Offline Cursive Character Recognition: A state-of-the-art comparison

John THORNTON, Jolon FAICHNEY, Michael BLUMENSTEIN, Vu NGUYEN and Trevor HINE
Institute for Integrated and Intelligent Systems,
Nathan Campus, Griffith University,
Brisbane, Queensland, 4111, AUSTRALIA
{j.thornton, j.faichney, m.blumenstein, v.nguyen, t.hine}@griffith.edu.au

Abstract. Recent research has demonstrated the superiority of SVM-based approaches for offline cursive character recognition. In particular, Camastra's 2007 study showed SVM to be better than alternative LVQ and MLP approaches on the large C-Cube data set. Subsequent work has applied hierarchical vector quantization (HVQ) with temporal pooling to the same data set, improving on LVQ and MLP but still not reaching SVM recognition rates.
In the current paper, we revisit Camastra's SVM study in order to explore the effects of using an alternative modified direction feature (MDF) vector representation, and to compare the performance of a RBF-based approach against both SVM and HVQ. Our results show that SVMs still have the better performance, but that much depends on the feature sets employed. Surprisingly, the use of more sophisticated MDF feature vectors produced the poorest results on this data set despite their success on signature verification problems.

1. Introduction

Recent work on offline cursive character recognition has indicated that support vector machine (SVM)-based approaches have greater discriminatory power than other competing techniques. In particular, Camastra's study (2007) showed his SVM-based character recogniser could significantly outperform both a multi-layer perceptron (MLP) implementation and a learning vector quantization (LVQ) approach on the large C Cube cursive character data set. Subsequent work using C Cube introduced a new hierarchical vector quantization (HVQ) algorithm that incorporates temporal pooling (Thornton et al., 2008). This approach outperformed both the MLP and LVQ results reported by Camastra but was unable to reach the recognition rates of the SVM-based recogniser. In parallel work, Liu (2008) evaluated a partial discriminative training approach on C Cube, improving on Camastra's results by using an SVM-based approach in combination with his own 200 element feature vectors.

One of the previously emphasised weaknesses of SVMs is the need for extensive parameter tuning in order to achieve optimal performance. In Camastra's study (2007), a time-consuming binary classification was performed where parameters were tuned for each letter class in the problem set. This overhead was used as an argument in favour of using HVQ, as HVQ can achieve robust, high quality recognition rates using default settings and without the need for feature extraction. However, in Liu's recent study (2008) it was mentioned that SVMs can be easily tuned using the variance of the sample vectors as a guide, although the exact method was not clarified. Liu's study also showed that significantly better recognition rates can be achieved by selecting a more extensive set of features, and both Liu and Camastra showed that the way characters are classed in both the training and testing phases has a major impact on performance.

In the current study, we aim to address several questions raised by this previous work. Firstly, as the selection of features is known to be an important determinant on performance, we decided to evaluate another well-known feature extraction method, the modified direction feature (MDF) technique of Blumenstein et al. (2007), on the C Cube data set. Secondly, we decided to investigate Liu's use of sample vector variance as a means of tuning SVMs, to see if we could reproduce Camastra's results without extensive experimentation. Thirdly, we revisited Thornton et al.'s HVQ study (2008), where relatively high recognition rates were achieved simply by presenting the HVQ with scaled bitmap images. This time we included contextual baseline information (already included in the C Cube data set) in the scaled bitmaps to see if this could improve the HVQ recognition rates. Finally, by a process of reverse engineering the C Cube dataset, we discovered that the feature vectors released on the cursive character challenge website (http://ccc.idiap.ch) did not match the corresponding character bitmaps, firstly in that the splits between training and test sets were different, and secondly in that 465 vectors could not be traced back to matching bitmap characters. We therefore decided to repeat a number of previous experiments to test whether the use different data sets has any significant effect on recognition rates.

The overall findings of the study are firstly that the MDF extraction technique is not competitive with either Camastra's 34 element feature vectors, or with Liu's 200 element vectors, and that MDF does perform better when teamed with SVMs rather that with neural networks. Secondly, we found that sample variance can provide reliable guidance for the selection of SVM parameter values. Thirdly, we found that augmenting the bitmaps presented to HVQ with baseline information produced a slight deterioration in performance, rather than the improvement we had expected. Finally, we found there was a difference in performance between the vectors available on the cursive character challenge website and the true vectors derived from the C Cube bitmaps.

