Optimizing Parameters in the Layered Search Space

Gleidson Pegoretti da Silva[1], Yen Kaow Ng[1], Yiping Yang[2], Bilan Zhu[1], Masaki Nakagawa[1]

[1] Tokyo University of Agriculture and Technology, Koganei-shi, 184-8588, Japan
E-mail:50007834304@st.tuat.ac.jp, {kalngyk,zhubilan,nakagawa}@cc.tuat.ac.jp

[2] Toshiba Solutions Corporation Embedded Solutions Division, Japan
E-mail:yiping_yang@msn.com

Abstract: This paper reports optimization efforts on a layered search space method aimed at accelerating the recognition of a large prototype set. The layered search space method classifies similar prototypes into clusters. Representative prototypes, each for one of these clusters, are then selected and further classified into higher-level clusters, and so on. In finding a prototype, we first identify the highest-level clusters where the prototype may be found, then proceed to identify the most likely sub-clusters within these clusters, and so on. Finally, we match the input with the prototypes in the identified lowest level clusters. Increasing layers will decrease the number of prototypes to be matched, but the precision of candidate selection will decrease and overhead will increase. Hence there are several parameters that one needs to adjust for the method to perform optimally. Most importantly, there is an optimal number of layers that accelerates the recognition without compromising the recognition rate. We used two efficient methods to approximately identify this number. Both the methods show that having two layers achieves this optimality for the recognition of handwritten Japanese characters.

Key Words: large category set, search space, pivot, candidate selection, character recognition

1. INTRODUCTION

The task of recognizing letters in a very large alphabet such as Chinese, Japanese and Korean faces problems with not only the recognition rate but also the recognition speed. Chinese, Japanese and Korean alphabets have thousands of characters, so their recognition takes significantly more time than the recognition of say, the Latin alphabet or numerical characters. While advances in the speed of computers have made the recognition of these alphabets reasonably fast, we are still very in need of faster character recognizers. These faster recognizers could be run on devices with less processing power like mobile phones or PDAs, or be used in more elaborated applications that demand better response time. They would also allow us to run multiple recognizers simultaneously to obtain consensus results which may be more accurate.

A general method for improving recognition speed is to perform a coarse classification (pre-classification or candidate selection) prior to fine classification [1, 2]. The coarse classification typically uses simpler classification algorithms or fewer features in order to achieve better speed than the fine classification. We distinguish between methods where the entire candidate selection process is performed during the recognition process [3, 4, 5, 6], and methods where the prototypes are pre-organized prior to a search [10, 11]. We call the former *dynamic* approaches, and the latter *static* approaches.

We earlier introduced a static approach called *Structuring Search Space (SSS)*, which works by pre-clustering prototypes into clusters and obtaining a representative (called pivot) for each cluster [7, 8, 9]. When given an input pattern, only up to a number of clusters where their representatives are close to the input pattern are selected for further searches. We further extend SSS, by clustering the clusters' representatives to realize a second layer of clusters and cluster representatives. When given an input pattern, the pattern is first compared with the representatives of the second layer clusters, where a fixed number of close enough representatives are then selected and the search is continued with the clusters they represent. If we further cluster the representatives of the second layer clusters, we can obtain a third layer, and so on. We call this a *Layered Search Space (LSS)* (see Fig. 1).

While increasing layers will decrease the number of prototypes to be matched, the precision of candidate selection will decrease and overhead will increase. In this paper, we use two distinct techniques to efficiently identify the optimal number of layers for the current task, with the corresponding numbers of clusters to create and to select during search, in each layer.

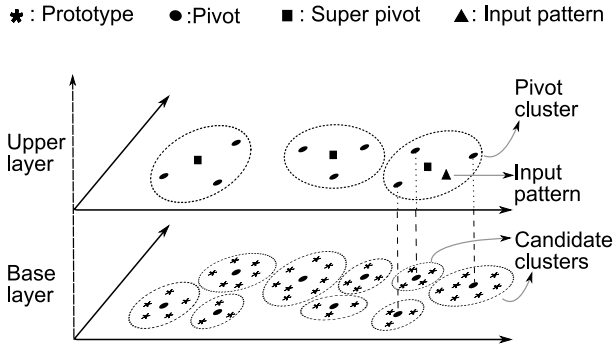

Figure 1: Conceptual figure of the LSS

2 LSS SYSTEM DESCRIPTION

As a character recognizer we use a two-stage recognizer (coarse and fine). for handwritten Japanese characters [12]. The recognizer represents each scanned image of a character pattern as a 256-dimensional feature vector. It scales every input pattern to a 64x64 grid by non-linear normalization. Then, it decomposes the normalized image into 4 contour patterns representing directional features of the 4 main orientations. Finally, it extracts a 64-dimensional feature vector for each contour pattern from the convolution with a blurring mask (Gaussian filter).

