2009 10th International Conference on Document Analysis and Recognition

Error-Correcting Output Coding for the Convolutional Neural Network for Optical Character Recognition

Huiqun Deng, George Stathopoulos, and Ching Y. Suen
Center for Pattern Recognition and Machine Intelligence, Concordia University, Canada
{hqdeng, suen}@encs.concordia.ca, g_statho@cse.concordia.ca

Abstract

It is known that convolutional neural networks (CNNs) are efficient for optical character recognition (OCR) and many other visual classification tasks. This paper applies error-correcting output coding (ECOC) to the CNN for segmentation-free OCR such that: 1) the CNN target outputs are designed according to code words of length *N*; 2) the minimum Hamming distance of the code words is designed to be as large as possible given *N*. ECOC provides the CNN with the ability to reject or correct output errors to reduce character insertions and substitutions in the recognized text. Also, using code words instead of letter images as the CNN target outputs makes it possible to construct an OCR for a new language without designing the letter images as the target outputs. Experiments on the recognition of English letters, 10 digits, and some special characters show the effectiveness of ECOC in reducing insertions and substitutions.

1. Introduction

A convolutional neural network (CNN) is a special neural network with an architecture that extracts topological features insensitive to shifts and distortions [1][2][3]. It is known that a CNN can perform character recognition with very high accuracy given isolated characters. However, correct segmentation of characters from text is not an easy task for noisy images, and incorrect segmentations degrade recognition accuracy of the classifier. In fact, humans cannot perform segmentation without knowing the character patterns of a language. This applies to artificial neural networks: correct segmentation requires knowing the patterns of the target characters. In [1] [2], it is shown that the CNN can be trained from 10 digits with left-right contexts, and can recognize the digits without segmentation, by sliding an image window along a text line and evaluating the confidence level of the CNN outputs for each window position. However, the potential for a CNN to perform segmentation-free optical character recognition has not been verified on a larger number of classes including 52 English letters, 10 digits and some special symbols.

In this paper, the potential of the convolutional neural network LeNet5 [1][2] is investigated as a classifier in a segmentation-free optical character recognizer of 81 classes including 52 lower and upper case English letters, 10 digits, 18 special symbols and the space between letters. The LeNet5 CNN is trained from letters positioned at the centers of 28 × 28 pixel windows and flanked with left-right context letters. When such a trained CNN is used with the sliding window to perform segmentation-free OCR, it has been found that erroneous insertions appear frequently even when the sliding window is centered at the gaps between adjacent letters. This is caused by the limitation of the place code used in the LeNet5 CNN (i.e., its k^{th} output is designed to be 1 if the input image is from class *k*, and all other outputs are designed to be -1), and by the fact that it makes its classification decision based on the neuron producing the highest output exceeding a predefined confidence level. It is difficult to define a confidence level between 0.00 and 1.00 appropriate for both recognition of on-center letters and rejection of off-center letters for a large number of classes. To overcome this problem, this paper proposes a new method of rejection and recognition without using a confidence level by considering the classification and rejection as a kind of telecommunications problem. In telecommunications, received noisy codes can be rejected or decoded by considering their Hamming distances to the designed code words. This work designs the target outputs of the LeNet5 CNN according to error-correcting theory to make the minimum code distance of the CNN outputs as large as possible. It is noted that such an idea has been called error-correcting output coding (ECOC) in machine learning [4][5]. The goal of this work is to enhance the rejection and recognition capability of the CNN to perform segmentation-free OCR. The motivations of using the CNN for segmentation-free OCR are as follows. First, using the CNN avoids the need to perform feature extraction, as it has a built-in structure similar to the perceptive field found in the visual cortex, and can extract topological features. In contrast, an HMM-based segmentation-free OCR requires pre-processing to extract topological features. Second, as shown later, constructing training data for a neural network is less expensive than for an HMM-

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.144

based segmentation-free OCR, which requires choosing representative word contexts for a language. Section 2 describes our method of constructing training images with centered letters and left and right contexts. Section 3 presents our method of designing the code words for the outputs of the LeNet5 CNN. Section 4 presents the training and test results, and examples of recognized text lines using the CNN-based segmentation-free OCR. Section 5 presents results and discussions, and Section 6 is the conclusion.

