural Network. The architecture of our model is shown in Fig. 5.

Finally, the test dataset is used to evaluate the model and a classification rate is printed as shown in Fig. 6.

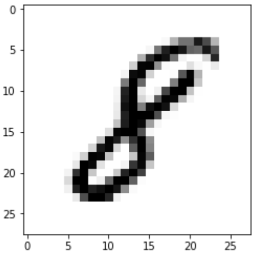
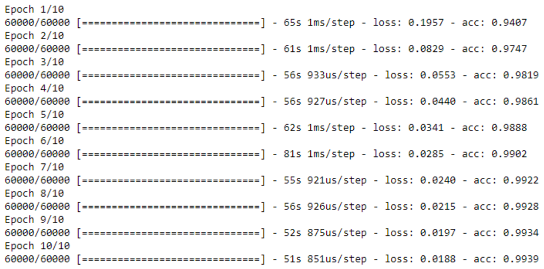
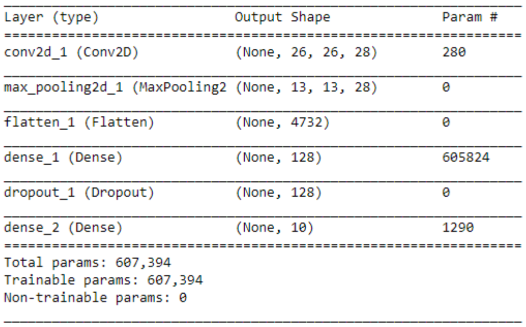


Fig. 3. The Grayscale Version of the Image Present in the MNIST Database.



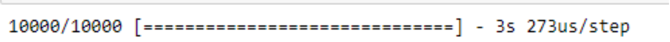
```
Epoch 1/10
60000/60000 [==============================] - 65s 1ms/step - loss: 0.1957 - acc: 0.9407
Epoch 2/10
60000/60000 [==============================] - 61s 1ms/step - loss: 0.0829 - acc: 0.9747
Epoch 3/10
60000/60000 [==============================] - 56s 933us/step - loss: 0.0553 - acc: 0.9819
Epoch 4/10
60000/60000 [==============================] - 56s 927us/step - loss: 0.0440 - acc: 0.9861
Epoch 5/10
60000/60000 [==============================] - 62s 1ms/step - loss: 0.0341 - acc: 0.9888
Epoch 6/10
60000/60000 [==============================] - 81s 1ms/step - loss: 0.0285 - acc: 0.9902
Epoch 7/10
60000/60000 [==============================] - 55s 921us/step - loss: 0.0240 - acc: 0.9922
Epoch 8/10
60000/60000 [==============================] - 56s 926us/step - loss: 0.0215 - acc: 0.9928
Epoch 9/10
60000/60000 [==============================] - 52s 875us/step - loss: 0.0197 - acc: 0.9934
Epoch 10/10
60000/60000 [==============================] - 51s 851us/step - loss: 0.0188 - acc: 0.9939
```

Fig. 4. Training Epochs.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 28)        280
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 28)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4732)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               605824
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 607,394
Trainable params: 607,394
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. CNN Structure of our Model.



```
10000/10000 [==============================] - 3s 273us/step
```

Fig. 6. Test Result.

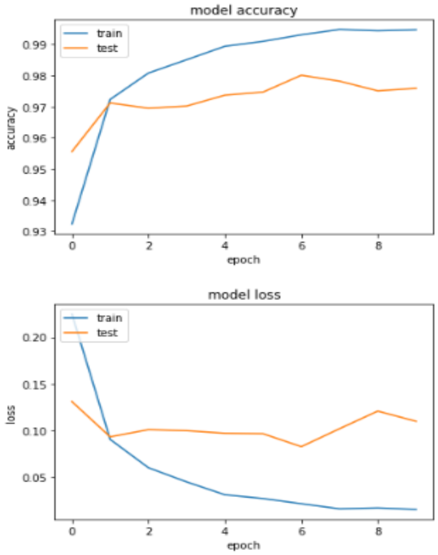


Fig. 7. Model Accuracy and Model Loss.

Model accuracy and model loss is shown in Fig. 7.

B. Phase 2: Sudoku Grid Detection and Extraction

We imported the camera captured Sudoku image into the model using Open CV which is one of the Open Source Computer Vision Library [25][9]. Red Green Blue scale image is converted into Grayscale image. Unwanted image background lighting is removed using the adaptive threshold method. The adaptive threshold module is used in uneven lighting conditions when you need to segment a lighter foreground object from its background. In many lighting situations, shadows or dimming of light cause thresholding problems as traditional thresholding considers the entire image brightness. Adaptive thresholding will perform binary thresholding (i.e. it creates a black and white image) by analyzing each pixel with respect to its local neighborhood. This localization allows each pixel to be considered in a more adaptive environment. The algorithm considers each pixel one at a time, calculate the mean of the local neighborhood 'window size' (x-windowSize/2, y-windowSize/2, x+windowSize/2, y+windowSize/2) and thresholds the current pixel to white if the difference between the calculated mean and the current pixel value is lower than the 'mean offset' [18][23]. Fig. 8 shows the pre-processed image.