The *coarse classification* selects 40 candidates with the shortest Euclidean distances between the categories' prototype and an input pattern, while the *fine classification* employs a modified quadratic discriminant function (MQDF2) [13] to select the best prototype from these candidates.

2.1 Layered Search Spaces

Figure 1 shows a conceptual figure of a two-layer LSS. For simplicity the feature space is drawn in only two dimensions, although a typical feature space for large character set recognition takes 256 or 512 dimensions. In the base layer, we first cluster the prototypes by the similarity of their features. The clustering is performed using k-means on Euclidean distance. A representative is then selected for each cluster, by taking the centroid of the prototypes of the cluster. We call these representatives *pivots*. These pivots are then clustered to form pivot clusters in the upper-layer. Again a representative is selected from each pivot cluster, by taking the centroid of the pivots. We call these representatives *super-pivots*. (Note that the prototypes in the base layer may themselves be centroids from an even lower layer.) Given an input pattern, the search proceeds in the following steps:

1. It is matched with all the super-pivots, and the super-pivots "close enough" to it are selected.
2. It is matched with all the pivots belonging to the pivot clusters represented by the above selected super-pivots, and the pivots "close enough" to it are selected.
3. It is matched with all the prototypes within the candidate clusters represented by the selected pivots, and the closest prototypes are selected.

Thus, the number of prototypes compared with the input pattern is reduced. A simple repetition of the clustering constructs a search space with more than two layers.

There are a few parameters in LSS which we need to consider:

1. The number of layers to construct.

2. For each layer, a number (n) of clusters to construct.

3. For each layer, two numbers (l and m) which define how "close" the representatives in the layer have to be to the input pattern, in order for the clusters they represent to be selected. The selection method uses l and m in the following way:

First, the method selects all the representatives within distance $m * d_{min}$ to the input pattern, where d_{min} is the distance of the closest representative to the input pattern. Then from these representatives, the method selects the l closest representatives to the input pattern.

3 OPTIMIZING LSS PARAMETERS

Given a number of layers to construct, we want to find n, l and m for each layer, such that the recognition time is optimized without compromising recognition rate. We optimize the LSS with respect to the HP-JEITA database. The database contains 580 persons' handwritings for 3214 categories of digits, Latin alphabet characters, symbols, hiragana (a set of phonetic Japanese characters), katakana (another set of phonetic Japanese characters) and Japanese Kanji characters of Chinese origin.

Even for just two layers, to find the optimal values for the six parameters on the database would take a very long time. Hence we use two efficient heuristical methods to find the parameters, and compare their results.

3.1 Data sets and original performance of recognizer

We first choose from the database 543 persons' patterns, that is, removing 37 imperfect collections. The database is then split into 5 testing sets (Te_1–Te_5) of 100 persons' patterns each. When Te_i ($1 \leq i \leq 5$) is used as a testing set, we use the remaining 443 persons' patterns to train the original recognizer. Table 1 shows the performance of the original recognizer on each set after training. The coarse classification (CC) rate is the percentage of times the input character is included in the set of prototypes selected in coarse classification. The whole recognition (WR) rate is the percentage of times the top prototype returned by the recognizer matches the input character. The recognition time is the total CPU time used in recognition.

Table 1: Performance of the original recognizer (CC=Coarse Classification; WR=Whole Recognition)

Test Set	rate(%)		time(ms)	
	CC	WR	CC	WR
Te_1	99.3	97.1	2.03	3.18
Te_2	99.3	96.4	2.03	3.19
Te_3	98.8	96.0	2.03	3.18
Te_4	98.8	96.0	2.02	3.18
Te_5	98.9	96.1	2.03	3.19
Avg.	99.0	96.3	2.03	3.18

3.2 Sequential optimization

In the first method we first optimize the n, l and m parameters in the base layer, and then optimize the n, l and m parameters in the next upper layer, and so on. We call this method *sequential optimization (SO)*. The merit of this method is that the process and intermediate result of optimization is visible. It can also be performed relatively quickly.

