Text Line Recognition - Document OCR

Alolika Gon
University of Massachusetts
Amherst
agon@umass.edu

Abstract

In this report, I have detailed out a two step method to recognize lines of text to be used as a part of an OCR pipeline. The solution is based on training two separate convolutional neural networks, one for character segmentation and another for per-character recognition. These neural networks are trained and tested on a synthetic dataset first. The text-line recognition method is then implemeted on lines of text in identity documents from MIDV-500 dataset, as-well-as on images of text paragraphs from old English newspapers and tested against the ground-truth provided in the dataset generated using ABBYY FineReader Engine 9.0.

1. Introduction

Text recognition using convolutional neural networks (CNN) is a much-researched topic. The process of performing OCR on machine-printed text images can involve several steps, such as layout analysis, zone extraction, per-field segmentation, field recognition, and language model post-processing. In the final project for COMPSCI-682 course, I focus only on character segmentation and recognition. For this project, I have worked on text-line recognition using two CNN architectures, one segmenter to segment an image of a line of text into images of individual characters and one classifier to recognize each of these character images. [5] has developed a similar two-step architecture for recognizing lines of text on identity documents and machine-printed 1961 census documents of England and Wales. For training the two CNNs I have used a synthetic dataset generated using the method described in [4]. I have implemented this two step method to recognize lines of text in identity documents taken from MIDV-500 dataset. The bounding-box for each line of text in the identity document images are taken as input. I have further used this method to recognize paragraphs of text in the the Europeana Newspapers Project Dataset [6]. For this, I have assumed that the layout of each paragraph in a page of text is known beforehand and the boundary points of each paragraph on the page of text is given as input. Each such region of text is then passed through an OCR pipeline for text recognition.

2. Relevant Work

The work done in this project is mostly motivated by the paper [5]. This paper introduces a two-step text-line recognition framework using two CNNs, one for per-character segmentation and another for recognition, and uses dynamic programming instead of image processing techniques in the segmentation step. This segmentation method is language independent and has been tested out on identity documents. The authors in [3] have listed a range of techniques that can be employed for character segmentation like projection analysis, connected component analysis, recognition-based segmentation, and others. The survey paper has mentioned some early works on projection analysis, like examining the ratio of the second-order derivative and height of a projection to find the local minima of a vertical projection ideal for word or character segmentation in [2] or using the ratio of the sum of the distances to local maxima peaks on either side of local minima to the height of the minima for character segmentation. [9] uses a prefiltering method to intensify the projection function. The filter performs AND operation on the adjacent columns prior to the projection in order to produce a deeper valley at columns where potions of the vertical edges of two adjacent characters are merged.

3. Technical Approach

3.1. Character Recognition

The classifier for character recognition, NN_{recg} is a light-weight neural network architecture made of nine convolutional layers followed by a fully connected layer as described in table 1. This CNN takes gray-scale images of size 15 X 19 as input and outputs the probability scores for $|A|$ possible characters. The number of trainable parameters depends on the size of alphabet $|A|$. For a classification task with 70 classes, the number of trainable parameters is $4.9\text{X}10^4$.

Layers						
#Layer	Type	Activation	#Filter	Kernel Size	Stride	Padding
1	Conv	ReLU	8	3 x 3	(1,1)	(0,0)
2	Conv	ReLU	8	3 x 3	(1,1)	(1,1)
3	Conv	ReLU	16	3 x 3	(2,2)	(1,1)
4	Conv	ReLU	16	3 x 3	(1,1)	(1,1)
5	Conv	ReLU	16	3 x 3	(1,1)	(1,1)
6	Conv	ReLU	32	3 x 3	(2,2)	(1,1)
7	Conv	ReLU	32	3 x 3	(1,1)	(1,1)
8	Conv	ReLU	32	3 x 3	(1,1)	(1,1)
9	Conv	ReLU	8	3 x 3	(1,1)	(1,1)
10	Fully Connected	Softmax	|A| neurons			

