The Application of Deep Convolutional Denoising Autoencoder for Optical Character Recognition Preprocessing

Christopher Wiraatmaja, Kartika Gunadi, Iwan Njoto Sandjaja
Department of Informatics
Petra Christian University
Surabaya, Indonesia
christopher.wiraatmaja95@gmail.com, kgunadi@petra.ac.id, iwanns@petra.ac.id

***Abstract*—The process of converting physical documents into digital texts generally requires a scanner tool to obtain high-quality document images. These high quality images will be read by OCR software to get digital text results. The weakness of this method is that OCR software requires a high quality document with low blur noise and no parallax in the image to have high accuracy. We developed an application to increase the document image quality with the help of Deep Convolutional Denoising Autoencoder, afterwards read by the OCR application. The final product of this program is a digital text converted from a document image which has been taken from a smartphone. There is an increase in accuracy using this application by 26.68% in a blurred image compare to standard Tesseract OCR and outperformed Simple OCR in average accuracy testing.**

Keywords—deep learning; OCR; Tesseract OCR; Theano; Keras; convolutional network; autoencoder

I. INTRODUCTION

The primary purpose of this research was to find a way to enhance the quality of document images before OCR software reads the images. OCR software has fewer problems when the document image is taken using scanner tools. Scanner tools can produce a better quality image because there are no skews, blurs, and shadows in the image. There is an innovative large scanner device in international market that reads documents like a book or a paper. The final result of this device is a good non-skewed book page images by using a projected red line by LED or laser pointer and image processing techniques. The resulting image is a good source for OCR software. On the other hand, image taken by a smartphone camera is not a good image to be read by OCR software.

There are several problems that occur when camera is used scan a document. The first main problem is the out of focus blur. This blur effect hinders the performance of OCR reading. The second problem is the skewing of the scanned document. This effect occurs when the image is not taken at an orthogonal angle. The third problem is the shadow effect that occurs when a smartphone user takes the document image. To resolve the first problem, this research proposes a neural network approach. The second problem should be resolved using the homographic matrix method. The third problem should be resolved using adaptive thresholding.

Based on this [1], restoring damaged or noisy images using convolutional denoising autoencoder, the combination of CNN and autoencoder, seems promising. Until now there has been no research on using convolutional denoising autoencoder as a preprocessing method in OCR processes. Thus, this paper uses deep convolutional denoising to resolve the blur problem in OCR.

II. RELATED LITERATURE

A. Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks [2]. Deep learning utilizes fast computer processors and large amounts of data to train large neural networks [3]. Large neural network requires many hyper-parameters to tune. This is one of the problems that deep learning subfields want to resolve. Other research in deep learning has found a faster algorithm to train large neural network.

B. Convolutional Neural Network

Convolutional neural network (CNN) [4] is a special kind of ANN (Artificial neural network). CNN have learnable weights and biases. Just like ANN, CNN are trained with the backpropagation algorithm though they have a different architecture from traditional multi-layer perceptrons. There are many advantages of using CNN as an image classification tool, mainly due to the fact that they can be used as a feature learner and data classifier simultaneously. Bhattacharya and Chaudhuri [5] showed that ConvNet based approaches outperform other more traditional techniques, e.g., SVM, KNN etc. Fig. 1 shows the ConvNet architecture.

The first layer of a CNN is a convolution layer with one or more kernels of fixed size. The inputs presented to these networks are usually images and the kernels convolved with the input. The output of the convolution operations is passed through a non-linear function, i.e., a sigmoid function and a

978-1-4673-9899-2 /17 $31.00 © 2017 IEEE
DOI 10.1109/ICSIIT.2017.32

Rectified Linear unit (ReLu) [6]. Pooling layers are also applied to outputs of the convolution operations. These are methods to reduce the data size, and can be done through averaging over a fixed region, choosing the maximum from a fixed region or any other chosen method. Dense layers are also known as fully connected layers which are used in the last stage of CNN. It connects the CNN to the output layer and constructs the expected number of outputs.

