Camera-based Sudoku recognition with Deep Belief Network

Baptiste Wicht, Jean Hennebert
University of Fribourg
HES-SO, University of Applied Science
Fribourg, Switzerland
Email: baptiste.wicht@unifr.ch

***Abstract*—In this paper, we propose a method to detect and recognize a Sudoku puzzle on images taken from a mobile camera. The lines of the grid are detected with a Hough transform. The grid is then recomposed from the lines. The digits position are extracted from the grid and finally, each character is recognized using a Deep Belief Network (DBN). To test our implementation, we collected and made public a dataset of Sudoku images coming from cell phones. Our method proved successful on our dataset, achieving 87.5% of correct detection on the testing set. Only 0.37% of the cells were incorrectly guessed. The algorithm is capable of handling some alterations of the images, often present on phone-based images, such as distortion, perspective, shadows, illumination gradients or scaling. On average, our solution is able to produce a result from a Sudoku in less than 100ms.**

***Keywords*-Camera-based OCR; Deep Belief Network; Text Detection; Text Recognition;**

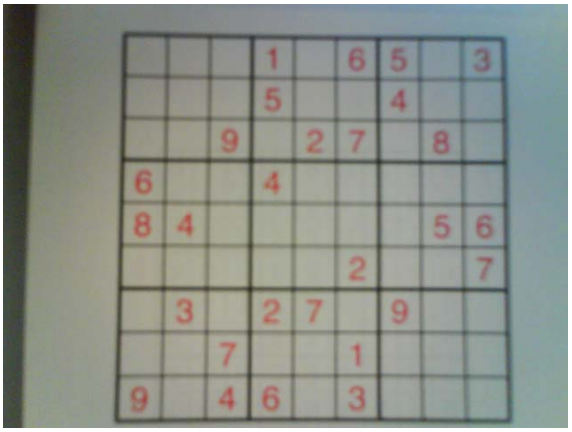

Fig. 1. Image of a Sudoku puzzle from our dataset

I. Introduction

Deep learning and more specifically deep belief networks have been used successfully on some scanner based digit recognition tasks [1]. An important advantage of such approaches is in the fact that few a priori knowledge is injected in the system. Typically, raw inputs (pixels) can be injected in the system which is able to learn features. The scientific question that we address in this paper is about the capacity of such deep learning procedures to handle more complex inputs, for example acquire from camera-based images that present more noise and distortion than scanner based images.

For this reason, we address in our work the specific problem of recognizing Sudoku puzzles from newspaper pictures coming from digital camera. With the ever-increasing number of smartphone, people have instant access to a digital camera. Thus, it is becoming more and more important to make use of these smartphone-based pictures. Moreover, as the phones are becoming very powerful, it becomes easier for the applications analyzing these pictures to directly run on them. Solving computer vision problems directly on the phone has the advantage that the user has directly access to the result.

A solution to solve this problem is presented and thoroughly tested. A Hough transform is used to detect the lines present in the images. From this information, the outer grid of the Sudoku is detected and then split in smaller cells. Once the characters are isolated, the digits are recognized using a Deep Belief Network (DBN). A dataset has also been collected during this project to assess the quality of the system. Although the proposed system has not been tested on a smartphone, everything has been designed to be able to adapt it as a smartphone application easily.

The rest of this paper is organized as follows. Section II covers the background of the problem. Section III analyzes the previous work achieved in different fields covered by this research. Section IV presents the dataset of Sudoku images that has been collected and that is used to evaluate the presented solution. Section V details the techniques used to detect the Sudoku grid and the digits inside it. Section VI describes how digits are recognized from the image. Section VII discusses the overall results of the system as well as its efficiency. Finally, Section VIII concludes this paper and presents some ideas for further improvements of the solution.

II. Background

A. Sudoku

The Sudoku puzzle is a famous Japanese game. It is a logic, number-based puzzle. This paper focuses on the standard Sudoku, played on a 9x9 grid. Each cell can either be empty or contain a digit from 1 to 9. The game begins with a partially filled grid and the goal is to fill every row, column and sub 3x3 square with the numbers, so that each number is present only once. Figure 1 shows a typical example from our dataset.

