A Novel Scheme for Implementation of the Scanning ntuple Classifier in a Constrained Environment

Sanaul Hoque, Michael Fairhurst

A Novel Scheme for Implementation of the Scanning ntuple Classifier in a Constrained Environment

Sanaul Hoque, Michael Fairhurst
Department of Electronics, University of Kent,
Canterbury, Kent, United Kingdom.
E-Mail: {S.Hoque; M.C.Fairhurst}@kent.ac.uk

Abstract

The scanning ntuple classifier is an efficient and accurate classifier for handwriting recognition. One of the major difficulties in implementing this scheme is its demand for a very large memory space, thus making it unsuitable for resource constrained systems such as embedded applications. This paper proposes some modifications to the basic sntuple algorithm which eliminates the necessity of normalizing the chain-code length, by adjusting the memory cell increments as an inverse function the chain length. The resulting system performance is shown to be superior to the standard sntuple configuration in both speed and accuracy when smaller and fewer sntuples are used, a configuration which also reduces the demand for memory.

Keywords: sntuple, handwriting analysis, OCR.

1. Introduction

Automatic understanding, processing and recognition of document images has witnessed rapid progress in recent years and has entered a completely new era. Industry is employing more and more automated systems based on pattern recognition and computer vision techniques, which are revolutionizing the way real life problems are solved in commercial environments. Typical current application domains include automated bank-cheque processing [1], automated character recognition [2], generic document processing [3], voucher processing [4], direct handwriting recognition [5], signature verification [6], and many others. In such practical applications there are two parameters of particular general interest. The first is the attainable system performance in terms of recognition or verification, and the second is the cost of the system in terms of execution speed or throughput, memory requirements, etc. Before adopting any particular implementation, both aspects of performance must be given due consideration. It is extremely important to devise a system which is accurate, but at the same time, the processing requirements must be constrained to make the adoption of such an automated system viable in an intended operational environment.

Weightless memory network classifiers (also known as n-tuple classifiers) have a long history as particularly simple and manageable but very fast classification structures for image recognition [7]. The scanning n-tuple (or simply *sntuple*) classifier introduced by Lucas *et al.* as a statistical-syntactic method is a variant of the n-tuple system and has attracted considerable research attention because of its high accuracy and speed in typical OCR applications [8]. A large number of publications can be found in the public domain focusing on further enhancement of the sntuple classifier performance (see, for example, [9, 10, 11, 12, 13]).

The focus of this paper is a handwriting recognition application using the scanning n-tuple classification algorithm [8, 14, 15]. In this paper we propose a modification to the original sn-tuple algorithm, reducing the computational cost significantly for certain operating conditions. This makes the algorithm more readily implementible, particularly in resource-constrained embedded systems applications.

2. The Scanning n-tuple classifier

The scanning n-tuple (or simply *sntuple*) classifier has been introduced by Lucas *et al.* as a statistical-syntactic method for high performance OCR applications [8]. This is a variant of the basic n-tuple system [7] except that it operates on the chain coded representation of the handwritten characters to be processed.

In the conventional n-tuple scheme, a 2-dimensional binary image is decomposed into many sets of n-points (known as n-tuples). Each such tuple is associated with 2^n memory locations (for any given class) which keep count of the occurrences of every possible bit pattern occurring during a training exercise. The process remains essentially the same for grey-scaled images except that σ^n memory locations are required per tuple, where σ is the grey-level resolution of the image. In subsequent use, contents of the memory locations corresponding to the n-tuple bit patterns found in the test image are summed. The label of the class producing the highest score is assigned to the unknown pattern.

Unlike conventional n-tuple methods, the scanning ntuple system operates on a chain-coded representation of the image. A mask is used as a template to determine the

sampled points of the n-tuple. These points are, usually, uniformly spaced from each other. Thus, by varying the spacings, many different masks can be created. Each of these masks is then applied repetitively to all points of the chain code string, in order to sample the entire pattern. Since a chain code descriptor can be associated with any of eight possible values (0 through 7), 8^n memory locations should be provided for each mask. Here, n is the size of the mask.

During the training cycle, for each position of a mask, the corresponding memory cell is incremented by **one**. During the test phase, all the contents of the relevant memory cells are added and the class generating the highest score wins. More detail of the algorithm (as well as appropriate pseudo-code), can be found in [14].