Table 1. classifier Architecture

Layers						
#Layer	Type	Activation	#Filter	Kernel size	Stride	Padding
1	Conv	ReLU	6	5x5	(1,1)	(0,2)
2	Conv	ReLU	6	5x5	(2,1)	(2,2)
3	Conv	ReLU	6	7x3	(1,1)	(3,1)
4	Conv	ReLU	6	5x5	(1,1)	(2,2)
5	Conv	ReLU	6	3x3	(1,1)	(1,1)
6	Conv	ReLU	6	5x5	(2,1)	(2,2)
7	Conv	ReLU	6	2x5	(1,1)	(1,1)
8	Conv	ReLU	6	5x5	(2,1)	(0,2)
9	Conv	ReLU	6	3x5	(1,1)	(1,2)
10	Conv	RBF	1	3x3	(1,1)	(0,1)

Table 2. Segmenter Architecture

3.2. Character Segmentation

The segmenter CNN, NN_{seg} is a fully convolutional network (FCN) which is employed to compute a vertical projection of the image and to decide whether there is a cut at a position or not. As described in table 2, the kernel size along with the padding and strides used in each of the convolutional layer is such that an input gray-scale image of shape HXW when passed through NN_{seg} will result in an output vector of shape 1XW. Each element in this output vector represents the probablity of the image getting segmented at that position. The local maximas (P1) in the generated distribution represent the points of cut. Probable bounding boxes for characters are then generated using pairs of non-zero points from P1 with distance from the predetermined interval $fd \in [fd_{min}, fd_{max}]$ and confidence of corresponding characters predicted by the classifier CNN.

3.3. Projection Analysis

In a binary image of text where the background pixels are black or zero and the foreground pixels are white, a local minima or a valley of zeros in the horizontal histogram projection represents the space in between two consecutive lines. Such a horizontal projection curve for an image of a paragraph of text is smoothed using a Savitzky-Golay filter to remove the distortions and then a local minima is found to segment lines in a paragraph of text.

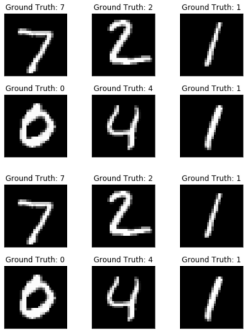

Figure 1. Example of images from MNIST dataset

4. Datasets

4.1. MNIST

The famous MNIST dataset [8] has been used as a starting-point for the classifier and to evaluate it's performance. The MNIST dataset consists of 70000 images of handwritten digits, 60000 of which form the training set, and the rest 10000 are treated as the test set. The state-of-the-art result reported for MNIST is the 0.23% error rate for non-ensemble classifiers [5]. The per-character classifier is trained on the training set of the MNIST dataset and then the classification accuracy is calculated on the MNIST test set.

4.2. Synthetic Dataset

For training the classifier and segmenter CNNs a synthetic dataset is used which can be generated using the method described in [4]. The link for downloading the dataset in mentioned in [5]. The dataset contains 7292 images similar to the example in Figure 2. The dataset is generated using several different fonts from GoogleFonts over backgrounds from images of various documents. The ground-truth for each such image is given in the JSON format:

- 'line_rect' is the bounding box for each line of text
- 'cuts_x' stands for the cuts between the letters, i.e. the ideal segmentation results.
- 'start_x' and 'end_x' outlines the exact boundary points for each character along the horizontal direction.
- 'let_blines' outlines the exact boundary points for each character along the vertical direction.
- 'values' are the Unicode character codes in decimal number system for each character in the line of text.

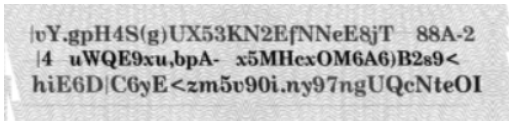

Figure 2. Example of image from synthetic dataset

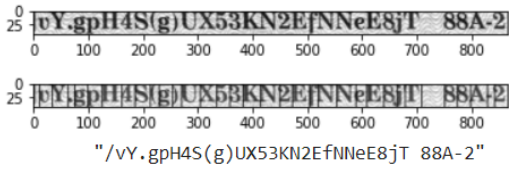

Figure 3. Example of cropped line of text, segmentation ground-truth, and per-character classification ground-truth.

Figure 4. ID image example and ground-truth in MIDV-500 dataset

4.3. MIDV-500

The MIDV-500 [1] dataset contains images of ID samples captured by smart-phones. Text lines in these ID images are printed with proportional and mono-spaced fonts. The dataset provides JSON files containing the bounding box for each line of text along with the actual text in string format. Example of an image and ground-truth from the dataset is shown in Figure 4.