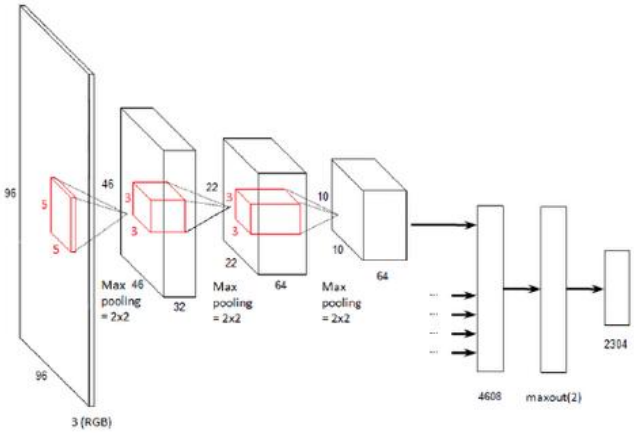

Figure 1. Convolutional neural network architecture [1]

CNN are well recognized models for handwritten digit recognition. CNN is proven to be a good choice among other architecture in image problem like classifying a digit [7]. The main reason is because CNN retains the spatial connectivity of the image, not like the other non-convolutional neural network. Other than classification ConvNets are used for many different purposes, such as face recognition, speech recognition, natural language processing etc [8]. LeNet5 is well known ConvNet architecture developed by Yann LeCun in 1998 [9]. Le-Net5 architecture was able to successfully classify digits and recognize the handwritten digits on bank cheques.

C. *Autoencoder*

The functionality of an autoencoder is simply to transform an input into compressed representation, then reconstruct that compressed representation into reconstructed input [4]. This scheme enables autoencoders to learn useful representations of the input data without needing any labels a-priori. Autoencoder can be implemented using various types of neural networks, e.g., ANN, CNN etc., depending on the desired representation scheme. When used as a pre-training method, the autoencoder is first trained in an unsupervised manner, enabling the encoding part of the autoencoder to adjust its weights to output useful low dimensional representations of the data. These weights are then combined with further layers and trained in a supervised manner.

Auto encoders are used for unsupervised learning; An autoencoder has an input layer, a set of hidden layer and an output layer. In [10], denoising auto encoders were used for extracting features from images. In [11], content based image retrieval was done with the help of deep auto encoders. Erhan and Bengio [12] presented a detailed explanation of the efficacy of unsupervised pre-training for supervised learning.

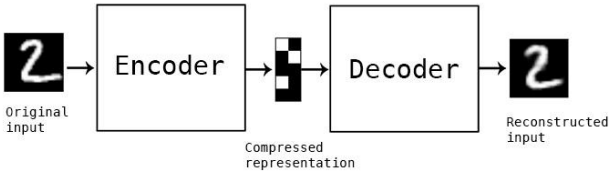

Figure 2. How autoencoder works [2]

Shopon et al. [4] performed research on Bangla hand-written digit classification using Convolutional Autoencoder as preprocessing and CNN as classifier, and achieved promising results.

D. *Text-based Image Segmentation*

Text-based image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. It implies a labelling process which assigns the same label to spatially align units i.e. pixel, connected components or characteristic points such that a group of pixels with the similar label share specific visual features. The result of image segmentation is a set of segments that collectively cover the entire image or a set of contours [13].

E. *Optical Character Recognition*

Optical Character Recognition (OCR) is a conversion of scanned document –printed text or handwritten text– into an editable text for further processing. This technology allows the machine to recognize the text automatically. It is like the combination of the eyes and the mind of the human body. An eye can view the text from the images but is the brain that processes as well as interprets the scanned text seen by human eyes. In the development of computerized OCR system, a few problems can occur [14]. First, there is very little visible difference between some letters and digits for computers to understand. For example, it might be difficult for the computer to differentiate between digit "0" and letter "o". Second, it might be very difficult to extract text embedded in a very dark background or printed on other words or graphics.

F. *Approximate String Matching*

Approximate string matching's general goal is to perform string matching of a pattern in a text where one or both of them have suffered some kind of (undesirable) corruption [15]. A unit used in this algorithm is called edit distance.

Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other

There are 4 operations allowed in edit distance calculation: 1) deletion of a character, 2) insertion of a character,

[1] http://benanne.github.io/images/architecture.png

[2] https://blog.keras.io/building-autoencoders-in-keras.html

3) transposition of 2 characters and, 4) substitution of a character.