An online character can readily be converted to a chain coded format simply by quantizing the pen movement to 8-direction codes. An off-line character image can also be transformed automatically by tracing its contour (see [16, 17] for details) and expressing the path in chain codes. Figure 1 illustrates the extraction of chain codes from an offline character image. The chain coded strings generated in this way differ significantly in their length. The length depends on the pattern class as well as writing style, degree of slant, image quality etc, and Figure 2 shows typical lengths of the chain-coded string for various characters. Since the masks in the sntuple scan over these variable length strings, a variable number of tuples can be fitted onto them and as such, image classes with shorter chains are adversely affected. Lucas *et al* overcame this problem by normalizing the extracted chain-coded strings to a fixed length before using them for training or testing. In this paper, we propose an alternative approach to overcome this problem.

3. Proposed modification

It has already been pointed out that the length of the chain coded string varies significantly, and characters with shorter strings are thus adversely affected due to the existence of a smaller number of available mask positions over the string. The remedy used by Lucas *et al* is to scale (usually by elongating) all the chains to a pre-specified fixed length, a problem which is not only time-consuming, but also invariably deforms the character shape.

In this paper we propose a scheme which does not involve normalization of the chain coded string. Instead, during the training phase, the system increments the memory locations, when addressed, by a factor of $\frac{k}{||y||}$ (where, k= any constant, $||y||$=length of chain coded string). Thus it offsets the negative impact of shorter scan space by storing higher values in the corresponding memory cells. Table 2 gives the pseudocode of the training cycle. The testing phase remains identical to the standard sntuple algorithm but excludes the chain-length normalization.

The proposed scheme not only removes the processing module required for length normalization, but also saves further processing time by permitting the scan to be carried out on the short chains. The consequent benefits that can be achieved are discussed in the following sections.

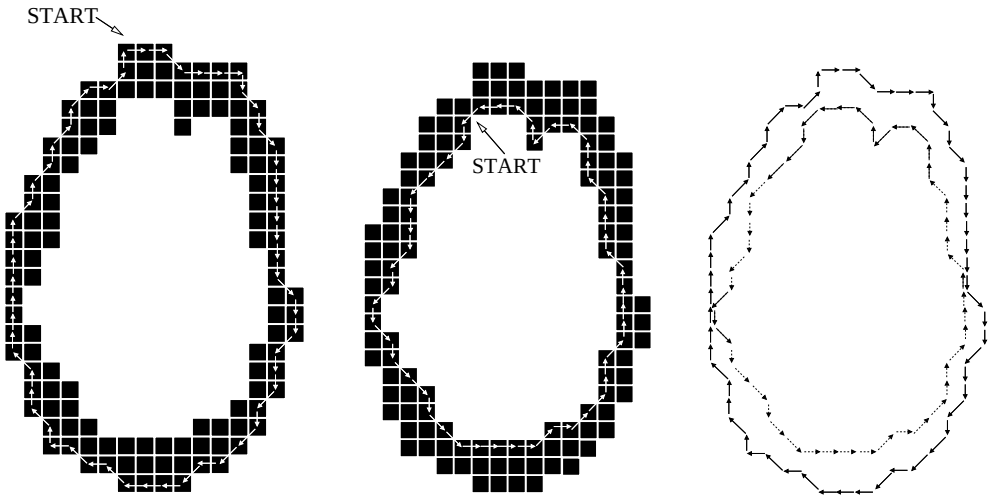
4. Experimental Results

In order to investigate the impact of the proposed modifications, experiments were conducted for the classification of pre-segmented offline handwritten characters consisting of digits and uppercase letters extracted from the NIST database [18]. No distinction is made between '0'/'O' and '1'/'I' character pairs. There are 30899 images for training and 20667 for testing, each of resolution 32×32 pixels. Figure 3 illustrates one sample from each character class in the database. Many different sntuple masks were generated (all with uniformly spaced bits). Although it is possible to mix differently sized masks in any one implementation, for simplicity only masks of similar size were used in the experiments reported here. In the implementations of the standard sn-tuple algorithm, extracted chaincoded strings were normalized to a fixed length of 200.