In the remainder of the paper we first provide brief descriptions of the MDF and HQV techniques used in the experimental evaluation. We then discuss the C Cube dataset and the anomalies we found between the vector and bitmap data published on the cursive character challenge website. Following this, we describe the methods used to tune the SVMs in the experimental study and finally we discuss our experimental results.

2. Modified Direction Feature

The modified direction feature extraction technique combines the use of direction features (DFs) (Blumenstein et al., 2003) and transition features (TFs) (Gader et al., 1997) to produce recognition rates that are generally better than either DFs or TFs used individually. MDF extraction proceeds as follows: after initial preprocessing that leaves only the boundary of a character, direction features are used to encode the direction of each line as follows: 2 for vertical, 3 for right diagonal, 4 for horizontal and 5 for left diagonal (see Figure 1). Using this information direction transitions (DT) equal to the corresponding direction feature divided by 10 are extracted for each row (left to right and right to left) and each column (top to bottom and bottom to top). In addition, any contiguous set of equal value direction features is replaced by a single value. This is illustrated in Figure 1:

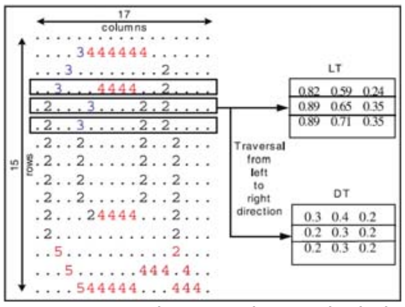

Figure 1. Example LT and DT calculations

Location transitions (LTs) are similarly calculated for each row and each column in both directions, with the relative start positions of each direction feature calculated as a proportion of the total width (in the case of a row) or height (in the case of a column). Given the initial set of LT and DT values corresponding to the actual number of rows and columns in the original character bitmap, the data is then normalised and locally averaged to fit into a space of 5 rows and 5 columns producing a final vector of 120 features (Blumenstein et al., 2007).

Experimental validation of the MDF approach (Blumenstein et al., 2007) using neural network techniques on the well-known CEDAR cursive character data set (Hull, 1994) confirmed that MDF performs better than either of the original DF and TF techniques on which it is based. In related work (Nguyen et al., 2007), MDF was applied to offline signature verification, where comparisons between neural network and SVM-based MDF implementations showed SVM to have the advantage. This study also indicated that an extended MDF approach (with added features relevant for signature verification) compares favourably to several of the best known signature verification techniques. However, more wide ranging comparisons between MDF and other state-of-the-art feature encoding techniques have yet to be made. In addition, apart from the signature verification paper of Nguyen et al., the published MDF results have been based on neural network implementations, so the question of the performance of MDF in conjunction with SVMs for general character recognition remains open.

3. Hierarchical Vector Quantization with Temporal Pooling

The HVQ with temporal pooling algorithm is a partial implementation of the work of George and Hawkins (2005) on hierarchical temporal memory (HTM). This biologically-inspired model places emphasis on the temporal aspect of pattern recognition, and consequently parses all images as ‘movies.’ The hierarchy itself is a full 4 level tree of degree 4 that processes a 32 × 32 pixel input character image. During training, each node receives input from the layer below, with leaf nodes receiving a 4 × 4 raw pixel image that is moved one pixel at a time across the node's receptive field, in a process known as sweeping. As the sweep progresses, we count how frequently one pattern follows another. This information is then used to create temporal groups that collect together patterns that have most frequently succeeded another during training. The same process of *temporal pooling* is repeated at each level up to the root node, where images are classified according to their character values. During recognition, an image is again swept across the leaf node sensors, as each non-root node estimates the membership probability of its input for each of its temporal groups. This information is propagated up to the root, which then outputs the most probable character classification (Thornton et al., 2008).

Thornton et al.’s study (2008) showed HVQ to be surprisingly robust to different forms of input (both skeletonisation and directional filters were applied) and to the setting of the HVQ parameters. In the various configurations evaluated, HVQ consistently achieved recognition rates of around 85% on the C Cube dataset (in comparison with the 89% for SVM reported in Camastra's study (2007)). However, as discussed below, there are issues relating to differences in the C Cube data set that put these earlier results in question. Additionally, the HVQ implementation did not take into account the contextual baseline information available in C Cube, which may have improved its performance. Consequently, a re-examination of the HVQ approach appears justified.