2. Constructing Training Images of Characters with Contexts

For a CNN-based OCR to perform segmentation-free character recognition, a sliding window is needed. Assume that the sliding window has a height H and width W, and that the distance between the ascender line and the descender line of a text line to be recognized is h (pixels) and its width is w (pixels). Before recognition, the image of the text line should be normalized by a ratio of H/h to have an ascender-to-descender distance of H and have a width of wH/h. The sliding window is shifted along the normalized text line and, for each window position, the image from the sliding window is sent to the CNN for classification or rejection. As the sliding window can cover left and right contexts of a letter, the CNN must be trained from character images with left and right contexts, instead of the images of isolated characters.

In this paper, such training images are synthesized from the binary images of isolated upper- and lower-case English letters, 10 digits and 18 special characters. The binary images were previously obtained at CENPARMI by scanning the prints of the isolated characters, each having 3 sizes (18, 11, 8 point), 4 styles (normal, bold, italic, bold italic) and 36 fonts (Times New Roman, Arial, etc.). Three different quantization levels were used to obtain the binary images of different quality.

In synthesizing the training images, the image of an isolated letter is normalized and pasted at the left-right center of a window with the same size as the sliding window, with the letter's vertical positions relative to the x-lines, ascender line and the descender line being unchanged. For example, if the height and width of the sliding window are 28, then the normalized Times New Roman English letters [a, c, e, m, n, o, r, s, u, v, w, x, z] have a height of 14 and sit on line 21, letters and special characters [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, b, d, f, h, i, k, l, t, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, !, $, %, &, +, -, /, <, >, ?, @, [,], #,×,€, ¥, ®, £] have a height of 21 and sit on line 21, letters [g, p, q, y] have a height of 21and sit on line 28, and the letter [j] has a height of 28 (approximately) and sits on line 28. The left and right neighborhoods of the center letter are partial or entire images of left and right context letters. The context letters are randomly selected from letters with the same style, font, and size as the center letter, and their images are normalized and added on the neighborhoods of the center letter. The distance between the center letter and the context letter is randomly determined, allowing possible contact to simulate the cases of touching pairs caused by the quantization of noisy images.

It should be noted that the ratio of the height of "x" and the height of upper-case letters is estimated to be approximately 1:2 according to the letter samples from Times New Roman. This ratio varies for other fonts such as Arial, Georgia, etc. In this paper, however, the same ratio was used when normalizing the images of letters in different sizes, styles, and fonts into a height of 14, 21, or 28 to synthesize the training images. The discrepancy between training images and text images degrades the recognition rate. Some examples of synthesized training images containing centered target letter and flanked letters are shown in Fig. 1.

Fig. 1 Examples of the synthesized training images of '3', '8', 'N', 'B', and 'a' with arbitrary contexts.

3. Error-Correcting Output Coding for the CNN

In LeNet5, the outputs of the CNN are designed according to the place code: for a letter image presented at the input, the CNN produces a single high positive level at one output with all other outputs being low negative levels, and the decision is given by the index of the highest positive output. Such a CNN needs 81 outputs to represent 81 classes, which is not efficient. Also, the place code introduces two problems for the segmentation-free OCR. First, even if the sliding window is not centered at a letter, the CNN may still produce positive levels in the outputs, and the highest level may be greater than the predefined threshold and this leads to the insertion of an extraneous character in the recognized output. Second, in the view of coding theory, the code distance given by the place code is 2. Such a small code distance does not have the ability to correct or reject errors. Viewing the classification and rejection as a problem of telecommunications, this paper presents the design of the CNN outputs according to code words with maximum minimum code distances. With increased code word distance, the CNN output patterns produced from un-centered sliding windows are expected to have

larger code distances from the target code words than those produced from centered sliding windows.