Each 9*9 grid is separated using the image slicer technique as shown in Fig. 9. It is the one that slices an image into titles and rejoins them. A maximum number of tiles that can be produced is 9800. This can be an arbitrary limit which ensures that row and column number can be conveniently represented by two digits.

In the same manner, the remaining 3*3 grids which have been separated from the 9*9 grid are handled and separated into the single grids.

C. Phase 3: Sudoku Grid Detection and Extraction

In this phase each separated grid can be manually differentiated what are the handwritten and the printed digits by denoting the handwritten digits with 2, printed digits with 1 and the left part, the empty grid, can be taken as 0 and this has been done only once so that it is a onetime process. At the classification process we used the saved database to classify the digits for the other Sudoku pictures [24]. The result what we have considered for this project is classified as shown in Fig. 10.

The classified Sudoku image is to be stored in the form of an array as shown in Fig. 11.

We can get Sudoku logic as well if we pass Sudoku logic to it.

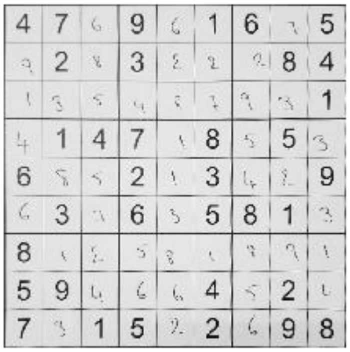


Fig. 8. Extracted Sudoku Grid.

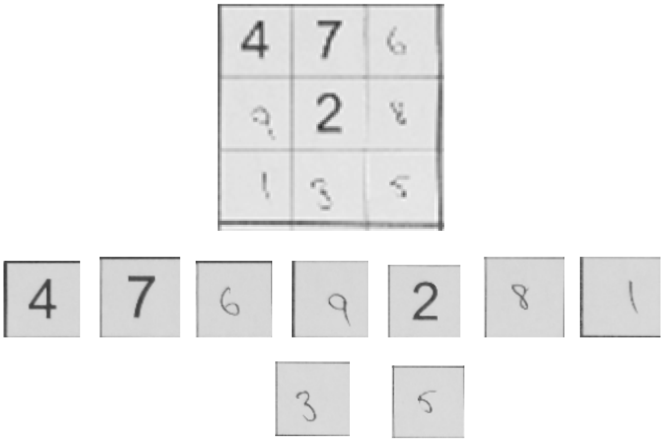
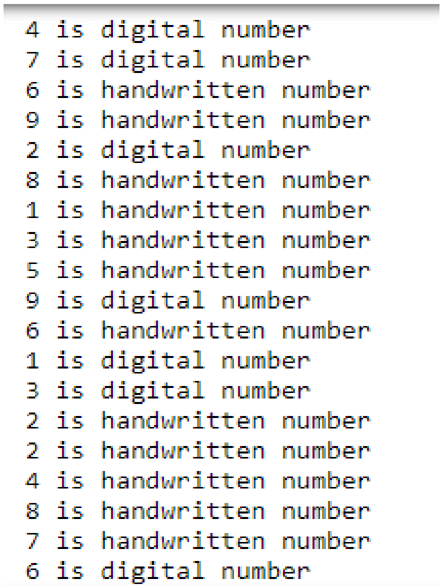
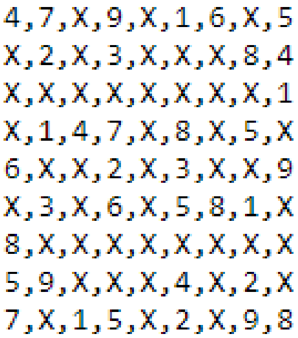


Fig. 9. Sliced Grids of Sudoku Image.



```
4 is digital number
7 is digital number
6 is handwritten number
9 is handwritten number
2 is digital number
8 is handwritten number
1 is handwritten number
3 is handwritten number
5 is handwritten number
9 is digital number
6 is handwritten number
1 is digital number
3 is digital number
2 is handwritten number
2 is handwritten number
4 is handwritten number
8 is handwritten number
7 is handwritten number
6 is digital number
```

Fig. 10. Recognized and Classified Handwritten and Printed Digits.



```
4,7,X,9,X,1,6,X,5
X,2,X,3,X,X,X,8,4
X,X,X,X,X,X,X,X,1
X,1,4,7,X,8,X,5,X
6,X,X,2,X,3,X,X,9
X,3,X,6,X,5,8,1,X
8,X,X,X,X,X,X,X,X
5,9,X,X,X,4,X,2,X
7,X,1,5,X,2,X,9,8
```

Fig. 11. Printed Digits are Stored in the Array Format.

VI. Significance and Scope of Study

The proposed model can be used for reducing manual work in education and merchant sectors. It may also play a crucial role in number recreations.

