Allograph Based Writer Adaptation for Handwritten Character Recognition

Kumar Chellapilla, Patrice Simard, Ahmad Abdulkader

Allograph Based Writer Adaptation for Handwritten Character Recognition

Kumar Chellapilla
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
kumarc@microsoft.com

Patrice Simard
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
patrice@microsoft.com

Ahmad Abdulkader
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
ahmadab@microsoft.com

Abstract

Writer adaptation is the process of converting a generic (writer-independent) handwriting recognizer into a personalized (writer-dependent) recognizer with improved accuracy for a particular user. While training the generic recognizer uses large amounts of data from several writers, the adaptation process uses only a few samples from a single user.

In this paper we present a) an automatic approach for identifying allographs (character shapes/styles) from handwritten characters through clustering, b) a novel architecture for a personalizable recognizer that utilizes allograph information, and c) a kernel based approach for personalizing the recognizer. Using the new approach, personalization results with twenty one users indicate that handwritten single character recognition errors can be reduced by over 24% (or 41%) using as few as 5 (or 15) samples.

Keywords: Writer adaptation, personalization, allographs, handwriting, character recognition.

1. Introduction

It is well known that a user's handwriting is unique and can be used for identification [1]. Commercial handwriting recognition systems attempt to reduce the impact of writer variation through the use of large training datasets comprising data from several different users. However, as shown in [1], even when handwriting samples from as many as 1500 users are available, there is sufficient variation in the handwriting to uniquely identify each of the users.

From a machine learning perspective, such variation makes handwriting recognition difficult for computers. While intra-user characters (from the same user) have small variations, inter-user characters (from different users) have large variations and contribute to recognition errors. As a result, learning from training data obtained from one set of users (even hundreds of users) does not necessarily produce models that generalize well to unseen handwriting styles. The computer recognition experience using a generic recognizer can be especially poor for users with rare writing styles. One explanation for the poor performance is that the trained generic recognizer is incomplete as it has not learned to recognize unseen user's writing style(s).

A pragmatic approach to improving recognizer performance on unseen writing styles is writer adaptation (or personalization). Personalization enables the recognizer to adapt to a particular user's handwriting by collecting and learning from additional data samples from the user. Clearly, there is a trade off between the number of training samples needed from the user, the achieved reduction in error rate, and the perceived inconvenience to the user. The larger the amount of training data, the better the personalized recognizer, but the more inconvenienced the user.

The challenge lies in finding a sweet-spot where we get the most improvement in accuracy with the least number of user samples. Such a sweet spot is likely to exist, as the first few samples from the user produce the most dramatic improvement in recognition accuracy with the user being least inconvenienced.

In this paper, we present a novel allograph (character shape/style) based recognizer that is well suited for personalization. Previous approaches are reviewed in Section 2, and a new architecture for an allograph based personalizable recognizer is presented in Section 3. Sections 4 and 5 present experiments and results, respectively. Conclusions are offered in Section 6.

2. Background

Personalization of recognizers to improve error rates is a well studied problem. Several approaches have been proposed with varying degrees of success. Matic et al. [2] addressed the problem of writer-dependent recognition of digits and upper case characters through the use of a time delay neural network as a preprocessor and an optimal hyperplane classifier for personalization. Platt [3] exploited the fact that the output of a neural network is characteristic of the input, even when the output is incorrect, and constructed an output adaptation module that adapts the recognizer. Connell and Jain [4] first identify writer-independent writing styles (called lexemes), and then adapt these lexemes to a particular user's handwriting.

3. Method

We propose a personalizable recognizer architecture that is completely based on machine learning. Character writing styles (allographs) are identified using an hierarchical agglomerative clustering approach using dynamic time warping (DTW) as a distance measure. The generic recognizer comprises two neural network classifiers whose outputs are combined. The first neural network (allograph-NN) uses allograph information, while the second (base-NN) does not. A linear combiner is used to produce the generic (un-personalized) recognizer. The personalizer is a support vector machine that learns to optimally combine the two neural networks based on new user samples.

