Matching Handwriting with Its Author

Ziran Jiang, Aditya R. Mundada

***Abstract*—Handwriting matching is a useful feature to identify individuals and is used for many purposes including bank check authentication and forensic investigation. This paper implements, compares, and optimizes handwriting matching using two algorithms: Naïve Bayes and Support Vector Machine (SVM). Handwriting samples are collected using the INKredible app [1] to obtain realistic samples similar to handwritings on paper. 100 writing samples are collected from each of the three authors (authors A, B, and C), and each sample is scaled to four different image resolutions. A preprocessing algorithm is developed using MATLAB to convert the collected samples to black and white and normalize the size of the handwriting. Naïve Bayes and SVM algorithms are implemented using MATLAB to distinguish between samples from authors A and B. SVM is also expanded to distinguish samples between all three authors. Performance of Naïve Bayes and SVM is compared, and the effect of image resolution and preprocessing is also analyzed.**

I. Introduction

Each person has a unique handwriting, and this makes handwriting a useful feature to identify individuals [10]. Handwriting matching is used by banks for check-writing and signature authentication. In forensic science, handwriting matching algorithm can aid handwriting analysis experts predict the author with more accuracy. The goal of this project is to manually implement and optimize handwriting matching, including: 1) Data Acquisition, 2) Image Preprocessing, 3) Algorithm Implementation (Naïve Bayes, SVM, and SVM for three authors), and 4) Algorithm Comparison and Optimization. The input to the Naive Bayes and SVM algorithms are all the pixel values ('1' for white, and '0' for black) from the handwriting sample image. The output of the algorithms is the prediction of the corresponding author.

II. Related Work

Although intuitively we know that handwriting is different for every individual, the uniqueness of each person's handwriting was studied and objectively validated by [10, 15] through a machine learning approach. Handwriting features can be divided into two categories: document examiner features, and computational features [15]. Document examiner features, such as handwriting embellishments, are often used by forensic handwriting examiners and are difficult to model using computers. [15] used these document examiner features and obtained promising results. Computational features can be easily modeled by machine learning algorithms, and are used in [16, 17].

Another challenge of handwriting matching is generating a large number of training samples for machine learning algorithms. Unlike typing in computer, handwritings requires manual work, and generating a large number of handwriting samples is very time consuming. A number of studies have been done to create algorithms that will automatically generate many artificial handwriting samples based on some original authentic samples written by human [11, 12, 13, 14]. [12] and [14] only require one original sample, and use a deformation model to deform the original sample to generate many new samples. This method however does not always generate natural-looking handwritings. The study done by [11] tries to learn the natural variation from multiple authentic samples, and create a distribution that describes the variation. It then synthesizes new samples from this distribution. This method requirs many original handwritings to have an accurate distribution.

III. Data Acquisition

We considered using the MNIST Database [2] for handwriting samples, however the MNIST Database samples are not associated with the corresponding authors. For handwriting matching, the algorithm needs to know the author for each training samples, so the MNIST Database could not be used as training or test samples. Instead, as a temporary measure at the initial stage of algorithm implementation, MS Paint was used to generate handwriting samples. Later we switched to using an app called INKredible [1], which allows directly writing on a tablet screen using stylus/finger. The INKredible app enables collecting realistic handwriting samples similar to the samples written on paper. Each handwriting sample was scaled to the following four resolutions using MATLAB: 64x64 pixels, 32x32 pixels, 16x16 pixels, and 8x8 pixels.

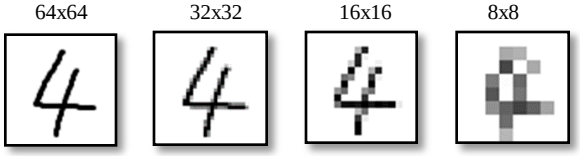

Fig. 1. Handwriting samples at different resolutions

100 handwriting samples were collected from each of the three authors (authors A, B, and C), and each sample was scaled to the four different resolutions mentioned above. As shown in Fig. 2, each author has different writing style, and the algorithms attempt to predict the author based on these differences. Fig. 2, only shows one sample from each author, but within the 100 samples from the same author, there is also certain variations from sample to sample. This sample-to-sample variation reflects the real world situation where a person writes slightly differently each time. The features for

the algorithms are all the pixel values (‘1’ for white, and ‘0’ for black) from the handwriting sample image.