Table 3 presents a comparison of recognition accuracies between the standard sntuple configuration and the proposed scheme during digit-only classifications. The figures presented are the mean accuracy over several runs using a number m of different n-bit sntuple masks. As the memory space required is an exponential function of n, experiments were conducted for $n \leq 5$. For $n > 5$, the memory requirements become excessively high and we excluded them from this investigation. Table 4 presents a similar comparison of recognition accuracies when alphanumeric characters (ie, 34 classes) are used. It is obvious from these two tables that for large values of n and m (ie, when many large masks are used), the original sntuple algorithm outperforms the proposed scheme. But for smaller n and m, the proposed scheme shows superior recognition rates. For example, in the 10-class scenario, for $n = 4,\ \ m = 3$, the proposed scheme achieves accuracy of 84.46% whereas the conventional algorithm achieved 82.80%. Performance improvements are also similar for the 34-class task.

It has already been pointed out that, by varying the construction of the sntuple masks, many sntuple classifiers can be implemented for the same values of n and m. The performance of these individual implementations can vary significantly and hence, in Table 3 and Table 4, we present the average recognition accuracies achieved over several possible implementations. It is not impossible, subject to availability of sufficient training samples, to find the optimum implementation (ie, an implementation that gives the best accuracy for a given $n,\ m$) for a given task domain. Table 5 presents the comparative accuracies when we picked the best achieved recognition rates for any given $n,\ m$. Again, it is evident that for smaller $n,\ m$, the proposed scheme outperforms the original sntuple configuration. Although not explicitly discussed in detail here, it is also observed that the variances from different implementations were much lower in the proposed algorithm.

One of the positive aspects of the sntuple scheme is its



Outer contour:

00700067676666666766566565545444 3. . .2221212121012

Inner contour:

565556665656767767700001011221222232223223452344

(d)

Figure 1. Extraction of chain-code from an off-line image by contour tracing; (a) outer contour tracing, (b) inner contour tracing, (c) contour profile, (d) resulting chain-coded descriptor strings of the character.

Table 1. Training algorithm for the original sntuple

Step 1: Extract chain-code representation
Normalize chain to a fixed length
Step 2: Initialise all n-tuples
Step 3: Train all n-tuples on all training patterns
For each class c
For each chain coded string y
For each sn-tuple n_{cj}
For offset $o := 0$ to $\|\|y\|\|$
Identify the corresponding memory location
Increment memory content by 1
Step 4: Convert frequency counts to log probabilities

Table 2. Training algorithm for the proposed scheme

Step 1: Extract chain-code representation
{Normalize chain to a fixed length} $\leftarrow$ **No longer required**
Step 2: Initialise all n-tuples
Step 3: Train all n-tuples on all training patterns
For each class c
For each chain coded string y
For each sn-tuple n_{cj}
For offset $o := 0$ to $\|\|y\|\|$
Identify the corresponding memory location
Increment memory content by $\frac{\mathbf{k}}{\|\|\mathbf{y}\|\|}$ $\leftarrow$ **Proposed modification**
Step 4: Convert frequency counts to log probabilities

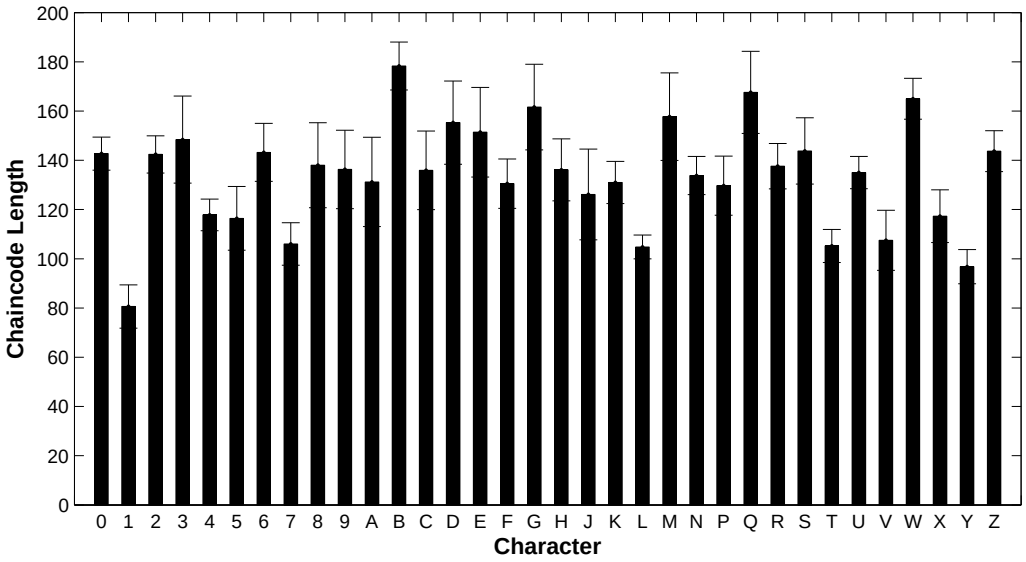


Figure 2. Mean length of extracted chain codes of the NIST characters

Figure 3. Some sample images from the NIST database used in this analysis

Table 3. Mean recognition accuracies (in %) as tested on Numerals (10-class)

