Performance evaluation of classifiers on handwritten character recognition

1Binu P Chacko
1Associate Professor
1Prajyoti Niketan College, Pudukad

Abstract - **Research on HCR including online (stroke trajectory based) and offline (image based) recognition have received intensive attention since 1960s. In online handwriting recognition, data are captured during the writing process on an electronics surfacing with a special pen. Offline HCR is a process of automatic computer recognition of characters in the optically scanned and digitized pages of text. This study concentrates on the offline recognition of handwritten Malayalam characters. It goes through major components of character recognition system such as preprocessing, feature extraction and classification. The character images are normalized and binarized during the preprocessing stage. Then, division point features are extracted from processed images and classified using SVM, ELM, and OS-ELM. Among the classifiers, SVM performed well on KUMCD database.**

keywords - **Division point feature, ELM, OS-ELM, SVM**

I. INTRODUCTION

HCR has been studied for more than fifty years to deal with challenges of large number of character classes, confusion between similar characters, and distinct handwriting styles across individuals. It is an integral part of handwritten text recognition [1]. The large variance of handwriting styles across writers is a big challenge to the robust handwritten character recognition (HCR) [2]. Writer adaptation is widely used to handle this challenge by gradually reducing the mismatch between writer independent system and particular individuals [3]. According to the type of data, handwriting recognition can be divided into online and offline. In online HCR, the trajectories of pen tip movements are recorded and analysed to identify the linguistic information expressed, while in offline HCR, character (gray scaled or binary) images are analysed and classified into different classes [1]. Algorithms designed to recognize handwritten characters are less successful than those of printed characters due to diversity in handwritten character shapes and forms [4].

HCR has wide application prospects, and there is great demand for it in industrial fields such as image recognition systems and handwritten text input devices as society develops and progresses. Especially in the field of image processing and pattern recognition, HCR has been extensively studied and developed [5]. Many, successful applications have been found in offline HCR such as document digitization and retrieval, postal mail sorting, bank cheque processing, form processing, pen based text input, historical document recognition, handwritten notes transcription, and so on [6]. Online HCR has been widely used for pen input devices, personal digital assistants, smart phones, computer aided education, etc. [1]. The methods used in handwriting recognition are divided into two categories: HCR methods based on feature extraction and pattern classification, and HCR methods that originate in deep learning [5].

Many of the existing OCRs follow the multistage pipeline structure of machine learning where different feature extraction algorithms and classifiers are employed separately [7]. These recognition systems consist of activities namely, digitization, preprocessing, feature extraction, and classification. Digitization is the process of converting a paper based handwritten document into an electronic form using a scanner or digital camera. During preprocessing, irrelevant bit patterns are detected and removed in order to increase the recognition accuracy [8]. It includes shape normalization, binarization, edge detection, skeletonization, pruning, noise removal, skew correction, etc. Shape normalization can reduce the within-class variations and hence increase the recognition accuracy. The widely used shape normalization methods are nonlinear normalization [9], bi-moment normalization [10], pseudo 2D normalization and line density projection interpolation. In this study, the line density projection interpolation (LDPI) method is used due to its superior performance [11]. The feature extraction is based on pixel level primitive features such as colour, structure, texture, projection histograms, and moment invariants. Once feature extraction has been performed, appropriate classifier is used to classify the characters. The commonly used classification methods are template matching, k-nearest neighbour algorithm, artificial neural network, support vector machine, etc. [5]. Selecting an appropriate feature extractor and classifier is a major concern in these schemes.

Several works have been reported in the literature on HCR. Wen et al. [12] have proposed a novel kernel and Bayesian discriminant based classifier for recognition of Bangla numerals. Wen et al. [13] have shown the use kernel PCA and SVM on recognition of Bangla numeral from the postal pin. Liu et al. [14] have extensively studied various gradient direction histogram features with classifiers like DLQDF, PNC, CFPC, and SVM on benchmark datasets namely, ISI-Kolkata Bangla numerals, CENPARMI Farsi numerals, and IFHCDB Farsi numerals. Similarly, Bhalerao et al. [15] have designed a classifier model with strength, angle, and histogram of gradient (SOG, AOG, HOG) feature and SVM classifier for Devanagari characters.