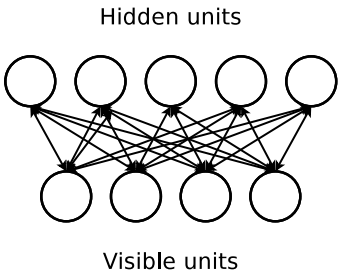

Fig. 2. A Restricted Boltzmann Machine

B. Camera-based OCR

Text detection and recognition in images acquired from scanners have been studied for a long time with very efficient solutions proposed [2]. On the other hand, camera-based computer vision problems remains challenging for several reasons:

- Camera are of various qualities and different camera may produce different pictures for the same scene.
- Focus is rarely perfect and zoom is often of poor quality.
- Light conditions may highly vary.
- The rotation of the image varies from one shot to another.

Pictures taken from newspaper have other caveats:

- Newspaper pages are never completely flat, resulting in curved images.
- Each newspaper uses different font styles and sizes.

C. Deep Belief Network

Deep Belief Network (DBN) were introduced by G. Hinton and R. Salakhutdinov in 2006 [3]. It is a novel way of training deep neural network efficiently and effectively. A DBN is a deep neural network composed of several layers of hidden units. There are connections between the layers, but not between units of the same layer. The hidden units typically have binary values, but extensions to a system with different types of units have already been experimented.

DBNs are typically implemented as a composition of simple networks, such as a Restricted Boltzmann Machine (RBM). Figure 2 shows an example of an RBM. Using RBMs for each layer leads to a fast, unsupervised, layer by layer, training. For that, contrastive divergence is applied to each layer in turn. To turn the network into a classifier, fine-tuning strategy can be applied to the whole network to finalize the training [1]. This is comparable to the backtracking algorithms used to optimize a standard neural network.

When trained in an unsupervised way, a DBN learns to reconstruct its inputs, making it an autoencoder. During learning, the DBN automatically learns features from the raw input. The advantage of this network is that they are generally directly fed with low level data such as pixel colors. It leads to simpler system with few pre-processing steps.

III. Related work

A. Camera-based OCR

In 2005, Liang et al. published a complete survey of Camera-based analysis of text and documents[4]. The various challenges of this problem are studied in detail. The various steps of image processing (text localization, normalization, enhancement, binarization) are analyzed and the different solutions are compared. Although there are many different solutions, they show that many problems still remain open.

In 2013, Jain et al. thoroughly explored the various challenges arisen by Mobile Based OCR[5]. The solutions adopted by standard systems to overcome these challenges are analyzed and compared. They focus on the processing steps allowing later traditional feature extraction and recognition techniques to work as usual. They have shown that even if solutions are getting better and better, there is still room for improvement.

In 2014, Chen et al. compared several features for a Product Identification task[6]. They especially found that resizing and cropping the image may lead to significant accuracy and performance gains. Moreover, global features based on color are generally the most performing one on this kind of images.

B. Sudoku

In 2012, Adam Van Horn proposed a very simple technique to recognize and solve Sudoku puzzles[7], also based on Hough Transform. The four corners of the Sudoku are detected based on the intersections of the detected lines. The digits are then centered in their cells and passed to an Artificial Neural Network (ANN). From each digit image, 100 features are computed. Blank cells are also classified by the ANN and not detected a priori. An improved backtracking algorithm is used to solve the puzzle. For lack of a complete test set, this method was only tested on few images.

Simha et al. presented a different technique in 2012[8]. An Adaptive thresholding is applied to the complete image. Then, the components connected to the borders are removed to reduce noise and improve the later character recognition steps. By using another Connected Components algorithm, the largest component area is identified as the grid. Characters inside the grid are then located by labeling the connected components. After that, a virtual grid is computed based on the enclosing box of the grid and each detected digit is assigned to a cell. Finally the digits are classified using a simple template matching strategy and the Sudoku is solved with a recursive backtracking strategy.