Original sntuple							Proposed Scheme						
	m							m					
n	1	2	3	4	5	6	n	1	2	3	4	5	6
1	39.84	-	-	-	-	-	1	40.54	-	-	-	-	-
2	56.09	61.33	65.08	67.76	69.73	71.28	2	58.66	64.50	67.60	69.45	70.41	71.19
3	64.99	71.95	76.37	79.45	81.75	83.32	3	68.87	75.64	78.89	80.60	81.52	81.95
4	70.08	78.40	82.80	85.45	87.07	88.12	4	74.87	81.84	84.46	85.59	85.91	85.92
5	74.08	82.66	86.73	88.68	89.80	90.51	5	78.74	85.25	87.18	87.73	87.83	87.67

Table 4. Mean recognition accuracies (in %) as tested on Alpha-numerics (34-class)

Original sntuple							Proposed Scheme						
	m							m					
n	1	2	3	4	5	6	n	1	2	3	4	5	6
1	27.18	-	-	-	-	-	1	28.19	-	-	-	-	-
2	47.72	54.35	58.77	61.84	63.96	65.62	2	48.69	56.49	60.51	62.88	64.37	65.37
3	58.29	66.54	70.83	73.64	75.66	77.06	3	60.37	68.65	72.19	74.00	74.88	75.32
4	63.89	72.72	76.82	79.28	80.74	81.66	4	66.82	74.62	77.38	78.47	78.81	78.93
5	67.56	76.52	80.29	82.21	83.22	83.84	5	70.73	77.85	79.93	80.59	80.69	80.59

Table 5. Peak recognition accuracies (in %) as tested on Numerals (10-class)

Original sntuple							Proposed Scheme						
	m							m					
n	1	2	3	4	5	6	n	1	2	3	4	5	6
1	39.85	-	-	-	-	-	1	40.54	-	-	-	-	-
2	60.53	66.52	70.30	72.77	74.42	75.77	2	63.64	68.90	71.08	72.21	72.68	73.14
3	70.99	77.85	82.46	85.17	86.66	87.46	3	75.51	80.51	82.46	82.87	82.88	82.93
4	77.24	84.74	87.84	89.50	90.20	90.47	4	81.50	85.71	86.75	86.73	86.76	86.61
5	82.13	88.42	90.66	91.30	91.55	91.73	5	84.51	88.03	88.31	88.35	88.61	88.58

capability for achieving very fast classification times. The proposed scheme eliminates the necessity of normalizing the length of the chaincoded string. Also, some additional time is saved because the scanning is performed on the smaller chain code strings that are extracted. Thus, the proposed scheme can perform much faster than the standard sntuple classifier. Table 6 presents the throughputs (in characters per second) achieved when the algorithms were tested on a SUN Workstation (SUN FIRE V440 with 4GB RAM). It is evident that, on average, 40-50% additional characters can be classified when using the proposed scheme in comparison with the original formulation.

5. Conclusion

This paper has introduced a modified scheme for the implementation of the sntuple classifier. In the original implementation, the extracted chain codes need to be length normalized. Otherwise, characters producing shorter chaincode strings were almost always classified incorrectly. In the scheme proposed here, the memory locations associated with each sntuple were incremented by a variable amount which is chosen as an inverse function of the chain length. This scheme therefore overcomes the particularly adverse effect of variable chain lengths and hence eliminates the necessity of chain code string length normalization.

The experimental results presented in the paper reveal that the proposed scheme outperforms the original sntuple by a significant margin when relatively fewer and smaller size sntuples are being used. This phenomenon is observed in both the 10-class and 34-class task domains.

The modified algorithm also works much faster as a direct result of the elimination of the feature normalization module and also because it scans over short chains, with 40-50% additional improvements in throughput being achieved.