4.4. Europeana Newspapers Project (ENP) Dataset

This dataset contains images of pages of newspapers from Europe's major libraries [7] and is available online on the Prima website. The dataset contains scanned images, metadata and ground truth. The dataset provides XML files with geometrical ground-truth in the form of boundary-points of each layout or paragraph of text i.e. a representation of the ideal result of a processing step like OCR or layout analysis on the level of individual newspaper page as well as the textual ground-truth. Figure 5 and 6 show an example of a newspaper page and one of the cropped out

Figure 5. Newspaper page image example from ENP dataset

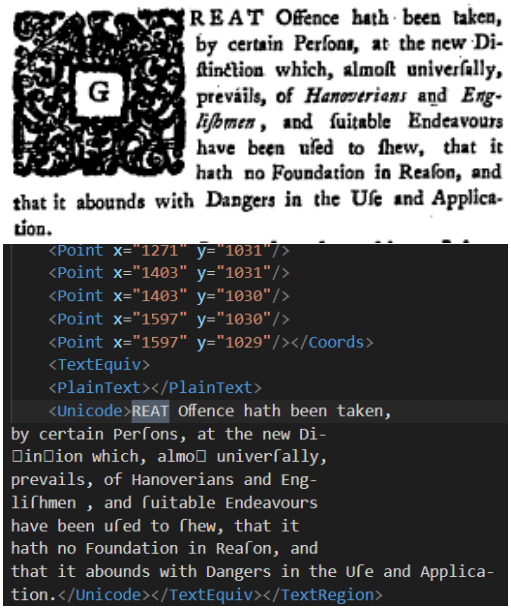

Figure 6. Example of text layout and ground-truth in ENP dataset

regions or layouts and the textual ground-truth. The text ground-truth is generated by a state-of-the-art commercial OCR engine, ABBYY FineReader Engine 9.0 and is not 100% accurate.

5. Experimental Evaluation

5.1. Classifier

The classifier CNN architecture as described in Table 1 is trained by reducing categorical-crossentropy loss. When trained on the MNIST training dataset with 10 output

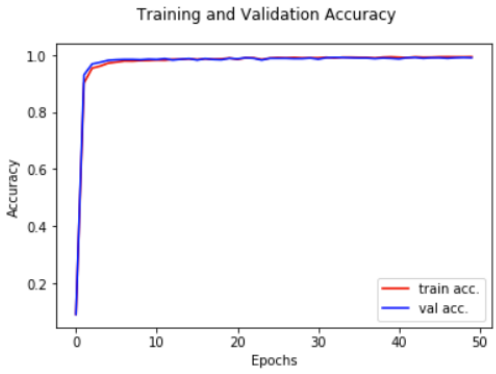

Figure 7. Accuracy plot of classifier for MNIST dataset

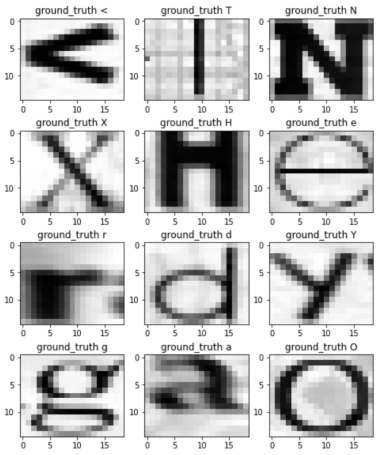

Figure 8. Examples of cropped character images

classes for 10 digits, the CNN gives an accuracy of 99.06% on the MNIST test dataset. All the intermediate layers are batch-normalized for faster training.The Accuracy plot is shown in Figure 7.

The same CNN architecture is trained on images of characters in the synthetic dataset as shown in Figure 8. These images are cropped using the 'start_x', 'end_x' and 'let_blines' values in the ground-truth and resized to 15X19. The total number of character classes is 70 including 26 upper-case letters, 26 lower-case letters, 10 digits and special characters like "$(), . - / <$ ". The CNN is trained on about 5000 images per character class and gives 95.22% top-1 accuracy and 99.44% top-3 accuracy. The CNN is trained by reducing categorical cross-entropy loss. The initial learning rate of 0.01 is dynamically reduced when the loss curve forms a plateau for 10 epochs. The kernel parameters are L2 regularized with a weight 0.001 to prevent over-fitting. The accuracy and loss plots are shown in Figure 9.