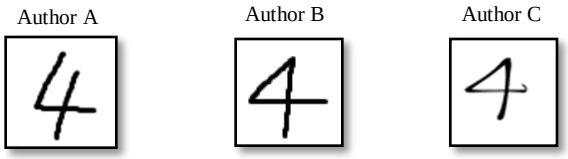

Fig. 2. Handwriting samples from authors A, B, and C

IV. IMAGE PREPROCESSING

While there are many image pre-processing techniques depending on the condition of scanned handwriting, we identified the following three preprocessing techniques as crucial to the functionality of handwriting matching algorithm:

1. Conversion of image to B/W – each pixel will either be ‘1’ (for white) or ‘0’ (for black).
2. Handwriting size normalization.
3. Background removal

We have implemented part 1 and part 2 of the preprocessing algorithm in MATLAB. Conversion to black and white is achieved using the in-built MATLAB function called ‘rgb2gray’. This function uses the luminance equation to convert RGB pixels to grayscale numbers. The equation used is:

$$0.2989R + 0.5870G + 0.1140B \quad \dots \quad (1)$$

The R, G and B in the above equation are respective color channel values for any given pixel. Once the image is converted to grayscale, we use the bounding box method to normalize our image. In this, the algorithm first finds a lower and upper, row and column bounds to fit the image in the smallest possible rectangle. The rectangle bounding box is converted to a square by expanding the smaller side to make it equal to the larger side. The new pixels that get added as a result of this operation are initialized to ‘white’ color. Note that the smaller side is expanded on either side to automatically center the image. This bounded box image can now be scaled to any pixel resolution using the MATLAB function ‘imresize’. The default algorithm used by ‘imresize’ to scale the image is bi-cubic interpolation. There are other options available as well, such as, nearest neighbor and bi-linear interpolation. Bi-cubic interpolation performs a weighted average computation in a 4x4 neighborhood of the pixel thus resulting in a more smoothened edge outputs for higher scaling factors as compared to bilinear interpolation which works in a 2x2 neighborhood. Thus, it was our choice of algorithm.

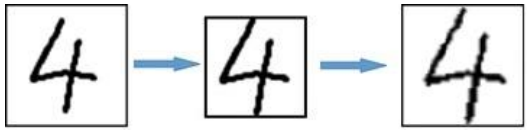

Fig.3. Bounding box based size normalization

V. NAÏVE BAYES ALGORITHM

Naïve Bayes algorithm was implemented to distinguish the handwriting samples from author A and author B. We varied the number of training samples (half and half from authors A and B) and used 100 test samples (50 from author A and 50 from author B). At each number of training examples, the average generalization error was obtained by averaging the generalization error collected over 100 runs, where each run used different randomly picked training samples.

To make a prediction on a new test sample, we compare the following equations (2) and (3):

$$\mathrm{p(y = 1|x)} = \frac{p(x|y=1)p(y=1)}{p(x)} \quad \dots \quad (2)$$

$$\mathrm{p(y = 0|x)} = \frac{p(x|y=0)p(y=0)}{p(x)} \quad \dots \quad (3)$$

In the equations above, y = 1 means the author is A, and y = 0 means the author is B. Since we always used half from author A and half from author B for the training and test samples, p(y=0) = p(y=1) = 0.5. Also,

$$\mathrm{p(x|y = 1)} = \prod_{i=1}^{n} p(x_i|y = 1) \quad \dots \quad (4)$$

$$\mathrm{p(x|y = 0)} = \prod_{i=1}^{n} p(x_i|y = 0) \quad \dots \quad (5)$$

Here, n is the total number of pixels in an image sample. Therefore, in a sample of 64x64 pixels image, n = 64×64 = 4096, which implies x_i is the value of i^{th} pixel. To calculate each $p(x_i|y = 1)$ and $p(x_i|y = 0)$, Laplace smoothing was used:

$$p(x_j|y = 1) = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1 \wedge y^{(i)}=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}+2} \quad \dots \quad (6)$$

$$p(x_j|y = 0) = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1 \wedge y^{(i)}=0\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}+2} \quad \dots \quad (7)$$

In equations (6) and (7), m denotes the total number of training samples, so $x_j^{(i)}$ means the value of j^{th} pixel in the i^{th} training sample.

