Applying Error-Correcting Output Coding to Enhance Convolutional Neural Network for Target Detection and Pattern Recognition

Huiqun Deng, George Stathopoulos, and Ching Y. Suen
Centre for Pattern Recognition and Machine Intelligence
Computer Science and Software Engineering, Concordia University
Montreal, Canada
huid@ieee.org, g_statho@cse.concordia.ca, suen@encs.concordia.ca

Abstract—This paper views target detection and pattern recognition as a kind of communications problem and applies error-correcting coding to the outputs of a convolutional neural network to improve the accuracy and reliability of detection and recognition of targets. The outputs of the convolutional neural network are designed according to codewords with maximum Hamming distances. The effects of the codewords on the performance of the convolutional neural network in target detection and recognition are then investigated. Images of hand-written digits and printed English letters and symbols are used in the experiments. Results show that error-correcting output coding provides the neural network with more reliable decision rules and enables it to perform more accurate and reliable detection and recognition of targets. Moreover, our error-correcting output coding can reduce the number of neurons required, which is highly desirable in efficient implementations.

Keywords-error correcting coding; neural network; pattern recognition; target detection; optical character recognition

I. INTRODUCTION

Traditionally, place coding is used to design the target outputs of a neural network for pattern recognition: the k^{th} output of the neural network is designed to be 1 (or high) if the input signal is from class k, and all other outputs are designed to be -1 (or low). A neural network with place coding classifies the input signals based on the index of the highest output exceeding a predefined threshold. However, place coding has the following problems. First, the number of outputs of the neural network must be equal to the number of classes to be recognized or detected, which is inefficient when the number of classes is large. Second, in view of coding theory, the Hamming distance given by place coding is $D = 2$, which does not have the capacity to correct errors when there are multiple outputs exceeding a predefined threshold. Third, it has been a difficult task requiring ad hoc approaches or knowing output distributions to pre-determine an appropriate threshold for a classifier to recognize or reject the inputs accurately, and high reliability of recognition often comes at the expense of a high false rejection rate. The above problems have been encountered in the recognition of connected characters when using the convolutional neural network (CNN) with place coding and a sliding window to perform segmentation-free optical character recognition [1]. It has been found that even when the sliding window is centered at the gaps between adjacent letters, the neural network often produces high-level outputs, resulting in frequent erroneous insertions.

It is well known that in digital communications, received signals can be decoded or rejected based on the minimum Hamming distance from the received data to the designed codewords. Making recognition/rejection decisions based on minimum Hamming distance has been applied to the task of recognizing connected characters, and it has been shown that the convolution neural network (CNN) with outputs being designed according to error-correcting codewords can recognize or reject input signals more accurately than with place coding [1]. It is noted that error-correcting output coding (ECOC) has been applied to divide a multi-class problem into an ensemble of multiple binary classification problems [2][3]. The current paper applies ECOC to the outputs of a single neural network to detect or recognize targets from multiple classes, designs the codewords, and investigates the effect of codewords on the performance of the neural network. The LeNet5 CNN [4] is used as the baseline classifier because it is an efficient neural network with an architecture similar to the perceptive field found in the visual cortex, and can be trained to extract relevant topological features from the training data, without the need of designing a feature extraction method. In the following section, the method of obtaining codewords for the classifier is presented. Section 3 presents the methods of rejection or recognition based on the Hamming and Euclidean distances. Section 4 presents the experimental results obtained in the recognition of images of hand-written digits and printed characters. The conclusion is contained in section 5.