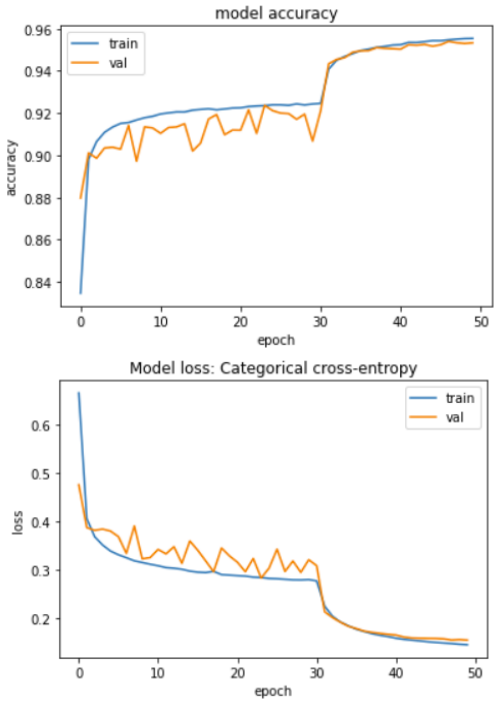

Figure 9. Accuracy and loss plots of classifier for synthetic dataset

5.2. Segmenter

To find the bounding-box around each character in a line of text, a cropped gray-scale image (Figure 3) of the text line resized to 33 X 700 is passed through the CNN described in table 2. This FCN generates output vector P, the probability of a cut between two characters at each of the 700 pixels along the horizontal axis. The model is trained on around 19,000 text line images taken from the synthetic dataset and tested on about 2000 images from the same dataset. The FCN is trained by reducing the mean square error between the generated projection and the ground-truth using Adam optimizer. All the intermediate layers are batch-normalized for faster training. The kernel parameters are L2 regularized with a weight 0.001 to prevent over-fitting. The model has been trained separately with different activation functions for the last layer. The mean-square error of the test dataset on using ReLU, sigmoid (Eq. 1), Gaussian radial basis function (Eq. 2), inverse quadratic radial basis function (Eq. 3) and inverse multi-quadratic radial basis function (Eq. 4) as the activation after the last layer is shown in Table 3. The training and validation loss during each epoch of training are plotted in Figure 10.

$$S(x) = \frac{1}{1 + \exp(-x)} \tag{1}$$

$$S(x) = \exp(-\epsilon x^2) \tag{2}$$

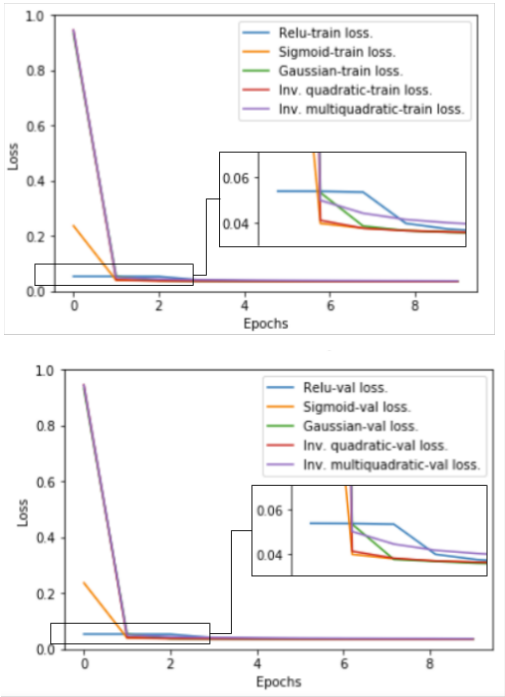

Figure 10. Training and Validation loss plots of segmenter network trained on synthetic dataset

Activation function	**MSE on test set**
ReLU	0.035
Sigmoid	0.034
Gaussian RBF	0.034
Inverse Quadratic RBF	0.035
Inverse Multi-quadratic RBF	0.037