To model p(x|y), the Naïve Bayes assumption assumes that the x_i’s are conditionally independent given y [7]. This means that for example we are assuming given the author is A (or B), knowing the value of pixel ‘i’ has no effect of our beliefs about the values of pixel ‘j’. This Naïve Bayes assumption does not entirely hold true in this case. Because we know that for any handwriting, especially for the high resolution images, if a pixel is black so that it’s part of the letter/number, then the adjacent pixels are also likely to be part of the letter/number. With this in mind, the Naïve Bayes algorithm was implemented and the generalization error results were obtained.

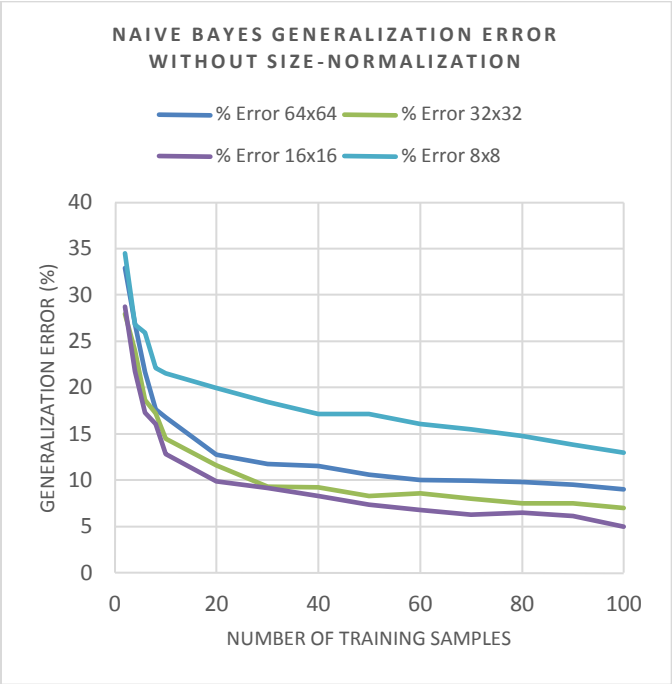

Fig. 4. Naïve Bayes generalization error without size-normalization and centering