Determining error-correcting code words for pattern recognition problems is an open research topic. This paper searches for the code words as follows. Let the designed minimal code distance be D, and the total number of the CNN outputs be N. To further reject false alarms, this paper designs the number of 1's in each code word as M. 1) Put an N-bit binary string with (N-M) 0's and M 1's in an empty codebook; 2) From [0, 0, …, 0] to [1, 1, …, 1], i.e., from all N-bit binary strings, find those with M 1's and take them as candidates, and denote the number of candidates as K; 3) for k=1 to K, calculate the Hamming distances between k^{th} candidate and all code words in the codebook; only if all its Humming distances are greater than or equal to D, put the k^{th} candidate in the codebook; 4) evaluate the next candidate as in step 3. In the end, if the number of code words in the codebook is less than that of classes, adjust the values N and D, and search again. Code words with larger N and D take a longer to find.

The final error-correcting code words are then selected from the codebook according to the two properties desired in [4]: each column of the codeword matrix should contain both 1's and 0's to form the target outputs for each output unit of the CNN, and different columns of the codeword matrix are not complementary to each other to ensure that different binary learners (i.e., different output units of the CNN) learn different tasks from each other. The code words with N=17, D=6, M=8 for the 81 classes obtained using the above method are shown in Table I.

Table I. The designed code words of the 81 characters

Codeword	Char	Codeword	Char	Codeword	Char
00000000011111111	a	00101010011110010	1	01010010011100110	S
00000011100011111	b	00101011100100110	2	01010010100101011	T
00000011111100011	c	00101100010011101	3	01010011111010001	U
00000101101101101	d	00101100101100011	4	01010100111100001	V
00000101110110110	e	00101101011101000	5	01010101101001010	W
00000110101111010	f	00101110110001010	6	01010110100010110	X
00000110111010101	g	00101111100010001	7	01011111001010000	Y
00000111010111001	h	00110000110111001	8	01100000110100111	Z
00000111011001110	i	00110001001101110	9	01100001011011100	!
00001001111011001	j	00110001111010010	A	01100001101110001	#
00001010110101101	k	00110010010010111	B	01100010011101001	$
00001011001110101	l	00110010101100101	C	01100100100011011	%
00010100110001111	m	00110011001011001	D	01100110001110100	&
00010101001010111	n	00110101100011100	E	01100111001000011	+
00011000100110111	o	00110111000110010	F	01100111100101000	/
00011000111101010	p	00111110011000001	G	01110010110001100	<
00011001010011110	q	01001000111110100	H	01110100010101010	>
00011010001001111	r	01001010000111110	I	01111001000010101	?
00011011100111000	s	01001010101010011	J	01111001110100000	@
00011011111000100	t	01001011010001011	K	10001001101110010	[
00011100001111001	u	01001100011011010	L	10001010011011100	]
00011101010100011	v	01001101001100110	M	10001100010101110	€
00011110010110100	w	01001101110000101	N	10001100110010011	¥
00100110000101111	x	01001110101001100	O	10001101001001011	£
00101000101011110	y	01010000101011101	P	10001110111100000	®
00101001000111011	z	01010001010101101	Q	10001111000010110	×
00101001011000111	0	00101010011110010	R	10010001011110001	space

4. Training, Recognition and Rejection Methods

This paper uses the LeNet 5 CNN as the base line classifier. The LeNet5 CNN has five layers: one input layer, 3 hidden layers and 1 output layer. Its input layer takes images with a height of 28 pixels and a width of 28 pixels, and its output layer has the same number of outputs as the number of classes to be recognized. The detailed descriptions about the structure of hidden layers can be found in [1][2]. This paper modifies the target outputs of the LeNet5 CNN according to the 1's and 0's of code words in Table I for the 81 classes. For example, given the code word [0000000011111111] for the letter 'a', the target outputs of the CNN are designed to be [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]. In this paper, such a modified CNN is called the CNN with ECOC, and the original LeNet5 CNN is called the CNN with place code. The CNN with ECOC and the CNN with place code are then trained respectively using the back propagation algorithm.