The optimization of each layer is performed empirically. That is, we test the recognition speed and precision using a range of values for n, l and m, taken at fixed intervals.

For each n and each Te_i ($1 \leq i \leq 5$), we obtain the l and m values which lead to the optimal speed in recognizing Te_i, without compromising the original recognition rate. The overall optimal recognition speed, l and m, at n is taken to be the average of five values obtained, each from one of the testing sets. For each layer, we use the n, l and m values which lead to the largest acceleration in recognition.

For the base layer, our tests show that the best speed-up is obtained when n is within the range of 100 to 400. Within the range, the optimal speed-ups are similar. Since having more pivots to cluster in the 2nd layer would help the clustering to be more refined (that is, members within the same cluster would bear more resemblance), we let $n = 400$ in the base layer and use the corresponding optimal values of l and m of each test set for the base layer. For the second layer, our tests show that the best speed-up is achieved when $n = 40$. At these parameters, the coarse recognition rates are preserved to up to a decimal place (see Table 2).

Table 2: Performance of 2-layer LSS with parameters from SO.

Test Set	base layer ($n = 400$)		2nd layer ($n = 40$)		CC rate (%)	CC time (ms)	WR rate (%)	WR time (ms)
	l	m	l	m				
Te_1	131	1.78	29	1.90	99.3	0.99	98.2	1.48
Te_2	132	1.78	29	1.93	99.3	0.98	98.3	1.48
Te_3	135	1.80	30	1.90	98.8	0.98	98.4	1.48
Te_4	127	1.78	29	1.88	98.8	0.98	98.4	1.47
Te_5	125	1.75	28	1.88	98.9	0.97	98.3	1.47
Avg	130	1.78	29	1.90	99.0	0.98	98.3	1.48

Since it is possible that some non-optimal values for a lower layer may result in better overall improvement at an upper layer, the parameters obtained using SO is not necessarily optimal. Hence we confirm these parameters with a second method.

3.3 Genetic algorithm

In the second method we first set the number of layers to construct, and then employ *genetic algorithm (GA)* to find the combination of n, l and m values for all the layers, which leads to the highest average speed in recognizing the testing set without losing recognition rate.

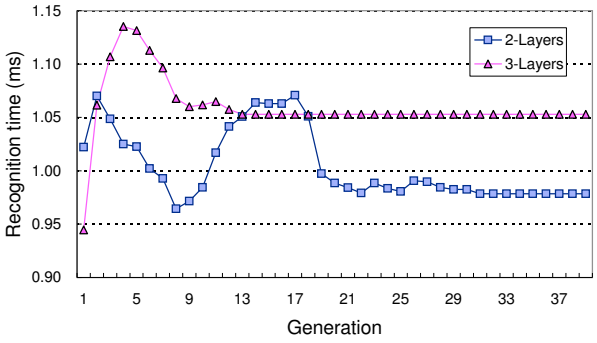

Figure 2: Recognition time of two-layered and three-layered LSS

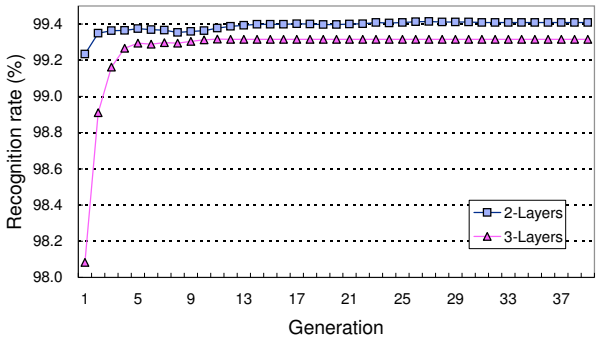

Figure 3: Recognition rate of two-layered and three-layered LSS

As testing test we use only a portion of Te_1 since the test is time-consuming. Each parameter set is considered an individual of a population. We start the population with 400 randomly created individuals. New generations are created by applying crossover and mutation over the selected individuals. At each iteration 10 best individuals are selected. Selection is performed using a fitness function which gives a weight of 0.98 to the recognition rate and 0.02 to the recognition time.

Figure 2 and 3 show the average of the 10 best individual performances during 40 generations. After the convergence of recognition time has been achieved we observe (Fig. 2) a significant difference of 0.08 milliseconds between the two and three layered LSS.