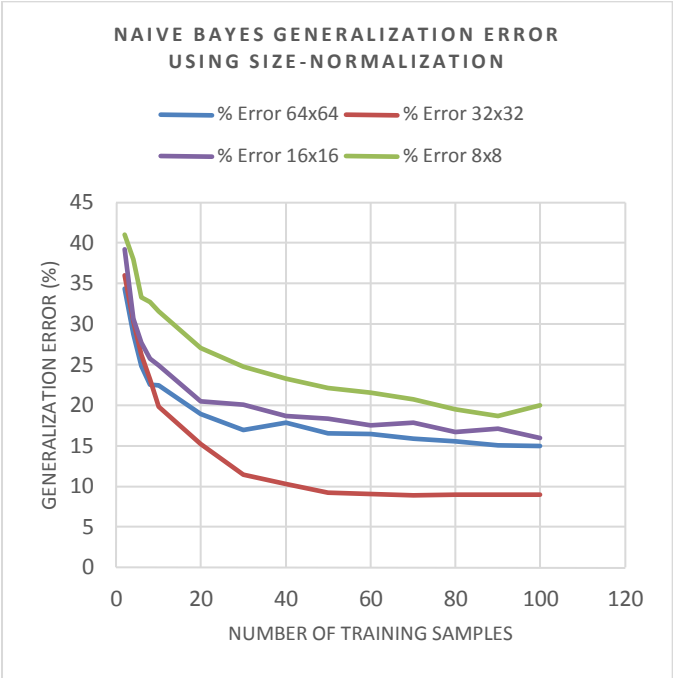

Fig. 5. Naïve Bayes generalization error using size-normalization and centering

Refer to Fig. 4 for the generalization error of Naïve Bayes without size-normalization and centering. The 16x16 images resulted in the lowest generalization error of 5%, followed by 32x32 (7%), 64x64 (9%), 8x8 (13%) images. The higher resolution samples (64x64 and 32x32) had higher generalization error compared to the 16x16 images because the number of training samples were not enough to generate an accurate result. For example a 64x64 image has 4096 features (pixels), and 100 training samples were not sufficient compared to the number of features in each image. For the 16x16 training images, the number of feature is 16×16 = 256, which is comparable to the 100 samples. As for the 8x8 samples, the performance was worse than 16x16 images because they lost too much of the original characteristics of the handwriting. However, note that even at the lowest resolution of 8x8 pixels, the algorithm still achieved 13% generalization error. This is impressive considering that an 8x8 resolution image has such a low resolution that it is hard to recognize even by human eyes, as shown in Fig. 1.

Refer to Fig. 5 for the generalization error of Naïve Bayes using size-normalization and centering. The results were worse than without using size-normalization and centering for all image resolutions. Size-normalization and centering did not improve the performance of Naïve Bayes. We think this is because size-normalization and centering actually reduced the difference of the writing samples from the two authors. For example, if one author generally writes big and the other author have smaller handwriting, this difference will be reflected in the raw image. Naïve Bayes algorithm then picks up this difference in the $p(x_i|y=1)$ and $p(x_i|y=0)$ calculation to make the prediction. However if the images are size-normalized, then the writing samples from the two authors will be scaled to similar sizes, and they will become more similar compared to the raw images. With more similarity, the prediction will become less accurate, and this is why the generalization of Naïve Bayes algorithm increased when using the size-normalization and centering.

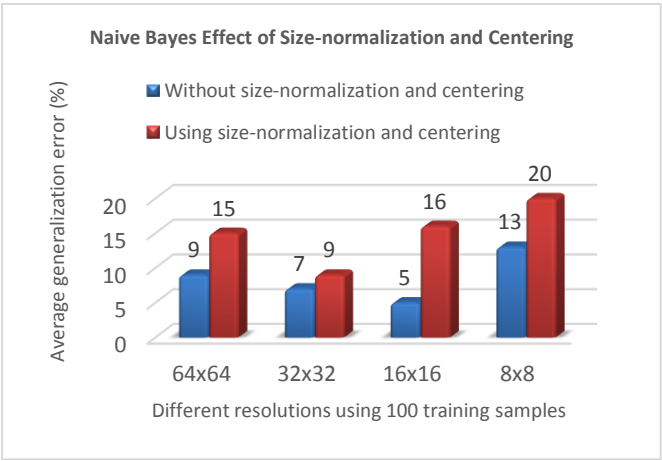

Fig. 6. Naïve Bayes effect of size-normalization and centering

VI. SUPPORT VECTOR MACHINE (SVM) ALGORITHM

Since we are evaluating supervised learning algorithms, a discussion without SVM is incomplete. We used an off-the shelf SVM training algorithm provided by MATLAB to understand if we can classify images to their respective writers. SVM is modelled using the primal optimization problem [7]:

$$\min_{\gamma,\omega,b} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m}\xi_i$$

$$s.t.\, y^{(i)}\left(\omega^T x^{(i)} + b\right) \geq 1 - \xi_i, i = 1, \dots, m$$

$$\xi_i \geq 0, i = 1, \dots, m$$

Here, ω denotes the weight matrix, $x^{(i)}$are our samples (images), $y^{(i)}$ is the class label, C is a parameter that does relative weighting of the twin goals of minimizing the functional margin and ensuring that all samples have functional margin of at least 1, and lastly, ξ_i is the quantity by which the functional margin for a sample may be less than 1 which results in an extra cost of $C\xi_i$. After writing the Lagrangian for the above problem and setting the partial derivatives of Lagrangian w.r.t ω and b to zero, we get our dual optimization problem [7]:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)}, x^{(j)}\rangle$$

$$s.t \begin{cases} 0 \leq \alpha_i \leq C, i = 1, \dots, m \\ \sum_{i=1}^{m}\alpha_i y^{(i)} = 0 \end{cases}$$