Table 3. Mean square error on test-set in the synthetic dataset

$$S(x) = \frac{1}{1+\epsilon x^2} \tag{3}$$

$$S(x) = \frac{1}{\sqrt{1+\epsilon x^2}} \tag{4}$$

Once the horizontal projection is generated using the segmenter FCN, the local maximas ($P1$) are considered as the probable points of cut along the horizontal axis of the input image. While processing the training images, the minimum (fd_{min}) and maximum (fd_{max}) per-character widths are recorded. Pairs of points from $P1$ with distance within the predetermined interval $fd \in [fd_{min}, fd_{max}]$ are used to generate candidate characters. Out of all possible character images, the top m candidate characters are classified

Character Image Post-Processing Score	Performance Metrics		
	PCR	Average Levenshtein Distance	Weighted Average Levenshtein Distance
Eq 5	38.84	25.08	24.96
Eq 5 + keep probability of 0.5	42.43	22.58	22.24
Eq 5 + keep probability of 0.75	55.67	16.64	16.81
Eq 6	37.16	24.04	24.93
Upper Bound Values	81.13	7.11	7.36

Table 4. Performance for test-set from synthetic

according to the scores defined by either either Eq. 5 or Eq. 6. While using score defined by Eq. 5, an additional keep probability can be considered to be used as a threshold on the scores predicted by the classifier model for the individual character images.

$$s = P(x_1) + P(x_2) \tag{5}$$

$$s = 0.5(P(x_1) + P(x_2)) + C(x_1, x_2) \tag{6}$$

In the above equations, x_1 and x_2 are the left and right boundary of a candidate character, $P(x)$ is the probability of the image getting segmented along x as predicted by the segmenter CNN, $C(x_1, x_2)$ is the confidence score of first alternative predicted by the classifier CNN for the character image cropped using the two boundaries. m here is taken as:

$$m = \frac{W_{img}}{fd_{mean}} \tag{7}$$

where W_{img} is the width of the image, which is 700 pixels in this case and fd_{mean} is the average per-character width recorded from the dataset. To compare the predicted text with the original and thus evaluate the overall pipeline, average Levenshtein distance and weighted average Levenshtein distance can be considered.In [5], a per-character recognition (PCR) rate is used to evaluate the overall performance which is defined as:

$$PCR = 1 - \frac{\sum_{i=1}^{L_{total}} min(lev(l_{i_{ideal}}, l_{i_{pred}}), len(l_{i_{ideal}}))}{\sum_{i=1}^{L_{total}} len(l_{i_{ideal}})} \tag{8}$$

where L_{total} is the total number of lines, $len(l_{i_{ideal}})$ is the length of the i^{th} line, and $lev(l_{i_{ideal}}, l_{i_{pred}})$ is the Levenshtein distance between the recognized text and the ground-truth. The average Levenshtein distance and weighted average Levenshtein distance between original and predicted

lines of text and PCR for the segmenter models with different final activation functions has been shown in Table 4. The PCR rate and average Levenshtein distances calculated by using 100% accurate segmentation values i.e. the ground-truth values in the synthetic dataset has been used to estimate the upper bound of these performance metrics.

5.3. Further Implementations

The overall pipeline with the two pre-trained neural networks and the post-processing techniques is applied on 330 lines of text cropped out of ID images from the MIDV-500 dataset. The size of these images varies greatly, from a minimum width of 7 pixels to a maximum of 1300 pixels and from a minimum height of 10 pixels to maximum 88 pixels. So resizing these images to a shape of 33X700 pixels as required by the segmenter network distorts the images. Sine we consider a fixed maximum number of classified characters as defined by Eq. 7, the pipeline on resized images does not perform well. But, instead of resizing the smaller images (Method 1), appending the image of the cropped line of text next to itself to construct a new image which is 700 pixels wide yields much better results. If an image has to be appended k times following this method, the resultant string is divided by $k+1$ and only the first part is considered as the predicted string (Method 2) or the ground-truth text is also appended to itself k times (Method 3). The average Levenshtein distance and PCR rate using the above methods and the segmenter classifier with Gaussian RBF as activation for the final layer are shown in Table 5.