C. Deep Belief Networks

In 2006, Hinton et al. proposed to use a DBN to perform Digit Recognition on the MNIST dataset[1]. The weights of the network are first initialized using a unsupervised algorithm. They are then fine-tuned using a contrastive form of the wake-sleep algorithm[9]. Their network achieved state of the art performance, outperforming the previous Support Vector Machine (SVM) leader. Since this outbreak, interest for Deep Learning architectures has been relaunched.

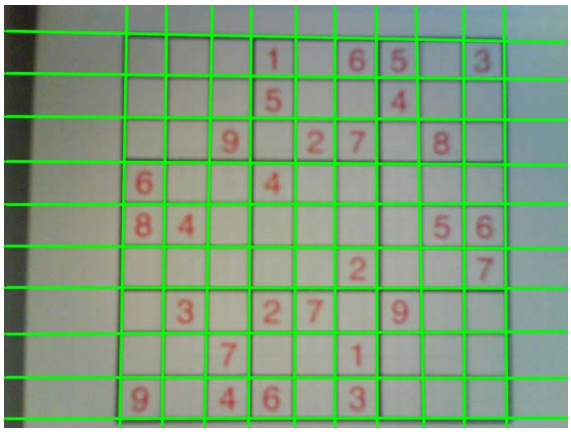

Fig. 3. Detected Lines

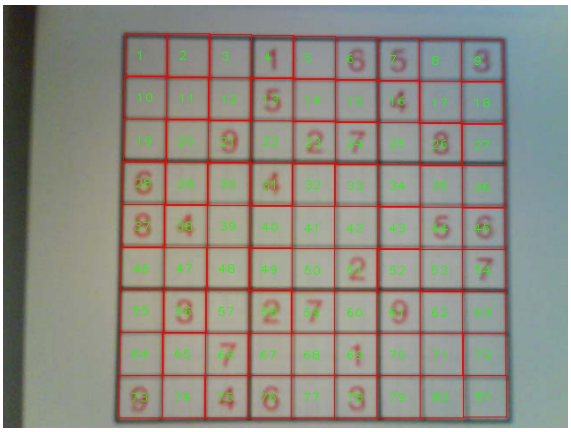

Fig. 4. Fully detected grid with cell numbers

In 2007, Ranzato et al. designed proposed a novel way of learning sparse feature for DBN[10]. Their algorithm, based on the encoder-decoder principle, is called Sparse Encoding Symmetric Machine (SESM). This new algorithm is particularly efficient to train and works without requiring any input processing. Trained networks achieves excellent error rates.

In 2009, Lee et al. presented a new type of DBN, a Convolutional Deep Belief Network (CDBN)[11]. This solution allows to scale the network to larger image sizes, a CDBN being able to classify full-sized natural images. They demonstrated excellent performance on visual recognition tasks. Moreover, their network achieved state of the art on the MNIST dataset.

Since then, Deep Belief Networks and Deep Architectures in general have been used in several domains (Face Recognition, Reinforcement Learning, Handwritten Characters Recognition, etc.). They have proved very successful, often achieving state of the art results.

IV. Dataset

As no free dataset of Sudoku images existed when this project started, it was decided to gather a new dataset to thoroughly test the proposed approach. For this purpose, 160 images has been gathered, from various cell phones and from different local newspapers. The dataset is separated into a training set of 120 images and a test set of 40 images. The actual dataset images are coming from seven different phone models from three different manufacturers. At the time of this writing, a second version of the dataset is developed with new pictures from more recent phones.

Each image is associated with some metadata indicating the content of the cells, 0 indicating an empty cell. The metadata are stored in a simple text file. The pixel resolution and color depth of each image is also kept in the metadata, as well as the brand and model of the cell phone that has taken the picture.