The 'svmtrain' function from MATLAB solves the above optimization problem and calculates the values of all the parameters. We used a linear kernel for our SVM classifier. Again, we had 100 samples from each author which were split into 50 training samples and 50 test samples. We have compared the performance of the algorithm over various image resolutions.

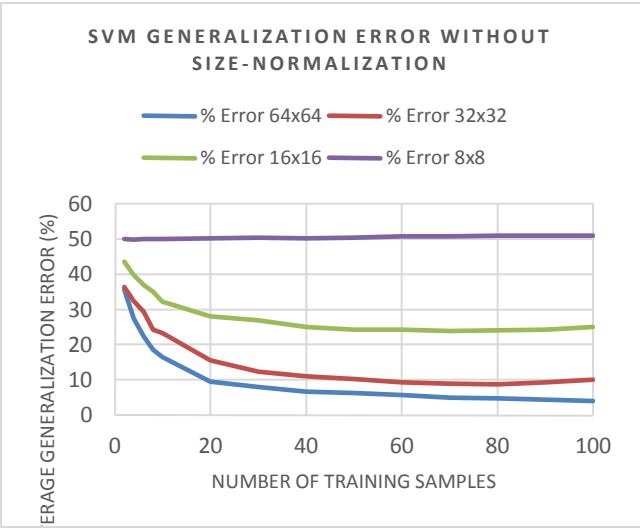

Fig.7. SVM generalization error without size-normalization and centering

Fig. 7 shows the generalization error for SVM without size-normalization and centering. The 64x64 images had the lowest generalization error of 4%, followed by 32x32 (10%), 16x16 (25%), 8x8 (51%). The 64x64 images had the lowest generalization error at 100 training samples, but the 32x32 and 16x16 images had results flattening between 60 to 100 training samples. For the 16x16 and 8x8 images, the generalization error was too high and the results were not useful to make a meaningful prediction. This result is intuitive as well since we lose too much information as we reduce the image resolution. At 8x8 resolution, the information loss is so much that it is impossible to distinguish between two images.

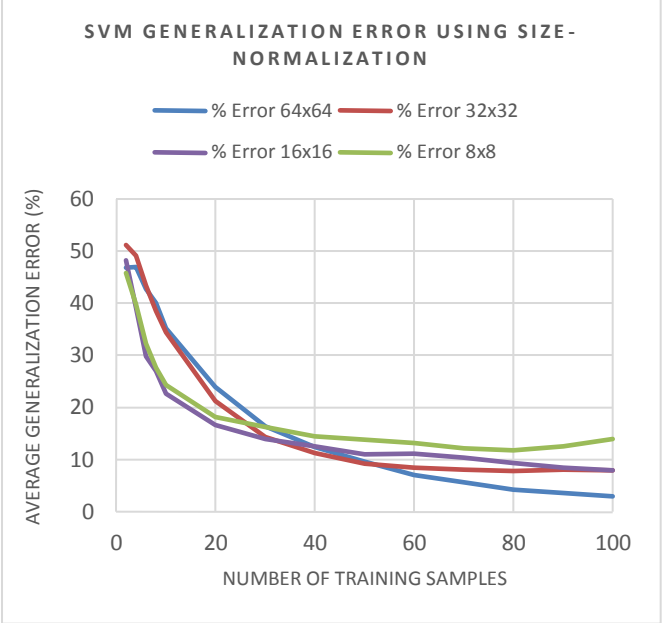

Fig. 8. SVM generalization error with size-normalization and centering

Fig. 8 shows the generalization error for SVM using size-normalization and centering. The performance was improved, especially for the low resolution images (16x16 and 8x8). The 64x64 and 32x32 images showed slight improvement as well. The results may seem odd but the key point to note here is that size normalization and centering "brings uniformity amongst chaos" in the low resolution images. When compressing the original image, there is no control over how the pixel information is truncated. The compression may lead to loss of information at different areas in the training and test image which will make it difficult for the test image to be identified. When size normalization is applied, this indeterminate loss of information is curbed to a certain extent since whatever the dimensions of the text, we center it and then scale it. This effect will be more pronounced in the samples where the text is present near one of the corners or edge of the image.