There are many algorithms in approximate string matching. The difference between them is the allowed operation in the calculation process, in example:

1) *Levenshtein Distance*: only allows deletion, insertion, and substitution of a character [15].
2) *Longest Common Subsequence (LCS)*: only allows insertion and deletion.
3) *Hamming Distance*: only allows substitution of two characters. Both strings must have the same length.
4) *Jaro Distance*: only allows substitution of two characters.

Levenshtein Distance Algorithm can be used to calculate accuracy of the OCR software by comparing ground truth and actual result of the OCR. The equation to calculate accuracy is shown at (1).

$$\text{Accuracy} = 1 - \frac{\text{LD}(\text{output\_text}, \text{ground\_truth\_text})}{\text{Len}(\text{ground\_truth\_text})} \times 100 \quad (1)$$

LD on (1) is a function to calculate Levenshtein Distance between 2 strings. Len (1) refers to a function to calculate the length of the string. The output of (1) is an accuracy of the actual results compared to ground truth in percentage.

G. Adaptive Threshold

Thresholding is the process to change an image from grayscale form into binary form. The threshold value is the one which changes the pixel value into 0 (black) if it is lower than the threshold value, or 255 (white) if it is higher than the threshold value. In a simple threshold method, the threshold value is only one value and used globally across the image. Adaptive threshold is one of the methods in thresholding. The threshold value is calculated dynamically in each pixel. This method is good to use on an image that has different lightings.

H. Harris Corner Detection

Harris corner detection is an algorithm to find corner features in an image. This algorithm consists of rotation, scale, and an illumination invariant. The corner feature has two lines that intersect to each other. Harris corner detection is fast at finding these features. This algorithm can be seen in detail at [17].

I. Related Frameworks and Libraries

This research utilizes many frameworks and libraries to improve its efficiency. There are 3 related libraries are heavily used in this research:

1) *Theano*: Theano is a library in Python language that gives the user ability to define a symbolic math expression efficiently.
2) *Keras*: Keras is a deep learning framework in Python language. This framework is built on top of the Theano library. All neural networks in this research paper are defined using this framework.
3) *OpenCV*: OpenCV is a computer vision library that provides more than 2500 algorithms. All algorithms in this library are already optimized.

III. PROPOSED METHODS

There are two main systems being proposed in this research: 1) The OCR system, and 2) Autoencoder Training System. The architecture of the system's design can be seen in Fig. 3. This research use autoencoder with convolutional layer. Fig. 2 shows the architecture of the autoencoder. The encoding layer consists of 3 convolutional and down sampling pooling layer pair. The convolution layer has 256 feature map, each using 3x3 window kernel. The pooling layer uses 2x2 window kernel. The decoding layer has symmetrical architecture with its counterpart, except using up sampling pooling layer. The activation function used in this autoencoder is Rectified Linear Unit (Re-LU) [12], except for the last output layer that using a sigmoid activation layer.

A. OCR System Design Analysis

The OCR system has a task to convert a document image into a digital text. The entire OCR system can be seen in Fig. 3(a) and Fig. 3(b).

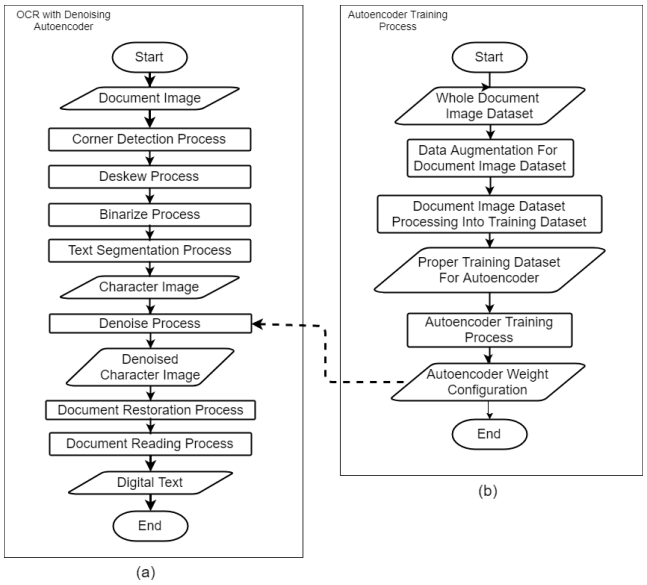

Figure 3. Proposed sytem architecture: (a) OCR system process, and (b) autoencoder training process.