II. CODING THE NEURAL NETWORK OUTPUTS

Determining codewords for pattern recognition problems is different from that for telecommunication problems. In telecommunications, information bits (source encoding) and their error-correcting bits (channel encoding) are often considered separately. For pattern recognition, a joint source-channel coding approach can be used to combine the class information with the error-correcting. In this work, the codewords for the outputs of the neural network are obtained by searching for the candidates as follows:

1. Specify the length of codewords N, where $N \geq log_2 C$, and C is the number of classes to be recognized;
2. Specify the number of 1's in each codeword M, and the minimum Hamming distance D;
3. Put an N-bit binary string containing M 1's in an empty codebook;
4. Evaluate another N-bit binary string:
   if it contains M 1's and
   if its Hamming distances to existing codewords in the codebook $\geq D$,
   then append it to the codebook;
5. Repeat step 4 until all possible N-bit binary strings have been evaluated;
6. If the number of codewords obtained is smaller than the number of classes, then re-specify N, M, and D, and repeat steps 3-5.

By making M not equal to $N/2$, one can avoid obtaining complementary codewords which could lead the classifier to learn correlated patterns and produce correlated errors for two different classes. The n^{th} column of the codewords forms the target outputs of the n^{th} neuron of the CNN output layer in response to different input classes. Final codewords are selected from the candidates such that the numbers of 1's and 0's in each column of the codewords are about the same. As an example, the set of codewords with length N=29 assigned to the 81 characters to be detected and recognized in the task B in section IV is presented in TABLE 1.

The length of codewords given by this method can be smaller than the number of classes, and the Hamming distance varies for different pairs of codewords. In contrast, the codewords given by [5] have a length $C(C-1)/2$, which is larger than the number of classes if $C \geq 3$, and the Hamming distance between any pair of codewords is the same. The design of the codewords in this paper can reduce the number of neurons in the output layer, making the implementation of the neural network more efficient. This paper obtained 5 sets of codewords for the experiment of 10-digit recognition, and 4 sets of codewords for 81-character recognition. The codeword length N, the number of 1's in each codeword M, and the minimum and maximum Hamming distances of different sets of codewords are listed in TABLE 2. Given a set of codewords, the target outputs of the LeNet5 CNN are redesigned: the number of target outputs is N; for class c, the k^{th} target output is 1 if the k^{th} digit of the codeword is 1, otherwise it is -1. The redesigned LeNet5 CNN is called a CNN-ECOC here. The baseline LeNet5 CNN has as many outputs as the number of the classes to be detected or recognized.

TABLE 1. The codewords (N=29, D=12) for 81 characters

Codeword	Char		Char	Codeword	Char
000101111011111 11100000001000	a	010011100010101 11100111000001	1	100110001111000 11011010001010	S
000110010100110 10010101111001	b	010100001010110 00111011001011	2	100110010011101 11001100010100	T
000110010110001 01100110011011	c	010100001110010 11100000111101	3	100111011000101 00011001110010	U
000110101010101 10010011011100	d	010100100101101 00110101010101	4	101000001100101 10100111011001	V
000110110000011 11011000001111	e	010101000110100 01011110011100	5	101000110110000 01011001101101	W
000110110100011 00101011110100	f	010101011000101 10000110001111	6	101001000101010 01001110110101	X
000111011010000 11111100100001	g	010101111110010 10010001100010	7	101010001111111 00101010000100	Y
000111101101111 00000110100010	h	010110000001110 11001010110011	8	101100010100010 11111000110010	Z
001001010011001 11001000111011	i	010111010101010 01110011001000	9	101100101111110 01000001010001	!
001001101001100 01111101001001	j	011000001100011 00011101100111	A	101101000110111 10010000101100	#
001001110110110 11000010000111	k	011000010011100 01100110101101	B	101110010010000 10110011000111	$
001010010000111 11100011101010	l	011000110101111 10001001001001	C	101111000001110 10001111001000	%
001010110011010 10101101010010	m	011001001000101 11110100110010	D	110000011101111 11000000100110	&
001010111101001 01110001000110	n	011001011011010 01010101010100	E	110000011111001 00101001011010	+
001011001110000 00101011110011	o	011001110100100 00000101111110	F	110000100011010 11110001001110	/
001011011110110 00110100001010	p	011010000111001 11010010010101	G	110000110001001 10100110110011	<
001011110001100 10110000110101	q	011010100110010 10111010101000	H	110001110111110 00101100100000	>
001100001001011 11010110101001	r	011010101011101 00010000101011	I	110010000001101 01010111111000	?
001100011001100 01001011011110	s	011011111000011 01001100101000	J	110010100100010 00001111011011	@
001100100010101 01111010110001	t	011101000011011 10100011110000	K	110011001100110 11011001000001	[
001100111100100 10011110100100	u	011101010011101 00111000000110	L	110011110000000 01111010100110	]
001101100101000 11010111010010	v	011110011001001 00101111000001	M	110100101010011 10111100010000	€
001101101000010 00110010111110	w	011110101000110 01100100000111	N	110101100010011 01000010101011	¥
001110000101001 10101100101110	x	100011111100011 10100100000101	O	110110101001100 00101001101100	£
001111100011011 00000100011101	y	100100100110100 01100111100110	P	110110111011010 00000010010110	®
001111100110101 01001001001010	z	100101000011100 00110100111011	Q	111001011100000 01010010011011	×
010011011101000 10001010111100	0	100101111000000 11000011110101	R	111001100100001 11101001010100	