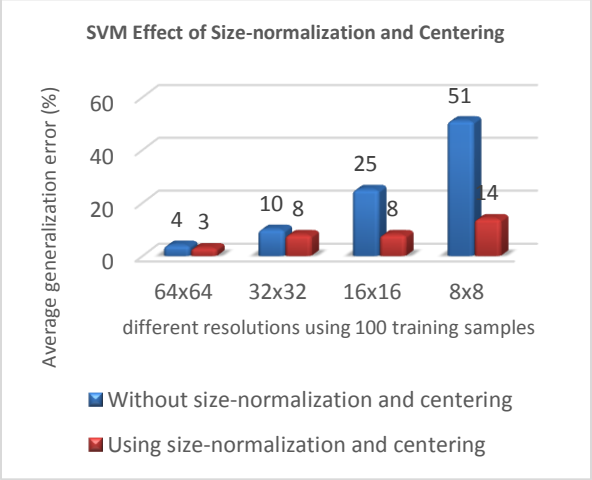

Fig. 9. SVM effect of size-normalization and centering

VII. SVM ALGORITHM FOR THREE AUTHORS

SVMs typically classify data in two classes using a separating hyperplane. However, the handwriting recognition is a multi-class problem. To solve this, we extend the SVM algorithm to a multi-class algorithm [3] by computing one classifier for each class by pitting that class against all other classes. Therefore, if we have 'k' classes in our sample space, we would need to compute 'k' classifiers. To classify a new sample, we evaluate the new sample against each classifier and choose the class whose corresponding classifier labels it as '1'. This is otherwise called one-against-all approach.

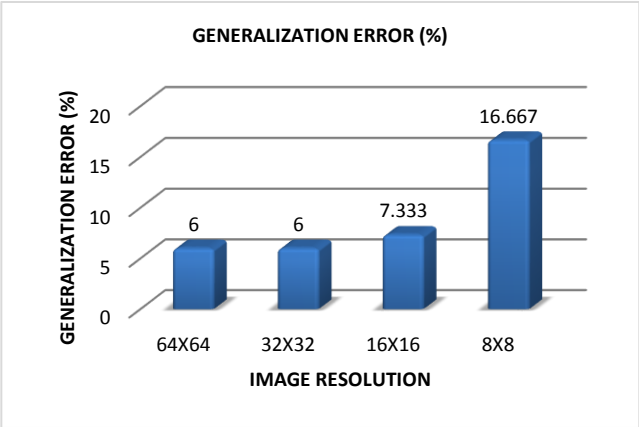

Fig.10. Multi-class SVM generalization error with size-normalization and centering

VIII. CONCLUSION

We compared the performance of the four combinations: Naïve Bayes/SVM, with/without size normalization and centering. At each image resolution and training sample size, the combination that gives the lowest generalization error is illustrated in Fig. 11. Based on this result, if the handwriting image resolution is 32x32 pixels or lower, using Naïve Bayes without size-normalization and centering generally has the best performance.

# of Training Samples	64x64	32x32	16x16	8x8
2				
4				
6				
8				
10				
20				
30				
40				
50				
60				
70				
80				
90				
100				

Legend

Naïve Bayes, no size-normalization and centering
Naïve Bayes, with size-normalization and centering
SVM, no size-normalization and centering
SVM, with size-normalization and centering

Fig.11. The best algorithm at different number of training samples and image resolutions.