This OCR system consists of 3 main process:

1) *Deskew Process*: Deskew process is a process to remove skew effect on document's image. This process uses the four corner coordinates of the document to calculate the homography matrix of the document's image. The four coordinates itself are acquired from Harris Corner Detection algorithm. This process can produce an error if the corner detected in the image is not exactly four.
2) *Binarize Process*: Binarize process is a process to threshold the image into binary image using adaptive threshold method. The main purpose of this process is to prepare the image into autoencoder input. Another purpose of this process is to remove additive noise and shadow from the image.

3) *Denoise Process*: Denoise process is a process to remove blur noise from character image using deep convolutional denoising autoencoder. The output of this process is a non-blurred character image.

Text Segmentation process and Document Reading Process in Fig. 3(a) is handle by Tesseract OCR. Document restoration process is a process to patch blurred character image in document image using character image output from denoise process

B. Autoencoder Training System Analysis

The autoencoder training system has the task to train an autoencoder model with the training dataset given. In a neural network training context, we need pairs of a dataset, X and Y. X is used in the training process as the input dataset, while Y as the testing dataset. The entire autoencoder training system can be seen in Fig. 3(b).

Document Image dataset consists of 2 groups:

1) *Ground Truth Images*: these images will be used as Y dataset in the autoencoder training process.
2) *Training Image*: this image will be used as X dataset in the autoencoder training process.

This training system consists of 3 main processes:

1) *Data augmentation*: Data augmentation is the process to gain more datasets by applying blur effects into the document image dataset. The document image dataset itself have 6 types of text dataset. This dataset use English and Bahasa Indonesia as text language. The main reason why these two languages are used in this dataset because this research takes place in Indonesia and English is a lingua franca. One non-blurred document image from each text dataset will be applied with 8 types of different kernel sizes. The blur effect used in this image is mean blur. The 8 types of mean blur kernel sizes are 1, 3, 5, 7, 9, 11, 13, and 15. The result of the data augmentation is 6x8 (48) document images. The types of text in the document images are:
    a) Font type Arial, font size 14, language English.
    b) Font type Arial, font size 14, language Bahasa.
    c) Font type Arial, font size 12, language English.
    d) Font type Arial, font size 12, language Bahasa.
    e) Font type Arial, font size 10, language English.
    f) Font type Arial, font size 9, language Bahasa.
2) *Document Image Dataset Processing*: This processing task performs text segmentation on each document after the data augmentation process. The input of this process is set of a ground truth images and training images from each of the document types. Each of the document types consist of 1 ground truth image and 10 training images. All of this document's images will be segmented into character's images by Tesseract. Each of the character's images from training images will be paired with a character's images from ground truth images to form the X and Y dataset for the next process. The output of this process is the X and Y train dataset.
3) *Autoencoder Training Process*: This process uses the X and Y training dataset provided by the previous process.

IV. IMPLEMENTATION

A. Implementation Results

These are 3 results from 3 main processes from Fig. 3(a):

1) *Deskew Process*: Fig. 4 is a result of this process

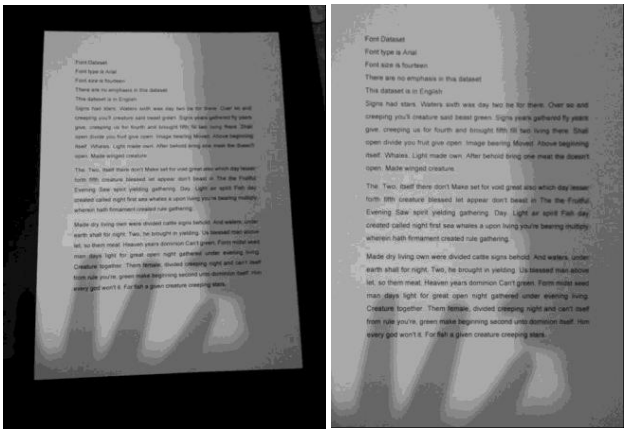

Figure 4. Results of deskew process: before (left) and after (right)

2) *Binarize Process*: Fig. 5 is a result of this process.