The training images have a size of 28 × 28, whose left-right center are letters or characters with arbitrary contexts as mentioned in Section 2. The OCR is designed to recognize 10 digits, 52 English letters, 18 special characters, and the space between letters or special characters. For each letter or character there are 4 styles, 3 sizes, 36 fonts, and 3 quantization levels. 2 training images are selected for each letter in each combination of style, size, font, and quantization level. The total number of training images is 70848. The training was terminated after 400 epochs, and the CNN with ECOC achieved a training error rate of 1.21% and a test error rate of 0.98%. The test dataset contains 11808 images of the letters and characters as mentioned above and all are different from the training images. As an illustration, the outputs of the CNN with ECOC given the test images of different characters are presented using dots in Fig. 2. To save space here, only the CNN outputs for the first 30 classes (a, b, …, z, 0, 1, 2, 3) are shown. In contrast, the CNN with place code obtained a training error rate 0.73% and a test error rate 1.19%. In evaluating classifiers, test error rates should be used. The CNN with ECOC obtained

relative error rate reduction by (1.19-0.98)/1.19=17.75%. This confirms that the recognition rate of the CNN with ECOC is higher than that of the CNN with place code, in spite of the fact that the number of the outputs N =17 is much smaller than that of the CNN with place code, and that the letters or characters are surrounded with arbitrary contexts. This is due to the error-correcting power of the ECOC. For the CNN with ECOC, a received code word is formed by converting positive outputs of the CNN to 1's and negative outputs to 0's, and the received code word is decoded as the code word that has the minimum Hamming distance to the received code word.

It should be noted that in a segmentation-free OCR, as the sliding window is shifted, many images sent to the CNN do not have letters or characters at the center of the window, and it is needed to reject such images to prevent erroneous insertions and to recognize images with centered letters or characters. The decision to classify or to reject an image can be made by the CNN with ECOC according to the Hamming distance between the received code word and its closest code word. The Hamming distance is calculated as the number of mismatched bits between the received code word and a designed code word. In this paper, if the minimum Hamming distance is less than or equal to 1, then the input image is recognized as the closest code word; otherwise the input image is rejected. The decision given an image from the sliding window can be formulated as:

$$decision = \begin{cases} class\ i, & if \ \underset{i}{\operatorname{argmin}}\{D(c,C_i) \le d\} \\ reject, & if \ \underset{i}{\operatorname{argmin}}\{D(c,C_i) > d\} \end{cases} \quad (1)$$

where c is the received code word, C_i is one of the designed error correcting code words, $D(c, C_i)$ is the Hamming distance between c and C_i, and d is the predefined Hamming distance for recognition or rejection. There is a trade-off between deletion rate and insertion rate: a smaller d can result in more deletions, more rejections or less insertions.