At 8x8 resolution, between 20 and 90 training samples, SVM performs slightly better than Naïve Bayes, but the difference not significant. If the training samples have high resolution similar to 64x64 pixels, if there are very few training samples (< 10), using Naïve Bayes without size-normalization and centering is still the best option. Only for 64x64 images with 10 or more training samples, SVM outperforms Naïve Bayes. This result is synonymous with the results obtained in the class – Naïve Bayes is quicker to learn while performance of SVM improves as the number of training samples increases. Although SVM had a narrower range of good performance compared to Naïve Bayes, SVM achieved the absolute lowest generalization error of 3% with 64x64 training samples and using size normalization and centering.

IX. FUTURE WORK

Handwriting matching algorithms, such as this, are typically used by banks and law firms for signature matching to detect potential fraud or establish authenticity. Our signatures typically do not reflect our actual handwriting – i.e. there is a significant difference between the handwriting style of a written paragraph and a signature from the same person. In addition to this, the size of the signature varies from document to document which adds to the complexity of problem. As future work, the challenge would be to extend this algorithm to signatures. The size normalization algorithm introduced in this project needs to be modified and extended to normalize the size of signatures not only from the same person, but across different people as well. Another challenge would be to increase the learning rate – that is lower generalization error for lesser number of samples – this is an important aspect because typically organizations have to work with very few samples to determine if the signature is authentic or not.

In order to work with very few samples, it will be helpful to automatically synthesize a large number of training samples based on the real signature samples. As explored in [8], an algorithm can be developed to naturally deform the original samples to generate many more training samples.

REFERENCES

[1] INKredible [online]. Available: http://inkredibleapp.com/

[2] THE MNIST DATABASE of handwritten digits [online]. Available: http://yann.lecun.com/exdb/mnist/

[3] Literature on multi-class support vector machines - http://nlp.stanford.edu/IR-book/html/htmledition/multiclass-svms-1.html

[4] Ideas on how to implement multi-class support vector machines - http://www.codeproject.com/Articles/106583/Handwriting-Recognition-Revisited-Kernel-Support-V

[5] https://en.wikipedia.org/wiki/Edge_detection - Image boundary detection and isolation.

[6] "Background Subtraction Techniques: a review", Massimo Piccardi

[7] A. Ng, "CS229 Lecture notes", in *CS229 (Machine Learning) class*

[8] Zheng, Y., & Doermann, D. (2005, August). Handwriting matching and its application to handwriting synthesis. In Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on (pp. 861-865). IEEE.

[9] Srihari, S., Zhang, B., Tomai, C., Lee, S., Shi, Z., & Shin, Y. C. (2003, April). A system for handwriting matching and recognition. In Proc. Symposium on Document Image Understanding Technology (pp. 67-75).

[10] Srihari, S. N., Cha, S. H., Arora, H., & Lee, S. (2002). Individuality of handwriting. Journal of Forensic Sciences, 47(4), 856-872.

[11] Wang, J., Wu, C., Xu, Y. Q., & Shum, H. Y. (2005). Combining shape and physical modelsfor online cursive handwriting synthesis. International Journal of Document Analysis and Recognition (IJDAR), 7(4), 219-227.

[12] Bunke, H. (2003, August). Generation of synthetic training data for an HMM-based handwriting recognition system. In null (p. 618). IEEE.

[13] Mori, M., Suzuki, A., Shio, A., Ohtsuka, S., Schomaker, L. R. B., & Vuurpijl, L. G. (2000, September). Generating new samples from handwritten numerals based on point correspondence. In Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition (pp. 281-290).

[14] Chui, H., & Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding, 89(2), 114-141.

[15] Pervouchine, V., & Leedham, G. (2007). Extraction and analysis of forensic document examiner features used for writer identification. Pattern Recognition, 40(3), 1004-1013.

[16] Srikantan, G., Lam, S. W., & Srihari, S. N. (1996). Gradient-based contour encoding for character recognition. Pattern Recognition, 29(7), 1147-1160.

[17] Cha, S. H., & Srihari, S. (2000, September). Multiple feature integration for writer verification. In Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition (pp. 333-342).

[18] Statistics and Machine Learning Toolbox, MATLAB.