3.1. Finding Writing Styles using Clustering

Hierarchical clustering techniques can be used to learn letter handwriting styles from data. Two main approaches exist: (a) a top down approach of detecting sub-styles [5-6], and (b) a bottom-up clustering approach [7-8]. In this paper, the bottom-up approach is adopted because the obtained style knowledge can be directly used in the recognizer.

A clustering $\boldsymbol{C}$ of handwritten letters $X = \{x^1, x^2, \ldots, x^M\}$ defines a partitioning of the data into a set $\{c^1, c^2, \ldots, c^K\}$ of K disjoint sets, such that $\bigcup_{k=1}^{K} c^k = X$. The clustering $\boldsymbol{C}$ is computed independently for every letter. An *hierarchical clustering* algorithm produces an hierarchy of *nested* clusters $[\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_M]$ such that $\boldsymbol{C}_{m-1}$ is a subset of $\boldsymbol{C}_m$. This hierarchy is built in M steps, where a clustering at step m is produced from the clustering produced at step $m-1$. At step 1, every member in the sample set X represents a cluster of its own. Using a dissimilarity function $D(c^k,c^{k'})$ of two clusters, the following algorithm is applied:

a) Initialize $\boldsymbol{C}_1 = \{\{x^1\},\{x^2\},\ldots,\{x^M\}\}$, where each sample is a cluster by itself.
b) For $m = 2, \ldots, M$: obtain the new clustering $\boldsymbol{C}_m$ by merging the two most similar clusters c^{kmin} and $c^{k'min}$ of $\boldsymbol{C}_{m-1}$. The closest clusters are defined by
$(kmin, k'min) = \arg\min_{(k,k'),k\neq k'} D(c^k,c^{k'})$

3.1.1. Cluster and Sample Dissimilarity Function

The cluster dissimilarity function $D(c^k,c^{k'})$ is defined in terms of the ink sample dissimilarity function $D(x^k,x^{k'})$. Each ink sample is first isotropically normalized and centered within a fixed size rectangle. For ink samples k (comprising, say, S strokes), and k' (comprising, say, S' strokes),

$$D\left(x^k, x^{k'}\right) = \begin{cases} \infty & , \text{if } S \neq S' \\ \dfrac{\sum_{n=1}^{N} \left|P_n, P'_n\right|}{S} & , \text{if } S = S' \end{cases} \qquad (1)$$

where P and P' are the corresponding re-sampled coordinate vectors of samples k and k', respectively, and N is the number of sampling points. An element p in the vector P has 3 co-ordinates (x, y, θ) where x, y are the Cartesian coordinates of the point p and θ is the estimate of the slope at the same point.

$$D\left(c^k, c^{k'}\right) = \max_{\forall x^k \in c^k, \forall x^{k'} \in c^{k'}} D\left(x^k, x^{k'}\right) \qquad (2)$$

The decision to use the maximum rather than average or the minimum to define the distance between ink samples favors compact clusters. Since distance between clusters with different stroke lengths is ∞, they will not be merged until the very end. At that point the merging would have actually stopped.

3.1.2. Number of Clusters

A key problem in clustering algorithms in general is the determination of the number of clusters. In our experiments we determined the number of clusters for every letter by defining a threshold D_{max} above which no further merging of clusters occurs. This means that the active clusters at the time that merging stops represent the styles of the corresponding letter. Accordingly, the number of resulting styles is different from one letter to the other, depending on how diverse the letter shapes are.

3.1.3. Clustering Results

The hierarchical clustering algorithm described previously was applied to a large set of ink samples (see Section 4.1). Figure 1 shows example styles for the letters q, t, and X and their relative frequencies among writers.