The dataset is available on Github: https://github.com/wichtounet/sudoku_dataset

V. Digit Detection

The system is developed specifically to work on a Sudoku grid containing 9x9 cells. The system works in several steps. First, the lines of the Sudoku are detected. Then, the grid is detected using the lines and the grid is split into 81 cells. Finally, at most one digit is detected in each cell.

A. Line Detection

Before detecting the lines, the image is binarized. This first binarization being here to help detecting the lines, it does not try to preserve the details of the digits. The image is converted to gray-scale. A median blur is used to remove noise and an adaptive thresholding algorithm is performed on the image to binarize it. A second median blur is applied to the binary image to remove further noise. Finally, a dilatation morphological operation is performed on the image to thicken the lines.

Once the image is binarized, edges are detected using the Canny algorithm[12]. Segments of lines are then detected using the Progressive Probabilistic Hough Transform[13] on the detected edges. The probabilistic version of the Hough transform is preferred over the standard form, because it is able to detect segments and not only complete lines. The Hough transform is a fairly standard computer vision technique. First conceived to detect lines in image, it has since been extended to detect arbitrary shapes, such as circles or ellipses. The Hough Transform is especially tuned to detect imperfect shapes, through a voting system.

Connected Component Analysis is then performed to cluster the segments. Only the biggest cluster of segments is kept. The segments that are approximately on the same line are then merged together to form bigger segments. All the segments are then converted to lines. If too many lines are detected (> 20), they are filtered:

1) Lines for which there are no other lines with a relatively equal angle are removed.
2) Lines which are too far from their neighbours compared to the other lines are removed.

Figure 3 shows the result of this step on a Sudoku image.

B. Grid Detection

Once the lines are detected, all the intersections between them are computed. Intersections very close to each other are

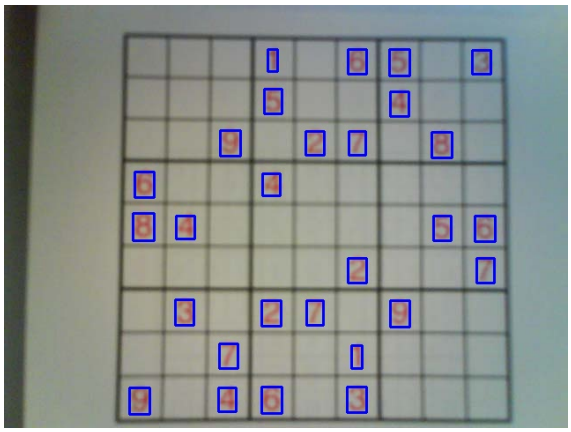

Fig. 5. Characters detected by the system

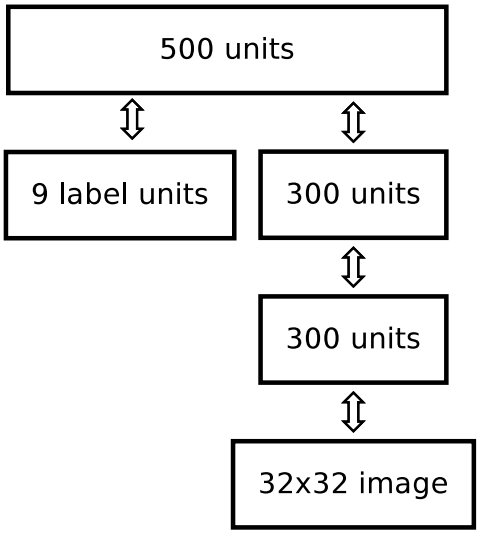

Fig. 6. The network used to classify the source images with digit labels. Each layer of the network is a RBM.

merged. If the lines are correctly detected, there will be exactly 100 detected intersections. If it is not the case, a Contour Detection algorithm[14] is used to detect the greatest contour of the image. The contour outer points are then used instead of the intersection points for the next steps.

To keep only the external points, the four corners of the grid are computed from the Convex Hull of the detected points. The quadrilateral formed by these four points is considered the grid. Each side is split into 9 segments of equal length. A quadrilateral is computed for each cell, from these segments. As the later steps only handles rectangles, the bounding rectangle of each quadrilateral is taken as the final cell.