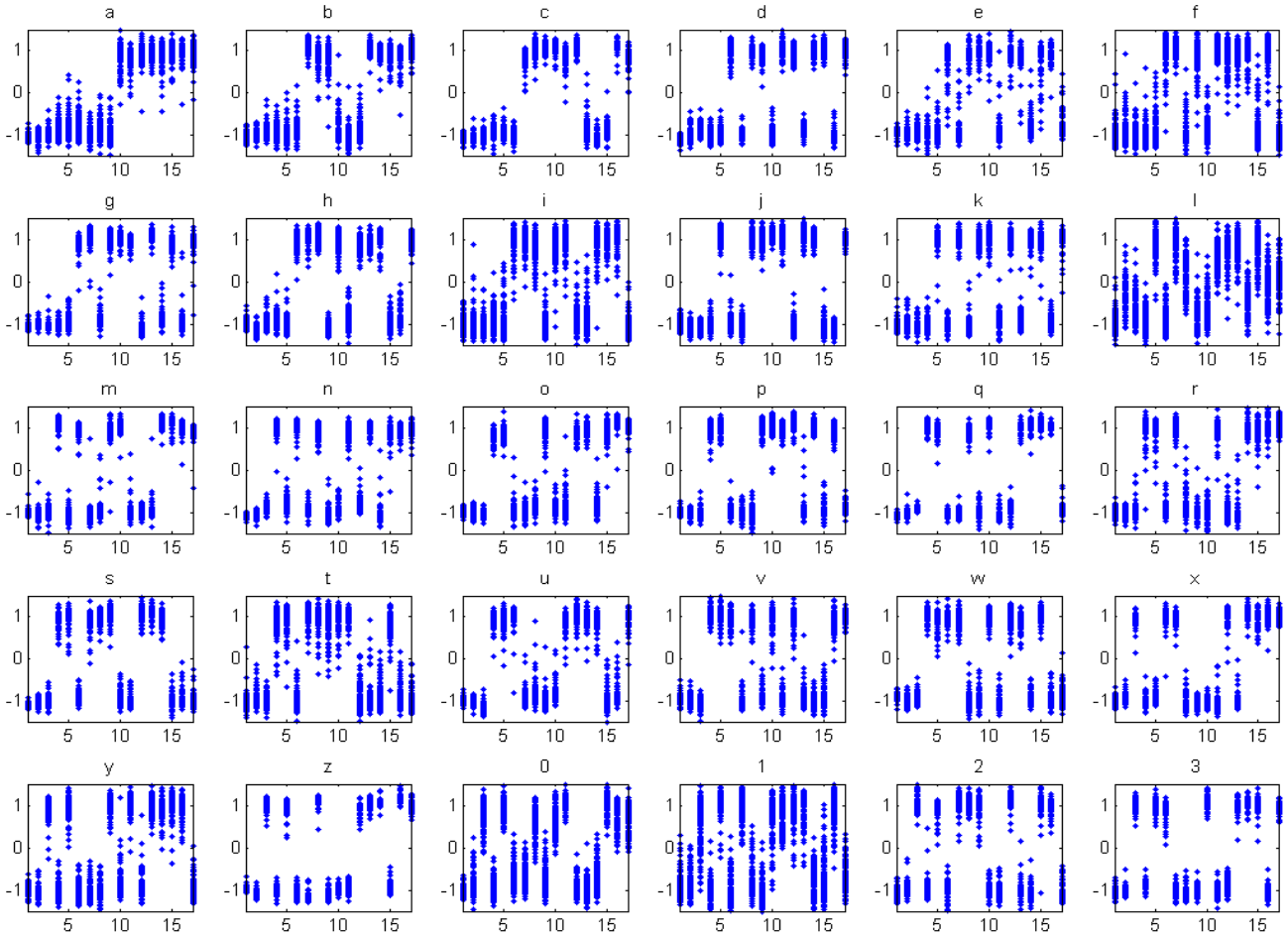

Fig. 2. The outputs of the CNN with ECOC given testing images of "a", …, "z", "0", …, "3".

5. Results and Discussions

The effects of ECOC to improve the recognition accuracy and reduce insertions for the segmentation-free OCR can be seen by comparing the recognition of text lines given by the CNN with place code with those given by the CNN with ECOC. In the experiment, d=1 is used in Equation (1), and the code words in Table I are used. As shown in Fig. 3, the CNN with place code produced many insertions even when the sliding window is centered at the gaps between letters, in addition to many substitutions when the sliding window is centered at characters. In contrast, the CNN with ECOC dramatically reduced insertions and substitutions. It is difficult to find a confidence level for the CNN with place code to reject insertions of off-centered characters and recognize centered characters. As mentioned in section 1, the CNN with place code often produces high level outputs for images of off-center characters, and it makes decisions based on the single highest output.

The mechanism for the CNN with ECOC to outperform the CNN with place code can also be explained in terms of pattern recognition, in addition to the minimum code word distance. As can be seen from Table I, the code words used in the ECOC form a binary tree. This enables the CNN with ECOC to be trained to extract N binary features from the training images such that each class can have at least D complementary binary features to be distinct from each other. One can increase the values of N and D to increase the number of distinctive features extracted by the CNN with ECOC. In contrast, the output layer of the CNN with place code acts as a one-node tree with branches as many as there are classes, and the decision is based on the single most outstanding feature, which can easily be produced from images of off-center characters, resulting in insertions.