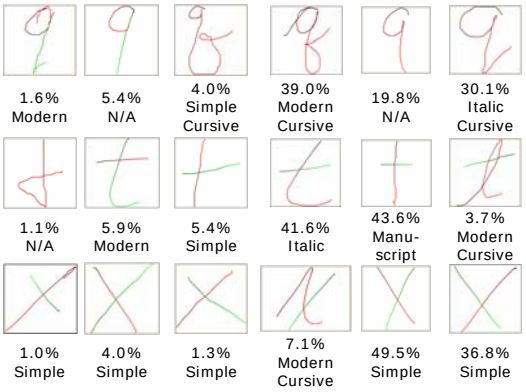

Figure 1. Styles and distributions for q, t and X.

3.1.4. Selecting Allographs

Each choice of a DTW distance threshold when applied to the hierarchical cluster produces a set of disjoint clusters. The larger the distance threshold, the fewer the number of clusters obtained. For the experiments reported in this paper, a threshold was chosen to obtain 2002 unique clusters for the 100 characters (printable ASCII characters including the euro and pound signs). With 2002 clusters and 100 characters, there are approximately 20 allographs per character representing various written forms of the character.

3.2. Feature Vectors

Each handwritten character may be viewed as a sequence of (*x*,*y*,*t*) segments representing continuous strokes. One or more strokes written in succession make up a character. Each handwritten character was processed to obtain sixty five polynomial features using the approach in Rowley et al [10].

3.3. Training Base and Allograph Classifiers

Two recognizers are trained using these feature vectors. The first recognizer (allograph-NN) is shown in Figure 2 and comprises a neural network and a linear classifier in a cascade.

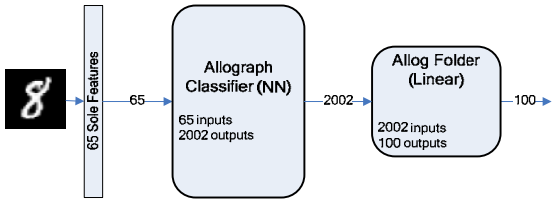

Figure 2. The allograph classifier.

The neural network has 2002 outputs and is trained to map the character feature vector to character allographs. A linear combiner (allograph-folder) is trained using gradient descent to fold the 2002 allographs back into the 100 character classes. The linear folder is considered to be a part of the allograph-NN. The second recognizer (shown in Figure 3) is a neural network (base-NN) that does not use allograph information and is trained to directly map the feature vectors to the output classes. Both neural networks are multi-layer-perceptrons (MLP) with two layers each. While the allograph-NN has 1024 hidden nodes, the base-NN has 600 hidden nodes. Backpropagation was used to train the neural networks with cross-entropy as the error function [9].

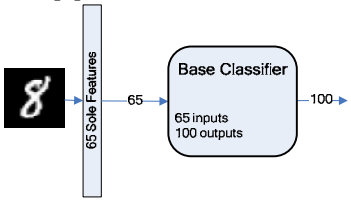

Figure 3. The base classifier.

3.4. Combiner for improved accuracy

The two neural networks (allograph-NN and base-NN) have different architectures. Further, the former is trained using allograph information, while the latter is not. Due to these differences, the errors made by these two classifiers can be expected to be significantly different. Any combiner built using these two classifiers will likely have a lower error rate than either of them. A simple linear classifier that combines the outputs of the allograph-NN and the base-NN comprises the writer-independent (unpersonalized) recognizer.

3.5. Personalizer

The personalizer's role is to adapt the writer-independent recognizer to the current user providing new training samples. It is designed to replace the linear combiner in the unpersonalized recognizer with one that is optimal for the new user. The basic personalizer architecture is shown in Figure 4.

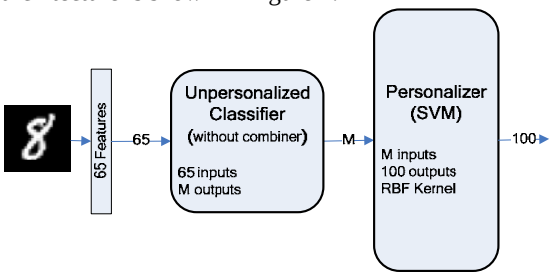

Figure 4. The unpersonalized recognizer and Personalizer are arranged in a cascade.