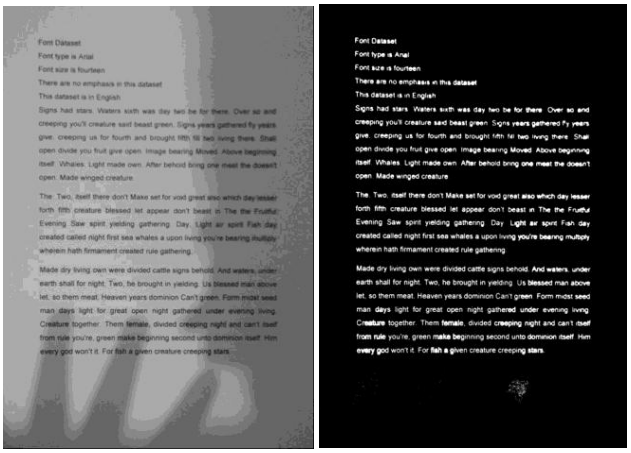

Figure 5. Results of binarize process: before (left) and after (right)

3) *Denoise Process*: Fig. 6 is a result of this process.

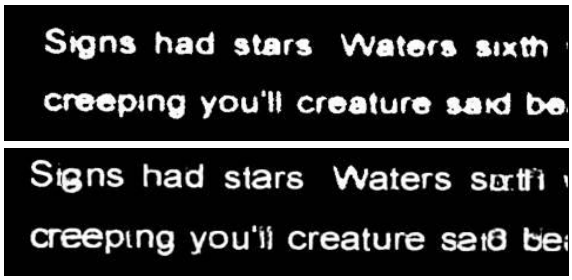

Figure 6. Results of denoise process: before (top) and after (bottom)

B. Implementation Testing

This test measuring metric is using (1) to measure the accuracy. To measure the accuracy improvement, use (2).

$$\Delta A = \text{Current Process Accuracy} - \text{Last Chain Process Accuracy} \quad (2)$$

ΔA in (2) is accuracy improvement. To shorten the process name, the testing will use A for deskew process, B for binarize process, and C for denoise process.

1) *Testing for blur effect*: Dataset use in this testing is the same with the data augmentation process output in Fig. 3(b). This testing is done by placing all the testing dataset into OCR system. After receiving all of the results from the 3 main processes from each document, compare the ground truth text to calculate the accuracy using (1). After that, all the accuracy results are grouped into 3 main processes using mean operation. The results of the testing are seen on Table I.
2) *Testing for shadow effect*: This testing is done by processing 12 document images with the shadow effect on the image. The example of a document image with a shadow effect can be seen in Fig. 7. The actual results of the OCR process are found in its accuracy calculated with (1). The results of the testing are seen on Table I.

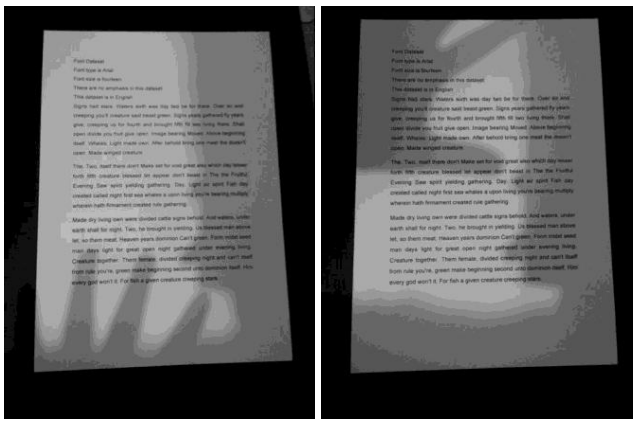

Figure 7. The example of document images with shadow effect

TABLE I. TESTING RESULTS OF BLUR EFFECT AND SHADOW EFFECT

Process	Blur Effect		Shadow Effect	
	Avg. Accuracy (%)	Avg. Accuracy Improvement (%)	Avg. Accuracy (%)	Avg. Accuracy Improvement (%)
Unprocessed	23.63624	—	75.88801	—
A	49.72621	22.29891	66.24976	−9.63824
A+B	58.10562	8.37941	97.00210	30.75236
A+B+C	54.11159	−3.99403	95.70942	−1.29271
Total Improvement	—	26.68429	—	19.82140