One of the bottlenecks in the implementation of an sntuple algorithm is its requirement for a very large memory space. If a number m of different n-bit sntuples are adopted in a C-class task domain, the total memory space required is given as:

$$S = m8^n C.$$

It is therefore apparent that an ability to operate at smaller n, m can reduce the memory demand for a system significantly. Embedded applications, which typically demand very constrained resources, may find the proposed scheme more suitable than the original sntuple algorithm.

References

[1] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price, and J.C. Simon, "The a2ia intercheque system: Courtesy amount and legal amount recognition for french checks," *Int. J. Pattern Recognition and Artificial Intelligence*, vol. 11, no. 4, pp. 505–548, 1997.

[2] M. C. Fairhurst and M. S. Hoque, "Moving window classifier: Approach to off-line image recognition," *Electronics Letters*, vol. 36, no. 7, pp. 628–630, March 2000.

[3] Y. Y. Tang, S.W. Lee, and C.Y. Suen, "Automatic document processing: A survey," *Pattern Recognition*, vol. 29, 1996.

[4] J. Mao, R. Lorie, and K. Mohiuddin, "A system for automatically reading iata flight coupons," in *Proceedings of 4th International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 153–157.

Table 6. Classification speed in characters per sec (for n=5)

Original sntuple							Proposed Scheme						
no of	m						no of	m					
class	1	2	3	4	5	6	class	1	2	3	4	5	6
10	4000	2222	1538	1176	952	800	10	6667	2857	2222	1818	1389	1176
34	1378	590	413	225	254	202	34	1879	752	559	323	350	285

[5] L. Vuurpijl and L. Schomaker, "Finding structures in diversity: A hierarchical clustering method for categorization of allographs in handwriting," in *Proceedings of 4th International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 387–393.

[6] L.L. Lee, T. Berger, and E. Aviczer, "Reliable online human signature verification system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 643–647, 1996.

[7] M. C. Fairhurst and T. J. Stonham, "A classification system for alpha-numeric characters based on learning network techniques," *Digital Processes*, vol. 2, pp. 321–329, 1976.

[8] S. Lucas and A. Amiri, "Recognition of chain-coded handwritten character images with scanning n-tuple method," *Electronics Letters*, vol. 31, no. 24, pp. 2088–2089, November 1995.

[9] E. H. Ratzlaff, "A scanning n-tuple classifier for online recognition of handwritten digits," in *Proceedings of 6th International Conference on Document Analysis and Recognition*, Seattle, Washington, USA, September 2001, pp. 18–22.

[10] G. Tambouratzis, "Improving the classification accuracy of the scanning n-tuple method," in *Proceedings of 15th IAPR International Conference on Pattern Recognition)*, Barcelona, Spain, 3-8 September 2000, pp. 1050–1053.

[11] G. Tambouratzis, "Improving the clustering performance of scanning n-tuple method by using self-supervised algorithms to introduce subclasses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 722–733, 2002.

[12] S. Hoque, K. Sirlantzis, and M.C. Fairhurst, "Bit plane decomposition and the scanning n-tuple classifier," in *Proceedings of 8th International Workshop on Frontiers in Handwriting Recognition)*, Naigara-on-the-lake, Canada, August 2002, pp. 207–211.

[13] S. Hoque, M.C. Fairhurst, and R.M. Guest, "The effect of inhibition-compensation learning scheme on n-tuple based classifier performance," in *Proceedings of 16th International Conference on Pattern Recognition*, Quebec City, Canada, August 2002, pp. 452–455.

[14] S. Lucas and A. Amiri, "Statistical syntactic methods for high performance ocr," *IEE Proceedings Vision, Image and Signal Processing*, vol. 143, no. 1, pp. 23–30, February 1996.

[15] S. M. Lucas, "Improving scanning n-tuple classifiers by pre-transforming training data," in *Proceedings of International Workshop on Frontiers in Handwriting Recognition-V*, ESSEX, UK, 1996, pp. 143–146.

[16] G. R. Wilson and B. G. Batchelor, "Algorithm for forming relationships between objects in a scene," *IEE Proceedings Pt. E*, vol. 137, no. 2, pp. 151–153, March 1990.

[17] G. R. Wilson, "Properties of contour codes," *IEE Proceedings Vision Image and Signal Processing*, vol. 144, no. 3, pp. 145–150, June 1997.

[18] NIST Special Databases 1-3, 6-8, 19, 20, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.