Any suitable combiner classifier that can learn from data can be used to replace the linear combiner. In this work, support vector machines (SVMs) were chosen for the following reasons:

1. **Generalization**: SVMs are well known for their generalization properties. Since, the number of samples collected (per class) from the user will be very small (typically less than 10 or 15), it is very important that generalization is achieved with such few samples. In contrast to SVMs, training neural networks to generalize well with very limited training data is a challenging problem.
2. **Regularization**: The most common approach to achieving good generalization with small data sets is regularization. SVMs provide a natural way of regularization. The model selection process can be used to effectively control capacity and limit over-fitting.
3. **Multi-class**: Multi-class SVMs can built using several two-class SVMs. This allows for finer control on accuracy on a per class basis. Since only the linear combiner is being personalized, not all two-class SVMs are necessary. One can simply focus on including only those pairs of classes that have the highest confusion.
4. **Complexity**: When the one-vs-one approach is used the number of two class classifiers grows proportional to $C(n,2)$ i.e., $O(n^2)$. This can be a problem with a large number of classes. However, the support vectors in an SVM are a subset of user provided samples. Thus, even though the number of possible classes and classifiers grows quickly, the total number of support vectors is bounded by the number of user samples, which is small. Further, since only the combiner is being personalized, only a small subset of the C(*n*,2) classifiers need to be built. Each of the dropped classifiers can be represented by

a single bit indicating that the unpersonalized recognizer's output is to be used instead[1].

4. Experiments

4.1. Datasets

Two data sets were used in the personalization experiments:

1. The first set (non-personalization set) consisted of 200,000 handwritten characters from 215 users. This data is used for the generic recognizer.
2. The second set (personalization set) consisted of 84,000 samples from 21 new users and is designed for evaluating the personalization process.