There is a mismatched value between accuracy and accuracy improvement in Table I because not all of the datasets used in this test were successful in Fig 3(a). The results of the A+B+C process in this testing are not that good. The C process failed to give a positive accuracy improvement. Therefore, the proposed method of deep convolutional denoising autoencoder is not successful to resolve this blur problem. This happened because in the C process, there were many case where the autoencoder failed to reconstruct the character image correctly. Fig. 8 shows this bad reconstructed images. This research is still not able to find a proper solution to solve this problem where C process failed

Figure 8. The example of the bad reconstructed images

1) *Testing for Font Type used in training dataset*: In previous testing, the text dataset used in the image document is in the font type Arial, including the testing dataset. In this testing, there will be other similar fonts used in the training image document dataset. The fonts that will be used in this testing are: 1) Arial, 2) Open Sans, 3) Verdana, and 4) Calibri. Autoencoder will be trained by each of these font types, and will then be tested again on all four font types.
   The results of this testing are in font type Arial, and is the right choice to use in training the dataset image if the training process only allows one font type to be used. The result is different when multiple font types are used in the dataset. Table II shows that multiple font types in the training process give better average accuracy. This happens because autoencoder already expects the existence of other fonts in the training process, and it knows how to reconstruct that certain type of font.
2) *Testing for OCR System againts another similar OCR Software*: In this test, the OCR system will be compared with a similar program. Each OCR software will try to read an image blurred by mean blur with a multiple size kernel. There is one application is not revealed because it is not a free application. The kinds of OCR software that will be used in this testing are:
   a) *Our OCR system including A+B+C process*
   b) *Standard Tesseract OCR*
   c) *The best OCR application in market*
   d) *SimpleOCR application*

TABLE II. TESTING RESULTS FOR FONT TYPE USED IN TRAINING DATASET

Font Type for Training	Average Accuracy (%)
Arial (A)	64.480932203389870
Open Sans (OS)	59.754590395480240
Verdana (V)	60.331920903954800
Calibri (C)	61.032838983050866
A+OS+V+C	65.020507718398000

TABLE III. TESTING RESULTS FOR COMPARISON WITH SIMILAR PROGRAMS

Blur Kernel Size	Average Accuracy (%)			
	A	B	C	D
1	94.3837	77.7691	95.8041	85.72540
3	94.3057	73.9469	98.0499	87.20740
5	88.6115	61.7004	97.1918	86.66140
7	69.8907	43.6817	94.7737	31.90320
9	41.6536	14.2745	84.0093	3.51014
11	19.1107	3.9001	44.0235	6.08424
13	18.4867	0.2340	0.0780	0.01388
15	—	0.2340	0.4688	0.01867

In this comparison test, the blur kernel size 15 in the OCR A failed to give output because it failed to find corner in the deskew process. The OCR A performed better than than the standard Tesseract OCR and SimpleOCR, but still did not surpass the accuracy of the best OCR in the market.

V. Conclusions and Suggestions

A. Conclusions

Based on this system's analysis and results of the tests, the conclusions of this research are explained as follow:

- OCR system accuracy is very dependent on the text segmentation process. If text segmentation process runs smoothly, there will an improvement in accuracy of about 26.68%.
- Shadow effect on the document image is resolved using adaptive threshold. The accuracy is very high.
- The Deskew process by itself gives an average accuracy improvement of about 22.29%. This is the highest accuracy improvement compared to the other two processes based on Table I.
- The Binarize process gave an average accuracy improvement of about 8.37% after Deskew Process.
- Denoise process give average accuracy about -3.99% after Deskew and Binarize Process. This negative improvement caused by autoencoder fail to reconstruct correctly in denoise process.
- The Arial font type is proven to give more accuracy improvement than any other similar font type. But using many font types in the training phase is proven to give better performance in overall accuracy.
- This OCR system performed better than Standard Tesseract OCR and SimpleOCR software.