Dash et al. [16] have proposed a hybrid feature based on Kirsh edge operator and curvature feature for modified quadratic discriminate function (MQDF) and DLQDF (discriminative learning QDF) classifier. They have also studied on zone based non-redundant socketwell transform and *k* nearest neighbour classifier for Odia numeral recognition [17]. Mishra et al. [18] have proposed contour based feature on HMM classifier. Similarly, Sethy et al. [19] have used texture feature of GLCM matrix evaluated on 2-level DWT image. They have also used 2-level DWT decomposition with various statistical features [20].

II. Character Databases

The character databases are an inevitable part of any research on HCR. It is necessary to compare the performance new works on standard databases. The databases contain character images in wide variety of handwriting styles. As it is well known that the difference in writing styles due to many factors such as age, culture, degree of education and origin result in a very large shape variability of characters [21].

Chinese handwriting database (both online and offline) released by CASIA contains isolated character samples and handwritten texts produced by 1020 writers. It involves 7,356 character classes (7,185 Chinese characters, 171 alphanumerics and symbols) and 5,090 text pages. The outstanding features of this database are unconstrained writing, concurrent online and offline data, combination of isolated samples and continuous scripts, deep annotation of script data, large category set, large number of writers and samples (gray scale images) [22].

Torki et al. [23] created AlexU Isolated Alphabet (AIA9k) database containing about 9,000 Arabic characters of 28 classes written by 107 persons. On this database they have used feature extraction techniques such as Histogram of Oriented Gradients (HOG), Speeded-Up Robust Features (SURF), and Scale Invariant Feature Transform (SIFT), and classifiers like ANN, logistic regression, SVM-linear and SVM-RBF [4]. HACDB is another Arabic character database containing 6,600 samples of 66 different classes written by 50 persons [24]. El-Sawi et al. [25] collected Arabic Handwritten Character Dataset (AHCD) of 16,800 images of isolated characters written by 60 participants.

ISI-Kolkata Bangla numeral dataset includes 23,392 samples from 1106 individuals [26]. ISI-Kolkata Odia numeral dataset accommodates 5,970 samples from 356 individuals [27]. NIT-RKL Bangla numeral dataset contains 4,345 samples collected from 150 students [7]. MNIST contains 70,000 gray scale images of handwriting digits centered on a 28 x 28 square canvas. For this study, Kannur University Malayalam Character Database (KUMCD) is used, which contains 14,700 gray scale images of 49 Malayalam characters collected from 690 writers [28].

Data augmentation techniques are a way to artificially expand the dataset. It was deemed necessary to improve the network performance. Some popular augmentation examples are horizontal flips, vertical flips, random crops, translations and rotations [4].

III. Feature Extraction

In the framework of handwriting recognition, the large variability of handwriting of different writers makes the selection of appropriate feature sets even more complex and have been widely investigated. The representation of characters belonging to different alphabets may require the use of different features [21]. The following are some of the feature extraction techniques used in character recognition domain.

The feature vector can be generated from a segmented image related to the concavity, contour, and the character surface [21]. In this method, character image is split into N x M zones of equal size. The purpose of zoning is to attain the neighbourhood traits [8]. For each zone, 13 concavity measurements are computed using 4-Freeman directions as well as other four auxiliary directions, normalized between 0 and 1. Then, in each zone, eight contour features are extracted from a histogram of contour direction obtained by grouping the contour line segments between neighbouring pixels based on 8-Freeman direction. The density of foreground pixels in each zone is calculated to produce the feature vector related with the character surface. Character images can also be described using features such as Fourier coefficients, Zernike moments, morphological features, Karhunen-Love coefficients, pixel averages in n x m windows, profile correlations, etc. [21].