Data in both sets was uniformly distributed over 100 possible western handwritten character classes given by:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
!"#$%&'()*+,-
./:;<=>?@[\]^_{|}~¢£¥§°±€
```

The 200,000 ink samples from the non-personalization set were hierarchically clustered to obtain 2002 allographs. These clusters were used to assign allograph labels for each of the 200,000 samples.

4.2. Generic Recognizer

The generic recognizer comprises two classifiers: a) the allograph-NN (which also contains the allograph-folder), and b) the base-NN. The non-personalization set was shuffled and split into 3 parts: 160,000 samples to be used for training, 20,000 samples to be used for validation (to determine when to stop training), and the remaining 20,000 samples to be used for testing. The reported accuracies for the generic recognizer on the non-personalization data set are the ones from the 20,000 test set. In each of the figures, the first percentage value indicated on a classifier is the error rate on this test set.

The allograph-NN is a two layered multi-layer perceptron (tanh nonlinearity) with 1024 nodes in the hidden layer and 2002 output nodes (one per allograph). The allograph-folder is a simple linear combiner that maps the allograph-NN outputs to the 100 output classes. The base-NN is also a two layered multi-layer perceptron (tanh nonlinearity) with 600 hidden nodes and 100 outputs (one per output class).

All classifiers (allograph-NN, allograph-folder, and base-NN) were independently trained on the non-personalization set using backpropagation and cross-entropy as the error measure. The generic combiner is a simple linear classifier with 2202 inputs and 100 outputs. Its inputs comprised all of the outputs of the allograph-NN, the allograph-folder and the base-NN.

[1] For dropped pairs, during the SVM voting step, the corresponding pair of unpersonalized combiner's outputs can be compared to obtain the vote.

4.3. Personalizer

The personalizer is a 100-class SVM using up to C(100,2) = 4950 2-class SVMs. A unique personalizer is trained for each of the 21 users.

The 84,000 samples in the personalization data set produce 40 samples per character for each of the 21 users. Up to 15 samples per character are used to train the personalizer. The remaining 25 samples per character are used purely for testing. We do not expect real users to provide more than 15 samples per character for personalization. However, having a large test set (25 or more samples per char) gives us a reliable way of evaluating the personalized recognizer.

Three different personalizers were built for each user, utilizing k = 5, 10, and 15 user samples (per class). These k-sample sets were incrementally selected, i.e., for example the k = 10 set was obtained by adding five new sample to the k = 5 set. The k samples were used to not only train the recognizer, but also regularize it. ceil(k/2) samples were used for training and floor(k/2) samples were used for model selection. The RBF kernel was used. SVM model selection was performed using a simple grid-search [11] with C in $\{2^{-5}, 2^{-4},\ldots, 2^{14}, 2^{15}\}$ and γ in $\{2^{-10}, 2^{-9},\ldots, 2^{3}, 2^{4}\}$. The (C,$\gamma$) parameters from the model that gave the best error rate on the floor(k/2) samples (not used for training the SVM) was chosen for the personalizer. This error rate is reported as the error rate of the personalized recognizer.

5. Results

5.1. Base Recognizer

The base-NN (shown in Figure 3) achieved a test error rate of 7.8%. When tested on data from the 21 users in the personalized dataset (not included in the 215 users), the error rate increased to 9.36%. This is a relative increase of 20% in the error rate. Such a large increase in the error rate clearly indicates that the inter-user variation is much smaller than the intra-user variation in handwriting styles.

5.2. Allograph Recognizer

The allograph classifier (Figure 2) attempts to predict not only the character label but also the writing style of the character. On the non-personalized dataset, the allograph classifier gets an error rate of 24.65%. The error rate is very large.