TABLE 2. The minimum and maximum Hamming distances of 9 sets of codewords and place coding

Tasks	Place coding	N=5 M=3	N=17 M=8	N=23 M=11	N=29 M=14	N=35 M=17
10 digits	2, 2	2, 4	6, 14	10, 16	12, 18	
81 chars	2, 2	N/A	8, 18	10, 20		14, 28

III. REJECTION AND RECOGNITION RULES

The rules for rejecting or recognizing the inputs are based on the minimum Hamming distance from the CNN output vector to the target output vectors, or the minimum Euclidean distance, respectively. The Hamming distance is used in the rejection rule, as it is an integer and hence more convenient to set the threshold level than the Euclidean distance (a continuous real number between 0 and 1). The Euclidean distance is used in the recognition rule as it is found that the recognition accuracy achieved when decoding using the minimum Euclidean distance is slightly higher than when using the minimum Hamming distance. In digital communications, decoding based on Euclidean distance is called soft decoding.

Let y_n be the n^{th} actual output of the CNN, $t_n^{(i)}$ be its n^{th} target output for class *i,* and $E(i)$ be the Euclidean distance between the output vector and the target output of class *i*. Then,

$$E(i) = \sqrt{\sum_{n=1}^{N} (y_n - t_n^{(i)})^2} \tag{1}$$

Denote $H(i)$ as the Hamming distance between the output vector and the codeword of class *i*, then

$$H(i) = \sum_{n=1}^{N} u(y_n t_n^{(i)}) \tag{2}$$

where $u(x)=1$ if $x \geq 0$, and $u(x)=0$ if $x<0$. The decision rules of the classifier can be expressed as:

$$decision = \begin{cases} class = i^* = \arg\min_i E(i), & if\ H(i^*) < d \\ reject, & if\ H(i^*) \geq d \end{cases} \tag{3}$$

where *d* is a predefined integer. By adjusting the *d* value, one can adjust the reliability of the recognition, and the rejection rate. In character recognition, if a rejection happens while the input is a target letter, the rejection is a deletion error; if the decision result is a letter while the input is not a target letter, the result is an insertion error; if misrecognition happens while the input is another target letter, the result is a substitution error. For CNN-ECOCs, as the normalized Hamming distance *d/N* increases*,* the rejection and deletion rates decrease, while the misrecognitions and false alarms increase.