The overall OCR pipeline is implemented and evaluated using newspaper clippings from the Europeana Newspapers Project Dataset [6]. The text font in these scanned images and the width to height ratio per character are quite different from that of the character images used to train the neural networks. Some characters in certain words are skewed and even connected. All these reasons make this a much harder problem. The ground-truth of this dataset provides boundary points for a vertical set of lines of text. The shapes of these layouts are not always rectangular. Such an example is shown in Figure 6. To extract each line, the pixel values along each row is summed up and the generated horizontal projection is analyzed. The cropped image of each layout is first inverted such that the foreground pixels forming the characters are high (1.e. white in color). The horizontal projection of this image is generated and smoothed and the points of local minima are noted for segmenting the paragraph into text line images. The best filter size and function type to be used for smoothing out the distortions in the projections can be determined by following a process similar to hyper-parameter grid search. In this case, the Savitzky–Golay filter is used with window size 21 and a first-order polynomial function and 20 layout images are manually selected for which the per-line segmentation is accurate. These text line images are then passed through the pre-trained segmenter and classifier. The performance on these images are shown in Table 6.

Image Pre-Processing Method	Character Image Post-Processing Score	Performance Metrics		
		PCR	Average Levenshtein Distance	Weighted Average Levenshtein Distance
Method 1	Eq 5	0.61	35.32	35.98
	Eq 5 + keep probability of 0.5	0.87	30.622	31.94
	Eq 5 + keep probability of 0.75	56.65	21.71	10.50
	Eq 6	12.98	34.51	31.57
Method 2	Eq 5	3.06	11.03	16.38
	Eq 5 + keep probability of 0.5	5.63	10.19	15.32
	Eq 5 + keep probability of 0.75	69.13	8.48	4.17
	Eq 6	8.66	11.03	16.38
Method 3	Eq 5	6.02	29.10	29.47
	Eq 5 + keep probability of 0.5	10.39	26.10	27.06
	Eq 5 + keep probability of 0.75	61.78	22.47	11.25
	Eq 6	7.88	32.32	31.67

Table 5. Performance on MIDV-500 dataset

Character Image Post-Processing Score	Performance Metrics		
	PCR	Average Levenshtein Distance	Weighted Average Levenshtein Distance
Eq 5	21.84	37.08	38.96
Eq 5 + keep probability of 0.5	23.43	36.58	38.24
Eq 5 + keep probability of 0.75	36.16	31.04	31.93

Table 6. Performance on examples from ENP dataset

6. Conclusion

The advantage of having two separate neural networks for character segmentation and recognition is that the segmenter neural network can be language independent. But as we see in the results above, we lose important information while pre-processing and resizing the images in order to pass them through the neural networks, specifically when the image size is a bit different from the size of images the networks are trained on. This approach can work well only for lines of text with similar number of characters per line, since the post-processing method considers a fixed number of classified characters depending on the synthetic dataset the segmenter is trained on. An end-to-end neural network to do both the classification and segmentation tasks together might solve this problem, but it is difficult to find or create a dataset required for efficient training of such an end-to-end model.

References

[1] V. V. Arlazarov, K. B. Bulatov, T. S. Chernov, and V. L. Arlazarov. Midv-500: a dataset for identity document analysis and recognition on mobile devices in video stream. 43(5), 2019.

[2] H. S. Baird, S. Kahan, and T. Pavlidis. Components of an omnifont page reader. In *Proc 8th ICPR*, pages 344–348, 1986.

[3] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(7):690–706, 1996.

[4] Y. S. Chernyshova, A. V. Gayer, and A. V. Sheshkus. Generation method of synthetic training data for mobile ocr system. In *Tenth International Conference on Machine Vision (ICMV 2017)*, volume 10696, page 106962G. International Society for Optics and Photonics, 2018.

[5] Y. S. Chernyshova, A. V. Sheshkus, and V. V. Arlazarov. Two-step cnn framework for text line recognition in camera-captured images. *IEEE Access*, 8:32587–32600, 2020.

[6] C. Clausner, C. Papadopoulos, S. Pletschacher, and A. Antonacopoulos. The enp image and ground truth dataset of historical newspapers. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 931–935, 2015.

[7] C. Clausner, C. Papadopoulos, S. Pletschacher, and A. Antonacopoulos. The enp image and ground truth dataset of historical newspapers. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 931–935. IEEE, 2015.

[8] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[9] S. Tsujimoto and H. Asada. Major components of a complete text reading system. *Proceedings of the IEEE*, 80(7):1133–1149, 1992.