4 RESULTS

Table 3 shows the optimal n, m and l values of a two-layered LSS obtained using SO and GA respectively. Similar performances are achieved using both methods. For the 2nd layer, the parameters found are very similar. For the base layer, the n and l parameters found are significantly different. However, the performance of the recognizer is insensitive to these parame-

ters. Our previous work demonstrated that any value for n from 100 to 400 would result in the optimal performance, while the optimal values for l vary according to n, where $l/n = 0.325$ when $n = 400$ and $l/n \approx 0.46$ when $n = 128$ [8]. Hence the results from SO and GA are consistent. This suggests that the parameters obtained using both methods are indeed optimal.

Table 3: Parameter set and average performance of a two-layer LSS.

	Base layer			2nd-layer			rate(%)	
Method	n	l	m	n	l	m	CC	WR
SO	400	130	1.78	40	29	1.89	99.0	98.3
GA	128	56	2.07	42	32	2.18	99.0	98.3

When a third layer is added in the SO method, no third layer n, l, m values which further speed up the recognition time could be found. Similarly, when three layers are used, GA was unable to identify any set of parameters which results in performance as good as the optimal when only two layers are used.

5 CONCLUSION

In this paper we presented a layered search space (LSS) method which naturally extends the structuring search space (SSS) method we introduced previously. We used two efficient methods to find the optimal number of layers in LSS. In both methods we obtained similar results, which show that the two-layer LSS is optimal for our task of Japanese character recognition.

ACKNOWLEDGEMENTS

We thank Professor Yamamoto and his colleagues for providing us with the database ETL-9 and HP-JEITA. This work is supported by the R&D fund for "development of pen & paper-based user interaction" under Japan Science and Technology Agency.

References

[1] S. Mori, K. Yamamoto, and M. Yasuda, "Research on machine recognition of handprinted characters", IEEE PAMI, vol.6, pp.386–405, 1984.

[2] T.H. Hildebrandt and W.T. Liu, "Optical recognition of handwritten chinese characters: Advances since 1980", Pattern Recognition, vol.26, no.2, pp.205–225, 1993.

[3] T. Wakabayashi, Y. Deng, S. Tsuruoka, F. Kimura, and Y. Miyake, "Accuracy improvement by nonlinear normalization and feature compression in handwritten chinese character recognition", IEICE PRU, vol.95, no.43, pp.1–8, 1995.

[4] N. Sun, M. Abe, and Y. Nemoto, "A handwritten character recognition system by using improved directional element feature and subspace method", J. IEICE, vol.78, no.6, pp.922–930, 1995.

[5] T. Kumamoto, K. Toraichi, T. Horiuchi, K. Yamamoto, and H. Yamada, "On speeding candidate selection in handprinted chinese character recognition", Pattern Recognition, vol.24, no.8, pp.793–799, 1991.

[6] C.H. Tung, H.J. Lee, and J.Y. Tsai, "Multi-stage precandidate selection in handwritten chinese character recognition systems", Pattern Recognition, vol.27, no.8, pp.1093–1102, 1994.

[7] Y.P. Yang, O. Velek, and M. Nakagawa, "Accelerating large character set recognition using pivots", Proc. 7th ICDAR, Edinburgh, UK, pp.262–267, 2003.

[8] Y. Yang, B. Zhu, and M. Nakagawa, "Structuring search space for accelerating large set character recognition", IEICE Transactions, vol.88-D, no.8, pp.262–267, 2005.

[9] Y. Yang and M. Nakagawa, "Improving the structuring search space method for accelerating large set character recognition", Proc. 9th IWFHR, Tokyo, Japan, pp.251–256, 2004.

[10] Y.H. Tseng, C.C. Kuo, and H.J. Lee, "Speeding up chinese character recognition in an automatic document reading system", Pattern Recognition, vol.31, no.11, pp.1601–1612, Nov. 1998.

[11] K. Fujimoto, H. Kamada, and K. Kurokawa, "Fast, precise pre-classification method using projections of feature regions", Technical report of IEICE. PRMU, vol.97, no.558, pp.25–32, 1998.

[12] J. Tsukumo and H. Tanaka, "Classification of handprinted chinese characters using non-linear normalization and correlation methods", ICPR, pp.168–171, 1988.

[13] F. Kimura, "Modified quadratic discriminant function and the application to chinese characters", IEEE PAMI, vol.9, no.1, pp.149–153, 1987.