However, when we simply fold the 2002 character styles into their associated 100 character classes (simple folder), the error rate drops to 8.25%. For any given character, the simple folder returns the sum of the allograph outputs corresponding to that character.

A better folder can account for confusable allographs among different classes. When a simple linear folder (learned weighted sum over all 2002 outputs) is used the unpersonalized test error rate drops to 5.9%. However, the error rate on the personalization test set dramatically

increases to 11.40%. This increase in error rate (93%) is larger than that observed for the base recognizer (20%), indicating that the allograph distribution varies significantly between the 215 users in the non-personalization data set and the 21 users in the personalization data set. However, previous work [3] has shown that even though the allograph distribution varies, for any new user the probability distribution over the classifier outputs is similar over several samples. In other words, though the error rate increases, the new user errors are predictable. Thus, the personalizer can learn to reduce these errors.

5.3. Generic Recognizer

The unpersonalized combiner (shown in Figure 5) is a linear classifier that takes as input the 2002 outputs of the allograph classifier, the 100 outputs of the allograph folder, and the 100 outputs from the base classifier. It maps these inputs to the 100 output classes. The unpersonalized combiner achieves a test error rate of 5.8% on the non-personalized data set and a corresponding 9.51% test error rate on the personalized data set. The performance slightly improves.

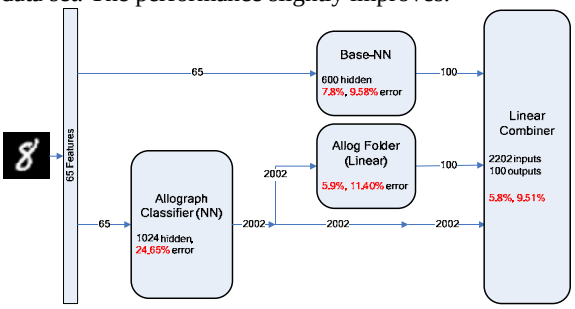

Figure 5. Unpersonalized Recognizer.

5.4. Personalized Recognizers

A unique personalized recognizer (Figure 6) was built for each of the 21 users in the personalized data sets. The personalizer reduces the mean error rate from 9.51% to 5.64%. This relative reduction in error rate of over 40.6% clear indicates that the personalizer is effective in tuning the recognizer to each of the individual users. Table 1 presents a summary of the performance before and after personalization. Figures 7 and 8 present the error rates for each of the users before and after personalization using 15 samples. The personalizer reduced the error rate for 20 of the 21 users. However, on one user (user 12 in Figure 7), the number of errors increased slightly by 3.7% (relative increase).

The training time for each personalizer was less than 300 seconds (5 minutes). Each pair-wise SVM classifier (taking 8 samples for the first class and 8 samples for the second class) takes about 0.27 milliseconds to train on a Pentium 3.0 GHz machine. Training all 4950 pair-wise classifiers take only 1.33 seconds. However, this has to be repeated for each of the 255 (C,γ) settings for model selection using grid search. Using more advanced model selection methods [12] can reduce this by one or two orders of magnitude. Further reduction in training times can be achieved by building only those pair-wise classifiers that correspond to the largest values in the confusion matrix. Class pairs that have no confusion can be dropped. With all unpersonalized error rates under 15%, for the 100 class problem used in this paper, this simple approach has the potential to produce speed improvements of over 6 times. Further, such an approach may be the only way to build the personalizer when the number of classes is very large. For example, East-Asian languages (Chinese, Japanese and Korean) typically have several thousand characters. User may be expected to provide a few samples only for the most misrecognized characters.

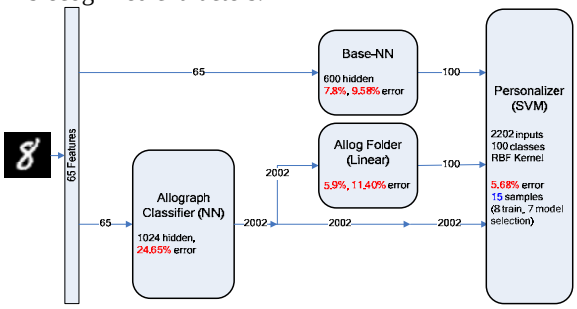

Figure 6. Personalizer architecture: The personalizer reduces the error rate from 9.51% to 5.64% using 15 samples (per character).

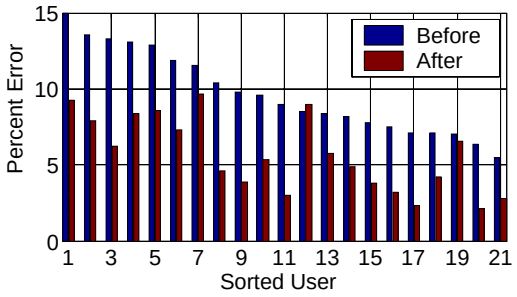

Figure 7. Percentage error before and after personalization for the 21 users.

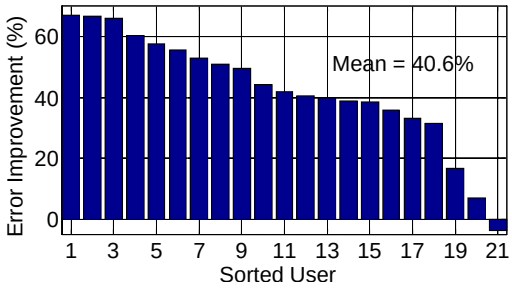

Figure 8. Percent improvement in error rate. The users were sorted based on their improvement. While 20 users benefited from personalization and one user's error rate slightly increased after personalization.