Insertions may occur in two situations: 1) when the center of the sliding window is far from the center of a character, and 2) when the center of the sliding window is adjacent to the center of a letter (1 to 2 pixels away). For the CNN with ECOC to reduce insertions occurring in the first case, longer code words and larger minimum code distances are desired. Longer code word length N allows the CNN to extract more binary features, and a larger minimum code distance ensures more discrimination features between classes. The class features are extracted by the hidden layers of the CNN. The methods for determining the error-correcting code words with maximum minimum code distance for the pattern recognition problem is an open topic. Insertions occuring in the second case are due to the fact that the hidden layers of the LeNet5

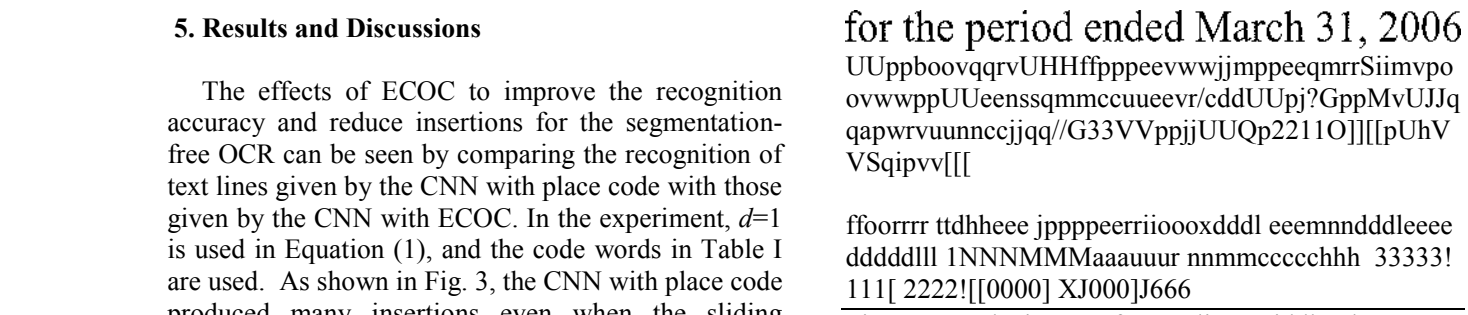

Fig. 3. Top: the image of a text line; middle: the recognition results given by the CNN with place code; bottom: the recognition results given by the CNN with ECOC.

CNN are designed for the recognition of isolated letters to be insensitive to the shifts of letters. To reduce insertions in the second case, one approach is to post-process the recognition results given by the CNN with ECOC according to the number of pixels between adjacent letters produced as demonstrated in [1]. For example, the two letters "ff" in the bottom of Fig. 3 were produced from the sliding window at two positions apart by 1 pixel, and they can be combined into one letter "f" in the post processing. Another way is to redesign the CNN such that it is more sensitive to the shifts of letters.

6. Conclusion

This paper shows that applying error-correcting coding to the convolution neural network can improve the recognition accuracy as well as reduce insertions for segmentation-free OCR. Further improvements can be made by using more realistic training data, code words with larger minimum code distances, and by modifying the hidden layers of the CNN to be sensitive to shifts of characters.

7. References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proceedings of The IEEE, VOL. 86, NO. 11, pp. 2278-2324, NOV. 1998

[2] http://yann.lecun.com/exdb/lenet/multiples.html

[3] C. Neubauer, "Evaluation of Convolutional Neural Networks for Visual Recognition", IEEE Trans. on Neural Networks, Vol. 8, No. 4, pp. 685-696, 1998.

[4] T. G. Dietterich, G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," Journal of Artificial Intelligence Research 2 (1995) 263-286.

[5] J. Zhou, H. C. Peng, and C. Y. Suen, "Data driven decomposition for multi-class classification", Pattern Recognition, 41:67-76, 2008.