Figure 4 shows a detected grid on a Sudoku image.

C. Character Isolation

Once each cell is detected, it is necessary to find whether there is a character inside the cell or not. If there is one, its position needs to be precisely found. This step works on the binary image. The lines detected in the previous step are removed from the image.

Then, the algorithm works as follow, for each cell. The sub image of the cell is extracted from the binary image. All the contours inside the image are detected. Only the contours that are of reasonable size are kept. The bounding rectangles of the contours are used for the next steps. Once these rectangles are found, overlapping contours are merged together (several characters are often detected as two or more distinct parts). If the image was of poor quality or if the grid was not perfectly detected, there will still be several candidates at this time. Heuristics based on the shapes of digits and their sizes are used to filter the candidates. At the end, the rectangle with the biggest area is taken as the best character candidate. The background pixel density is used to determine if it is really a character and not an empty cell.

Once the character is properly isolated inside a rectangle, the digit image is centered inside a white squared image of side equal to the maximum of both dimensions of the rectangle. The squared image is resized to a 32x32 image and this image is binarized using a less destructive binarization than the one that was applied to the whole image: an adaptive thresholding, followed by a simple median blur with a small window size to remove the last noise.

At any time during this step, if there are no more candidate or if the last candidate is filtered by the heuristics, the cell is considered empty and 0 is returned.

Figure 5 shows the detected characters on a Sudoku image.

VI. Digit Recognition

Once each digit has been detected, the remaining task is to label each image with a label from 1 to 9 (empty cells are classified a posteriori by previous steps). Each digit is a binary image of 32x32 dimension.

A DBN is used to recognize the digits. The network has four layers (300 hidden units for the first layer, then 300 again for the second, then 500 and finally 9 units for the final layer). Each layer is a RBM. The first layers are using a logistic sigmoid as activation function whereas the last layer uses a simple base-e exponential. This network is made of 551700 parameters. Figure 6 shows the DBN used for this task.

The DBN is used as a feature extractor and a classifier at the same time. No features are extracted from the image. The input of the first layer is 1024 binary units (32x32) and is fed the binary values of the cell image. The output of the last layer is 9 label units indicating a digit from 1 to 9. The last layer is not binary, but contains activation probabilities.

A. Training the network

The network shown in Figure 6 is trained on the complete training set. After digit detection and empty cell removal, 3497 digits are present in the training set.

The weights of each RBM are initialized using a Gaussian distribution of mean 0.0 and with a standard deviation of 0.01. The biases of the visible unit i of the first RBM are initialized to $log(p_i/(1-p_i))$ where p_i is the proportion of training

vectors in which unit i is on. All the other biases are initialized to 0.0

Each RBM is first trained in an unsupervised way, one by one, using CD_1 contrastive divergence. Batches of 10 images are used to train the RBMs. The learning rate has been selected by experiment to be 0.1. The momentum for the first 6 epochs has been fixed to 0.5 and is then increased to 0.9 for the remaining epochs. Weight decay is applied to the all the weights with a weight cost of 0.0002. 20 epochs of contrastive divergence are performed on the network on each layer. The unsupervised training completed in 280 seconds on a 3.3GHz Intel Haswell processor. The last layer of the network is not trained with contrastive divergence and its weights are left to values given by the Gaussian initialization.

In the second phase of training, the complete network is "fine-tuned" with the labels associated with each sample image. Several methods can be used to optimize a deep network, for instance: Stochastic Gradient Descent (SGD), Limited memory Broyden-Fletcher-Goldbard-Shanno (L-BFGS) or Conjugate Gradient (CG). In general, CG and L-BFGS methods are faster than SGD methods and may lead to better models. They also have a better parallel potential (on GPUs) and can be distributed on different locally-connected machines. Moreover, CG may perform better than L-BFGS on large neural networks[15].