The above decision rules are based on all outputs of the neural network and are more reliable than those based on the index of the single highest output. The reliability is measured as:

$$reliability = \frac{number\ of\ correctly\ recognized\ samples}{number\ of\ accepted\ samples} 100\% \tag{4}$$

IV. EXPERIMENTS AND DISCUSSIONS

To evaluate the effects of ECOC on the performance of the CNN, two different tasks are designed: *A*) the recognition of isolated hand-written digits (segmented targets); and *B*) the detection and recognition of connected characters (un-segmented targets) in TABLE 1 from text line images. For the first task, the training data consists of the 60,000 images of hand-written digits in the MNIST training dataset [6]. For the second task, a training data is constructed which consists of 106,272 images of printed characters with arbitrary left- and right- contexts synthesized from the CENPARMI isolated letter dataset. Using training images with arbitrary left- and right- contexts instead of images of isolated characters to train the neural network is crucial for the neurons to learn and then accurately detect and recognize targets within unknown contexts. Details about building the training images can be found in [1]. Each character has 4 styles (normal, bold, italic, bold italic), 36 fonts (Times New Roman, Arial, etc.), 3 sizes (18, 11, 8 point), and 3 different quantization levels. Each character sample is located at the left-right center of the 28×28 window, with its position relative to the x-line being unchanged [1].

The standard back propagation algorithm is used to train the neural network. The initial learning rate is set to 0.00005. The learning rate is reduced by a factor of 0.75 for every set of 50 epochs. The CNN-ECOCs corresponding to the 9 sets of codewords, and the baseline CNNs with N=10 and N=81 are trained from the two datasets for 800 epochs, respectively. Distortions are applied to the training images at each epoch to introduce variance in each of the training samples. For both tasks, there is no overlap between the test and training datasets.

A. Recognition of isolated hand-written digits

To compare the performances of CNN-ECOCs (N=5, 17, 23, 29, and 35) and the baseline (LeNet5 CNN with place coding), the 10,000 images of hand-written digits in the MNIST test set [6] were used. Each trained CNN-ECOC was tested for different rejection rules in Eq. (3): d = 1, 2, 3 (not for N=5), 4 (not for N=5), 5 (not for N=5), and no rejections, respectively. The baseline CNN was tested for rejection rules: the maximum output level is lower than 0.2, 0.1, 0.05, and no rejections, respectively. Their results are shown in Fig. 1. As can be seen, all CNN-ECOCs obtained better trade-offs between the reliability and the false rejection (deletion) rate than the baseline (“place”) did. All CNN-ECOCs obtained their highest reliabilities using the rejection rule Eq. (3) for d=1. CNN-ECOC with N=29 obtained the best trade-off among others. This means that CNN-ECOC 35 and CNN-ECOCs with longer codewords may need longer training time to reach their potentials, and that there may be optimal codewords for the class distributions. Designing codewords with considerations of class distributions may further improve the performance of the CNN.

B. Detection of characters within contexts

Another benefit of ECOC is to allow the CNN to detect targets within unknown contexts more accurately than the CNN without ECOC. The effect of different codewords on the performance of the CNN in detecting targets is investigated via the task of segmentation-free optical character recognition. The characters in TABLE 1 are the targets to be detected within text lines, which are scanned images normalized to a height of 28 pixels. To simulate target detection, a sliding window of 28×28 pixels is shifted

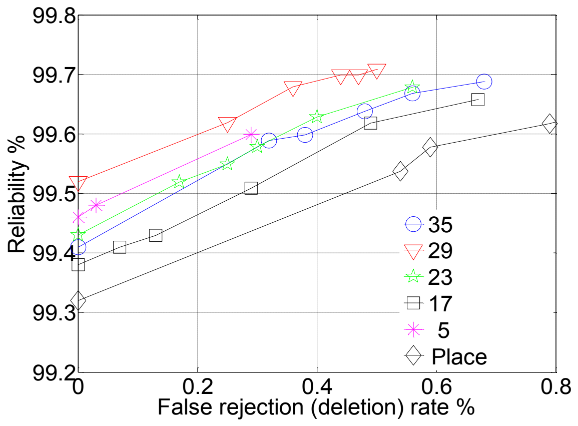

Fig.1. Reliability and false rejection (deletion) rate obtained for isolated hand-written digits.