4. The C Cube Dataset

The C Cube dataset is available for download on the cursive character challenge (CCC) website (http://ccc.idiap.ch). It contains 38160 training characters and 19133 test characters taken from the CEDAR database and from the United States Postal Service database. On the website there are 4 files directly available: the training and test set character files (training.chr and testing.chr) and the training and test vector files (training.vec and test.vec that contain Camastra's 34 element feature vectors). There are two additional files in the CCC data directory (http://www.idiap.ch/~vincia/ccc): testOrdered.vec and trainingOrdered.vec. Upon analysing these files we discovered that the vectors in test.vec and training.vec do not exactly correspond to the characters in test.chr and training.chr. Firstly, while there is a match in the number of characters in each class of the .chr files with the corresponding vectors in the .vec files, the actual characters are distributed differently. For example, many of the "a" test vectors actually encode "a" characters that appear in the training.chr file. Secondly, we found 465 vectors that could not be matched to any bitmap characters. In contrast, the testOrdered.vec and trainingOrdered.vec files do perfectly match the corresponding .chr files. This means that in our previous study (Thornton et al., 2008), we were comparing HVQ on a different split of the C Cube data than that on which Camastra's SVM approach was evaluated – because we were testing HVQ directly on the character data. We therefore decided to repeat the earlier study, but this time reconstructing the character split corresponding to the test.vec and training.vec files. For the 465 vectors that could not be matched we selected the best value characters whose 34 element vectors had the minimal Euclidean distance from the unmatched vectors. We term this data set as *Split A*, and the data set corresponding to the original .chr files as *Split B*.

5. SVM Tuning

One of the problems with using SVMs is the large space of possible parameter settings and the sensitivity of SVMs to these settings. For our experimental study, we used the SVM radial basis function (RBF) kernel as it has produced the best performance both in Camastra's (2007) and Liu's (2008) studies. In practice, the RBF kernel requires the tuning of a single γ parameter (the second C parameter being fixed at 10). In Camastra's study, a separate value of γ was obtained for each class, via a procedure of binary classification. This proved to be a very time-consuming process. In contrast, Liu mentions in several studies that γ can be set according to the variance of the training set. For instance, in (Liu et al., 2003) σ^2 was set to 0.3*var* where "*var* is the estimated variance of the training set" and γ was set to $1/2\sigma^2$. Interestingly, this value of γ was then used in all SVMs rather than tuning separately for each binary classification.

In the current study we decided to follow up on Liu's work and find the best single γ value for each classifier using the training set variance as a guide. After some preliminary experimentation, we estimated σ^2 directly using the *within-class variance*, defined as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{N_c}\sum_{j=1}^{c_i}\sum_{k=1}^{N_f}(x_{ijk} - \bar{x}_{ik})^2}{(\sum_{i=1}^{N_c} c_i) - 1}$$

where N_f is the number of features per vector, N_c is the number of classes, c_i is the number of vectors in class i, and $\bar{x}_{ik}$ is the mean value of feature k within class i. We then estimated the variance-based gamma as $\gamma = 1/2\sigma^2$.

6. Results, Discussion and Conclusions

The experimental results in Table 1 uncover a number of interesting points. Firstly, there is a consistent difference between recognition rates on Split A and Split B of the C Cube data set – with all techniques performing better on Split A by a margin of 2-3%. These results clarify and confirm Thornton et al.'s (2008) finding that Camastra's 34D-SVM is better than HVQ on the C Cube dataset, when compared on *different* C Cube splits. Now, using equivalent splits, 34D-SVM still does better than HVQ, but by a smaller amount (the margin is now approximately 2% on equivalent splits rather than the previous 4% between different splits). This difference can be further reduced to 1% if we tune the temporal group size (see HVQ-16 in Table 1).

Secondly, we have come very close to Camastra's (2007) reported recognition rate for 34D-SVM of 89.61% on Split A, but this time using the training set variance to calculate the SVM γ parameter. This has avoided the lengthy manual tuning of each letter's classifier that was employed in the original study. The fact that such near-optimal results can be obtained using a single, statistically determinable γ setting also weakens the case for HVQ (which was fairly robust to different parameter settings), as an SVM is now equally easy to set up.

Nevertheless, HVQ still has the more general advantage that it can achieve high recognition rates without having to transform a bitmap image into a domain-specific feature set.