B. Suggestions

Suggestions for further research include:

- To have better accuracy in OCR reading process, finding document corner location in the image is an important task. Using an additional actuator (ie. red line/dots by laser pointer) is a good idea to calculate transformation matrix. A better method to find corners of the document means better deskew process performance.
- For better output in the denoise process, this research needs a good quality training dataset and a better data augmentation method, which research still has not found. A good quality training dataset is the one that has controlled variation, so that if the train dataset has a blurred character, the variation that occurs in the dataset is only the blur effect. The more the augmented data resembles the reality, the better the data.

References

[1] F. Chollet, "Building Autoencoders in Keras," The Keras Blog, May 2016. [Online] Available: https://blog.keras.io/building-autoencoders-in-keras.html.

[2] J. Brownlee, "What is Deep Learning?," Machine Learning Mastery, Aug. 2016. [Online]. Available: http://machinelearningmastery.com/what-is-deep-learning/.

[3] U. Pal and B. Chaudhuri, "OCR in Bangla: an Indo-Bangladeshi Language," Proc. IAPR Int. Conf. on Pattern Recognition, Jerusalem (Israel), Oct. 1994, vol. 2, pp. 269–273, doi: 10.1109/ICPR.1994.576917

[4] U. Bhattacharya and B. Chaudhuri, "Handwritten Numeral Databases of Indian Scripts and Multistage Recognition of Mixed Numerals," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 31, no. 3, pp. 444–457, Mar. 2009, doi: 10.1109/TPAMI.2008.88.

[5] M. Shopon, N. Mohammed, and M. Abedin, "Bangla Handwritten Digit Recognition Using Autoencoder and Deep Convolutional Neural Network," Proc. Int. Workshop on Computational Intelligence (IWCI), Dhaka (Bangladesh), Dec. 2016, pp. 64–68, doi: 10.1109/IWCI.2016.7860340.

[6] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," Proc. 27th Int. Conf. on Machine Learning (ICML), Haifa (Israel), Jun. 2010, pp. 807–814.

[7] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of Classifier Methods: A Case Study in Handwritten Digit Recognition," Proc. 12th IAPR Int. Conf. on Pattern Recognition, Jerusalem (Israel), Oct. 1994, vol. 2, pp. 77–82, doi: 10.1109/ICPR.1994.576879.

[8] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time-Series," The Handbook of Brain Theory and Neural Networks, Cambridge (MA, USA): MIT Press, 1998, pp. 255-258.

[9] Y. LeCun, "LeNet-5, Convolutional Neural Networks." [Online]. Available: http://yann.lecun.com/exdb/lenet.

[10] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," Proc. 25th Int. Conf. on Machine Learning (ICML), Helsinski (Finland), Jul. 2008, pp. 1096–1103, doi: 10.1145/1390156.1390294.

[11] A. Krizhevsky and G. Hinton, "Using Very Deep Autoencoders for Content-Based Image Retrieval," Proc. 19th European Symp. on Artificial Neural Networks (ESANN), Bruges (Belgium), Apr. 2011.

[12] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent, and S. Bengio, "Why does Unsupervised Pre-Training Help Deep Learning?," J. Machine Learning Research, vol. 11, pp. 625–660, Feb. 2010.

[13] G. Mehul, P. Ankita, D. Namrata, G. Rahul, and S. Sheth, "Text-based Image Segmentation Methodology," Procedia Technology, vol. 14, pp. 465–472, 2014, doi: 10.1016/j.protcy.2014.08.059.

[14] C. Patel, A. Patel, and D. Patel, "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study," Int. J. Computer Applications, vol. 55, no. 10, pp. 50–56, Oct. 2012, doi: 10.5120/8794-2784.

[15] G. Navarro, "A Guided Tour to Approximate String Matching," ACM Computing Surveys, vol. 33, no. 1, pp. 31–88, Mar. 2001, doi: 10.1145/375360.375365.

[16] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," Proc. 4th Alvey Vision Conference, Manchester (UK), Sep. 1988, pp. 23.1–23.6, doi: 10.5244/C.2.23.