Nowadays, when the tip calculation is done manually, that may sometimes end with miscalculation. To avoid the miscalculation, the receipt can be scanned so that our model can predict the tip for automatically accurate total amount calculation. This may help with the reduction of profit loss for merchants.

In few education sectors, feedback and multiple-choice questions are still given in old paper writing format. To avoid manual correction and ranking, our model can be used for enhanced accuracy.

VII. Conclusion and Future Work

A model to recognize and classify both printed and handwritten digits was created using python version 3 in jupyter notebook. The process was split into three phases and the agile methodology was followed for the project to create a model. The CNN model and various image processing techniques were used to achieve 98% of accuracy.

The model created to recognize and classify both the handwritten digits and the printed digits separately was implemented by using various machine learning techniques and image processing techniques [21]. The significance of the work is that since the model can recognize the handwritten letters as well with the printed letters, it will be possible that both handwritten and printed sentences can be identified and classified as well.

Acknowledgment

The authors would like to express special thanks to the Society for Design and Process Science for their support for this research.

Reference

[1] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in Proceedings of the 26th Annual International Conference on Machine Learning, ser. ICML ’09. New York, NY, USA: ACM, 2009.

[2] B. Wicht and J. Hennebert, “Camera-based sudoku recognition with deep belief network,” in Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of, Aug 2014.

[3] Bishop, M. Christopher. “Pattern Recognition and Machine Learning,” 2006 Springer.

[4] Berthier, Denis. “The Hidden Logic of Sudoku,” 2007 International conference.

[5] Boyer, Christian. "Supplément de l'article "Les ancêtres français du sudoku"," on December 10, 2006. Retrieved August 3, 2009.

[6] Lewis, R (2007) “Metaheuristics Can Solve Sudoku Puzzles” Journal of Heuristics, vol. 13 (4), pp 387-401.

[7] Perez, Meir, and Marwala, Tshilidzi (2008) “Stochastic Optimization Approaches for Solving Sudoku,”.

[8] http://intelligence.worldofcomputing/brute-force-search Brute Force Search, December 14th, 2009.

[9] P. Simha, K. Suraj, and T. Ahobala, “Recognition of numbers and position using image processing techniques for solving sudoku puzzles,” in Advances in Engineering, Science and Management (ICAESM), 2012.

[10] S. M. Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana, and Abdullah Al Jobair “Handwritten Digit Recognition using Machine Learning Algorithms,” in Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence Volume 18 Issue 1 Version 1.0 Year 2018 Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Online.

[11] Baptiste Wicht, Jean Henneberty “Mixed handwritten and printed digit recognition in Sudoku with Convolutional Deep Belief Network,” in International Conference of August 2015.

[12] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning", 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16).

[13] Y. Lecun, C. Cortes, C. J. C. Burges, The MNIST Database” Courant Institute NYU, 2014.

[14] V. Nair, G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", Proc. 27th Int. Conf. Mach. Learn.

[15] R. Salakhutdinov, G. Hinton, "Replicated softmax: An undirected topic model", Adv. Neural Inf. Process. Syst. 22-Proc. 2009 Conf.

[16] Y. LeCun, K. Kavukcuoglu, C. Farabet, "Convolutional networks and applications in vision", Circuits and Systems (ISCAS) Proceedings of 2010 IEEE International Symposium.

[17] E. Alpaydin. “An Introduction to Machine Learning”. The MIT press, Cambridge, Massachusetts, London, England, 2004.

[18] T.CoverandP.Hart. “Nearest neighbor pattern classification”. IEEE Transaction on Information Theory.

[19] R.Duda, P.Hart,and D.Stork. “PatternClassification”. Wiley Interscience, 2nd ed.

[20] S. Eyheramendy, D. Lewis, and D. Madigan. “On the naive bayes model for text categorization”. Proceedings Artificial Intelligence Statistics, 2003.

[21] T. Mitchell. “Machine Learning”. McGraw-Hill.

[22] I. Rish. “An empirical study of the naive bayes classifier”, 2001.

[23] N. Roussopoulos, S. Kelley, and F. Vincent. “Nearest neighbor queries”. International Conference on Management of Data, 1995.

[24] K. Weise and W. Woger. “Comparison of two measurement results using the bayesian theory of measurement uncertainty”. Measurement Science and Technology.

[25] Q. J. Wu. “Machine Learning and Computer Vision”. University of Windsor, Windsor, ON, Canada, 2007.

[26] R. Deriche, “Using Canny's criteria to derive a recursively implemented optimal edge detector”, Int. J. Computer Vision.

[27] M. A. Nielsen, “Neural Networks and Deep Learning”, Determination Press, 2015.

[28] M. Perez and M., Tshilidzi, “Stochastic Optimization Approaches for Solving Sudoku”, 2008.

[29] P. J. Simha, K. V. Suraj, and T. Ahobala, “Recognition of Numbers and Position using Image Processing Techniques for Solving Sudoku Puzzles”, 2012.

[30] B. Wicht and J. Hennebert, “Camera-based Sudoku recognition with Deep Belief Network”, 2014.