For these reasons, a CG method has been chosen to "fine-tune" the classifier. A nonlinear Conjugate Gradient optimization method has been implemented([16], [17] and [18]). A Polack-Ribiere flavor of the Conjugate Gradient method is used to find the search directions. The step sizes are guessed with a line search using cubic and quadratic polynomial approximations and a Wolfe-Powell stopping criteria. CG can be made more scalable by using minibatch training. In this case, batches of 100 images were chosen. The network is trained for 10 epochs. The complete fine-tuning step took about 40 minutes to complete.

B. Testing the network

The network shown in Figure 6 is tested on the complete test set. After digit detection and empty cell removal, 1156 images are present in the test set.

To test the network, the state of the visible units of the first layer is set to the binary values of the pixels of the image to classify. The values of the hidden units are sampled from the inputs. The next layers are using the activation probabilities of the previous RBMs as input. Finally, the topmost label unit with the highest activation probability is taken as the output of the network.

The best trained network achieved an error rate of 0.605% on the 1156 images to classify. Most errors correspond to similar images. Other errors are coming from an image of very poor quality. In this case, the binarization really destroyed information about the digits and the recognition step was made harder. As the recognition step depends on the detection step, if digits are not detected or wrongly detected (displaced or only part of the digit), the training and testing of the digit recognized is impacted. On the complete set, the network achieved 0.47% of error rate.

Step	Min	Max	Mean	Median
Image Loading	1568	70136	2576	1659
Line Detection	27181	52840	31421	30983
Grid Detection	61	16131	1403	70
Digit Detection	9412	12301	10256	10227
Digit Recognition	30847	61237	44869	44127
Total	77835	188232	90238	88333

TABLE I
COMPUTING TIME NECESSARY FOR EACH STEP OF THE PROPOSED SYSTEM. ALL THE TIMES ARE IN MICROSECONDS.

VII. PERFORMANCE DISCUSSION

A. Overall results

On the test set, the complete system has an error rate of 12.5% (five Sudoku images were not perfectly recognized). This error rate is computed at the Sudoku level. A puzzle is considered wrong as soon as one cell of is not classified correctly. If cells are considered directly, the error rate is as low as 0.37% (12 errors on 3240 cells). 58% of the errors on the cells are coming from the detector not identifying correctly an empty cell or not detecting a digit. The remaining errors are coming from wrong classification by the DBN.

Considering the poor quality of some images or the bad conditions in which some other images were taken, these results are rather satisfying. The recognition task is not very complex since all digits are computer-printed. However, the classification is highly depending on the quality of the detection steps and on the quality of the binarization that is done on the image. As the dataset contains images that were taken in variable conditions, no thresholding method is able to perfectly binarize all the images. As such, it causes some digits to be wrongly classified by the DBN.

B. Performance

Table I shows the computing time for each step of the complete process. The experiment has been repeated 3 times and only the experiment with the lowest total time has been taken into account. All the experiments have been run on a 3.3GHz Intel Haswell processor.

The recognition itself is the most time-consuming task of the process. The classification time depends on the size of the network and the number of digits in the Sudoku. The DBN implementation itself has already been tuned for efficiency. On the other hand, it is also the task that is the easiest to parallelize. Indeed, each cell can be independently classified. A bit less time-consuming, the Line Detection task is also quite expensive. The Hough Transform is slow and a lot of processing, transformations and filtering is done on the detected lines. The Digit Detection time is not negligible, but is not as critical as the first other two steps. Detection of the grid is generally almost free. The large difference between the mean and the median here is due to the fact that when the lines are not detected correctly, the system fall back to a contour

detection algorithm which is much more expensive than the simple detection of the grid using the lines.

Given that the algorithms were not specially tuned for performance and the large remaining areas of optimizations, the results are quite satisfying. On average, less than 100ms are necessary to completely detect and recognize a Sudoku inside an image.