Thirdly, the results show that the MDF extraction technique is consistently worse than the other techniques, regardless of whether a neural network RBF or SVM approach is used. This fills an important gap in the literature, as MDF has not previously been directly compared with HVQ or with either of Liu or Camastra's feature sets. A comparison of MDF-RBF and 34D-RBF with the corresponding SVM implementations (MDF-SVM and 34D-SVM) also confirms that SVMs perform better than neural network RBFs on this data set. The relatively poorer performance of MDF indicates that Liu and Camastra have included more discriminative features in their vectors. In particular, MDF does not use the C Cube baseline information. This suggests that an augmented MDF may at least be able to reach the performance of the 34D vectors.

Fourthly, an evaluation of HVQ augmented with baseline information (drawn as dotted lines on the input images) caused recognition rates to decline by 1% to 83.68% on Split B. Consequently, we did not pursue this line of investigation further.

Finally, Table 1 reproduces Liu's 2008 result for his 200D feature set, using an SVM RBF kernel on the C Cube data set, and setting γ according to the sample variance.[1] This shows that Liu's feature set still has the better performance by a margin of nearly 4%.

In conclusion, this study has produced a fairly clear ranking of techniques on the C Cube dataset: first is Liu's 200D-SVM, second is Camastra's 34D-SVM, third is Thornton et al.'s HVQ and fourth is Blumenstein et al.'s MDF. The study has also shown that SVMs can be easily tuned on the basis of training sample variance, thereby eliminating a major obstacle to their practical deployment. In future work, it would be worth exploring whether the robustness of HVQ can be reproduced in other domains, and to experiment with combining features from Liu and Camastra's vectors with MDF and then to test these combinations on a broader range of cursive character databases.

Table 1. Overall recognition rates. All results report training on 52 letter classes and classifying on 26 letter classes. HVQ-x = Hierarchical Vector Quantization with maximum temporal group size = x. RBF = Matlab Radial Basis Function neural network with 5120 centres. SVM = SVM*light* software using the RBF kernel (http://svmlight.joachims.org/). 34D = Camastra's (2007) 34 element feature vectors. 200D = Liu's 200 element feature vectors (the 200D-SVM results are those reported in (Liu, 2008)).

	Split A		Split B	
Method	γ	Recognition Rate	γ	Recognition Rate
HVQ-32	n/a	87.16%	n/a	84.72%
HVQ-16	n/a	88.27%	n/a	85.58%
MDF-RBF	n/a	83.23%	n/a	80.92%
34D-RBF	n/a	87.15%	n/a	84.27%
MDF-SVM	0.28	85.92%	0.28	83.60%
34D-SVM	6.28	89.24%	6.36	86.20%
200D-SVM	5.31	93.11%	n/a	n/a

References

Blumenstein, M., Liu, X. Y. & Verma, B. (2007). An investigation of the modified direction feature vector for cursive character recognition. *Pattern Recognition 40*(2), 376-388.

Blumenstein, M., Verma, B. & Basli, H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters. Proc. Int. Conf. On Document Analysis and Recognition – ICDAR-03, pp. 137-141.

Camastra, F. (2007). A SVM-based cursive character recognizer. *Pattern Recognition 40*(12), 3721-3727.

Gader, P. D., Mohamed, M. & Chiang, J. H. (1997). Handwritten word recognition with character and inter-character neural networks, *IEEE Trans. Syst. Man Cybernet.—part B: Cybernetics 27*, 158–164.

George, D., Hawkins, J. (2005). A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. Proc. Int. Joint Conf. on Neural Networks - IJCNN-05, Montreal, Canada, pp. 1812-1818.

Hull, J. J. (1994). A database for handwritten text recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 550-554.

Liu, C.-L. (2008). Partial discriminative training for classification of overlapping classes in document analysis. *IJDAR 11*, 53-65.

Liu, C.-L., Nakashima, K. Sako, H. & Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition 36*, 2271-2285.

Nguyen, V., Blumenstein, M., Muthukkumarasamy, V, & Leedham, G. (2007). Off-line signature verification using enhanced modified direction features in conjunction with neural classifiers and support vector machines. Proc. Int. Conf. on Document Analysis and Recognition – ICDAR-07, Brazil, pp. 734-738.

Thornton, J., Faichney, J., Blumenstein, M. & Hine, T. (2008). Character recognition using hierarchical vector quantization and temporal pooling. Proc. Australasian Joint Conf. On Artificial Intelligence, New Zealand, pp. 562-572.

[1] Liu rescaled his vector data so his γ value is not directly comparable to the other SVM γ settings.