pixel by pixel along the text line, and the image from the sliding window is sent to the CNN. The effects of ECOC on the performance of the CNN in target detection can be seen from the results obtained from text lines. Some results produced by the CNN-ECOCs and the baseline for the text line image in Fig. 2 are listed in TABLE 3. Repeated detections may happen when the sliding window is shifted around the centre of a target character, and are not considered as errors which can be removed by post-processing. As can be seen, for CNN-ECOCs, as *d/N* increases, deletion errors decrease, while insertion errors (false alarms) increase. In general target detection and recognition tasks, one can get a trade-off between the deletion rate and the false alarm rate by adjusting the values of *N* and $d=1, 2, \ldots, \lfloor D/2 \rfloor$, which is more reliable than the ad hoc search for a continuous number for the threshold. As can also be seen, the baseline CNN produced more insertion errors (false alarms) than the CNN-ECOCs did, even though it makes rejections if the minimum Hamming distance is greater than or equal to $d=D/2=1$ (i.e., it accepts the inputs only if the Hamming distance is zero). Clearly, with place coding, the CNN produced more undetectable insertion errors (false alarms) than CNN-ECOCs.

V. CONCLUSION

This paper shows that error-correcting output coding (ECOC) enhances the neural network to achieve better trade-off between high reliability and low false rejection (deletion) rate, as well as reduce false alarms compared to the neural network without ECOC. Moreover, the method of obtaining the codewords in this paper can reduce the number of neurons required, leading to more efficient implementations. Further improvements are expected by designing the codewords with due considerations of class distributions. CNN-ECOCs are also applicable in such tasks as accurate and reliable detection and recognition of face/speech/sound objects.

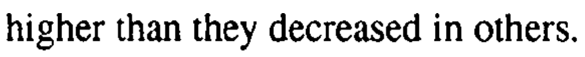
higher than they decreased in others.

Fig. 2. The text line image used in the example.

TABLE 3. The detected characters within the image

1/35	hhhhiiiqggggghhheeerrrr hhhnnnnhhheeyyyyy dddddeeccccrreeeeasseeeexdddddiiiiinnnnooo oohhhheeeerrrssss
1/29	hhhhiiigggghhhherr thhhaauunnnnthhhheeyyyyyydddddeeccccccraa aasssseeddddiiiiimnnnnooohhhheeerssss
1/23	hhhhiiiigggghhhhrrr tthhhhaaauuuummnnnnttthhhhyyyyyydddddee ccccrrraaausssseeddddiiiiimmnnnnooooothh hheerrrssss
1/17	hhhhiiigggghhhhheeerrrrrrr tthhhhmaaaaauummnnnnnthhhhceeeyyyyyy dddddeeccccrrrneeaaaausssssseeexxdddddliiiii immmnnnnoooooothhhhmeeeeurrrrssss
1/Base line	1hhhhhhmiiiiiqqjkgggggggggyhhhhhuuuweeee errrrrr tthhhhmmuuuuaaaaauuuummmnnnnnnmm 1tttUhhhhhmmuweeeeeeeqqqqqqyyyyyyyi ddddi1ukkeeeeeuxcccccmrrrrreeeeeeeeaaaaau usssssssssxweeeeeeeaxxxxdddddddf iiiiiihhhhmmmnnnnnnmm ooooooom1ttUtthhhhhhmuuuweeeeeerrrrrmssss sss

REFERENCES

[1] H. Deng, G. Stathopoulos, and C. Y. Suen, Error-Correcting Output Coding for the Convolutional Neural Network for Optical Character Recognition, 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, July, 2009, pp. 581-585.

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

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

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-Based Learning Applied to Document Recognition, *Proceedings of The IEEE,* Vol. 86, No. 11: 2278-2324, Nov. 1998.

[5] L. I. Kuncheva, Using diversity measures for generating error correcting output codes in classifier ensembles *Pattern Recognition Letters* 26: 83-90, 2005.

[6] http://yann.lecun.com/exdb/mnist/