VIII. Conclusion and Future Work

We designed and implemented a complete solution to detect and recognize a Sudoku in an image taken from a phone-camera. Our system proved to work well with images under different conditions (light, blur, shadows, small rotations, distortions, etc.), The system fully recognized 35 Sudokus out of 40 from the test set, the others only having small errors on some digits. The DBN have proved a very accurate classifier for the digits, reaching an error rate of 0.609%. The proposed implementation is able to complete in less than 100ms on average for an image. We also collected and made available a dataset of 160 images of Sudoku puzzles.

While our method performed well on our dataset, there is still room for improvement in our system:

- Instead of a complex algorithm to detect the grid and the cells, it could be better to directly use a DBN to detect the digits inside the detected grid or detect the grid itself.
- In cases where Line Detection is not completely accurate and contour detection is used instead, differentiating the grid lines from the digits is not always perfect. The complete algorithm should be unified and improved. Both ways of detecting the grid should work together instead of the second being only used as a fall back.
- While being not slow, there is still room for improvements to make sure the solution works as fast as possible. For instance, digit recognition could be run in parallel on several digits at the same time or the slow line detection algorithm could be improved.
- Several heuristics are too tightly tied to the dataset images. The dataset will need to be completed with images of higher quality to represent more types of images. It will be necessary to make the heuristics more adaptive to handle these new types of images.
- Since our system handles specifically smartphone pictures, it would make sense to port the system to a smartphone application. In that case, an integrated solver would be a great addition to the system.
- Recognition of Sudoku with both handwritten and printed digits could be an interesting challenge.

Acknowledgment

We would like to thank all the people who contributed to the dataset by sending us Sudoku images taken from their phones, in particular Patrick Anagnostaras.

Implementation

The C++ implementation of our recognizer is available online: https://github.com/wichtounet/sudoku_recognizer

The C++ DBN library, used by the recognizer, is also available freely: https://github.com/wichtounet/dbn

A recent version of the Clang compiler is necessary to compile these tools. While the project should be able to build on Windows, it has not been tested under platforms other than Linux.

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[2] S. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical character recognition—a survey," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 01n02, pp. 1–24, 1991.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google

[4] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 7, no. 2-3, pp. 84–104, 2005.

[5] A. Jain, A. Dubey, R. Gupta, and N. Jain, "Fundamental challenges to mobile based ocr," vol. 2, no. 5, May 2013, pp. 86–101.

[6] K. Chen and J. Hennebert, "Content-Based Image Retrieval with LIRe and SURF on a Smartphone-Based Product Image Database," in *6th Mexican Conference on Pattern Recognition (MCPR2014)*, 2014. [Online]. Available: http://ccc.inaoep.mx/~mcpr2014/

[7] A. Van Horn, "Extraction of sudoku puzzles using the hough transform," 2012.

[8] 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*. IEEE, 2012, pp. 1–5.

[9] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The wake-sleep algorithm for unsupervised neural networks." *Science*, vol. 268, p. 1158, 1995.

[10] Y. Marc'Aurelio Ranzato, L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," *Advances in neural information processing systems*, vol. 20, pp. 1185–1192, 2007.

[11] 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, pp. 609–616. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553453

[12] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1986.4767851

[13] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 119–137, Apr. 2000. [Online]. Available: http://dx.doi.org/10.1006/cviu.1999.0831

[14] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following." *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985. [Online]. Available: http://dblp.uni-trier.de/db/journals/cvgip/cvgip30.html#SuzukiA85

[15] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning." in *ICML*, L. Getoor and T. Scheffer, Eds. Omnipress, 2011, pp. 265–272. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2011.html#LeNCLPN11

[16] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Pittsburgh, PA, USA, Tech. Rep., 1994.

[17] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, Feb. 1964. [Online]. Available: http://dx.doi.org/10.1093/comjnl/7.2.149

[18] C. E. Rasmussen, "Minimize a differentiable multivariate function, implementation in matlab," 2006. [Online]. Available: http://learning.eng.cam.ac.uk/carl/code/minimize/minimize.m