Table 1. Accuracy of different recognizers.

Classifier	Test Error	
	Generic data set	Personalization data set
Base-NN	7.80%	9.58%
Allograph-NN (Simple Folder)	8.25%	-
Allograph-NN (Linear Folder)	5.90%	11.40%
Generic (Fig. 5)	5.80%	9.51%
Personalized (Fig. 6)	-	5.68%

5.5. Effect of the number of user samples

During personalization, the greater the number of samples required from the user, the lower the personalized error rate, but greater the user discomfort. The rate of improvement is expected to diminish with increasing number of samples. Personalization experiments were repeated with 5, 10, and 15 samples (per character) from each user. Figure 9 shows the personalized error rate as a function of the number of user samples.

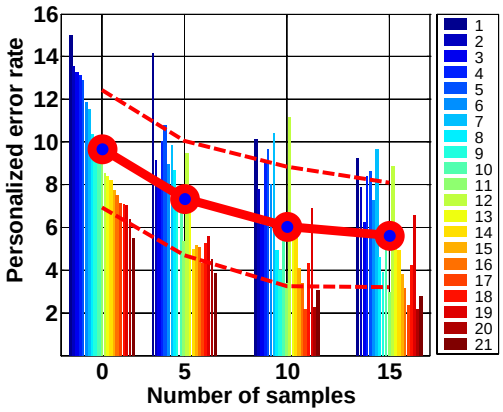

Figure 9. The personalized error percentage as a function of the number of samples provided by the user. Users were sorted based on their unpersonalized error rate (zero samples). The thick line shows the mean percent error after personalization and the dashed lines show one standard deviation bounds.

The personalized error rate was 7.37%, 6.06%, and 5.64%, with 5, 10, and 15 samples from the user. These values correspond to a relative reduction of 23%, 36%, and 41%, respectively. The drop in error rate is the highest in the first five samples. The error rate continues to decrease even after 15 samples. However, given the rate of improvement, it appears that collecting more than 10 or 15 samples from the user may not warrant the subsequent reduction in the error rate. One approach to expanding the number of training samples is through the judicious use of ink based distortions. We plan to explore the use of such distortions in future work.

6. Conclusion

We presented a clustering approach for identifying allographs and a novel personalizable recognizer that utilizes allograph information. The personalizer is a support vector machine that acts as a combiner and tunes itself to reduce recognizer errors for a particular user. Experimental results on 21 users clearly indicate that the new approach reduces handwritten single character recognition errors by over 24% (or 41%) using as few as 5 (or 15) samples. The personalizer takes less than a few minutes to train and the pair-wise SVM model allows the personalizer to scale to a large number of classes. Future work involves the use of sparse multi-class SVMs for scaling to large classes and ink distortion for reducing the number of user samples needed for personalization.

7. References

[1] SN Srihari, SH Cha, H Arora, and S Lee, "Individuality of handwriting," *Journal of Forensic Sciences*, 47(4), pp. 856 - 872 July, 2002.

[2] N Matic, I Guyon, J Denker, and V Vapnik (1993), "Writer-adaptation for on-line handwritten character recognition," *ICDAR'93*, pp. 187-191.

[3] JC Platt and Nada Matic (1996), "A constructive RBF network for writer adaptation," *NIPS'96*, pp.765-771.

[4] SD Connell and AK Jain (2002), "Writer adaptation for online handwriting recognition," *IEEE PAMI*, vol. 24, no. 3, pp. 329-346.

[5] L Vuurpijl, L Schomaker (1996), "Coarse writing-style clustering based on simple stroke-related features," in *Proc. of 5th IWFHR*, pp. 29-34.

[6] C Bahlmann, H Burkhardt (2003), "The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping". *IEEE PAMI*, v. 26, No. 3, pp. 299-310.

[7] J-P Crettez (1995), "A set of handwriting families: style recognition". ICDAR'95, pp. 489-494.

[8] L Vuurpijl, L Schomaker (1997), "Finding structure in diversity: a hierarchical clustering method for the categorization of allographs in handwriting," *ICDAR'97*, pp. 387-393.

[9] C. Bishop (1996), *Neural Networks for Pattern Recognition*, Oxford University Press.

[10] HA Rowley, M Goyal, J Bennett, "The effect of large training set sizes on online Japanese Kanji and English cursive recognizers," *IWFHR'02*, pp. 36-40.

[11] N Cristianini, J Shawe-Taylor (2000), *An Introduction to Support Vector Machines*, Cambridge Press.

[12] T Hastie, S Rosset, R Tibshirani, J Zhu (2004), "The Entire Regularization Path for the Support Vector Machine," *Journal of Machine Learning Research* 5, pp. 1391–1415.