Gradient direction feature can be extracted from both binary images and gray scale images. For *binary images*, normalization based gradient feature (NBGF) and normalization cooperated gradient feature (NCGF) is extracted. In either case, gradient elements are decomposed into eight directions and each direction extracted 8 x 8 values by Gaussian blurring. The gradient is computed by Sobel operator and its direction is decomposed into its two adjacent standard chaincode directions by the parallelogram rule. By NCGF, the normalized character image is not generated, but instead, the gradient elements of the original image are directly mapped to direction maps incorporating pixel coordinates transformation. By NBGF, the features are extracted from the normalized image. For *gray scale image*, gradient feature extraction methods NBGF and NCGF are directly applicable [6].

The division point feature related with character surface is used in this study. This method relies on iterative subdivision of the character image so that the resulting sub-images at each iteration have balanced number of foreground pixels. In the first iteration step (L = 0), the character image is subdivided into four rectangular sub-images using a horizontal and a vertical divider line. Initially, a vertical line is drawn that minimizes the absolute difference of the number of number of foreground pixels in the two sub-images to its left and to its right. Subsequently, a horizontal line is drawn that minimizes the absolute difference of the number of foreground pixels in the two sub-images above and below it. The pixel at the intersection of two lines is referred to as division point. In the succeeding steps, each sub-image obtained at the previous step is further divided into four sub-images using the same procedure. The coordinates (x_i, y_i) of all division points at the last level (i.e., 4^L division points) are stored as the feature [29].

The use of huge quantity of features may produce decay in the efficiency of learning algorithms, especially in the presence of irrelevant or redundant features. Thus feature selection methods are used for searching the optimal subset of features from the whole set of available ones, in order to obtain best recognition results. The feature selection process typically consists of three basic steps: a search procedure for searching candidate feature subsets, a feature subset evaluation strategy and a stopping criterion. The search procedure is repeated until the stopping criterion is satisfied. To manage the increasing number of available features, a large number of feature selection methods have been proposed [21].

IV. CLASSIFICATION

The classification schemes such as MQDF, NPC, DFE, GAN, AFL, k-NN, Bagging and Random Forest are discussed in this section. The MQDF is a modification of the multivariate Gaussian based QDF by regulating the minor eigenvalues of each class to a constant, such that the discriminate function can be calculated from the principal eigenvalues and their corresponding eigenvectors only, and the regulation of minor eigenvalues benefits the generalization performance. The nearest prototype classifier (NPC) has lower runtime computation cost than MQDF when using few prototypes per class. There are many algorithms for prototype learning, including clustering, learning vector quantization (LVQ) and many generalized discriminative learning methods. The discriminative feature extraction (DFE) is a discriminative subspace learning method, mostly combined with prototype learning: the subspace is optimized jointly with the prototypes by stochastic gradient descent, usually under the minimum classification error (MCE) criterion. The DLQDF is a discriminative version of MQDF, with the parameters (class means, eigenvectors and eigen values) optimized by stochastic gradient descent under the MCE objective [6]. Besides the DLQDF, some alternative improved versions of MQDF are proposed, including the MQDF with pairwise discrimination [30], compact MQDF with subspace parameter compression [31], compressed and discriminatively trained Gaussian model [32], and MQDF estimated from reweighted samples [33]. All these methods effect in improving the classification accuracy of MQDF or reducing the storage and computation complexity.

k-Nearest Neighbour (k-NN) algorithm is a well known nonparametric method that can be used for both regression and classification. According to this approach, an unknown sample is labelled with the most common label among its *k* nearest neighbours in the training set. The rationale behind k-NN classifier is that, given an unknown sample **x** to be assigned to one of the c_i classes of the problem at hand, the a-posteriori probabilities $p(c_i|\mathbf{x})$ in the neighbourhood of **x** may be estimated by looking at the class labels of the k nearest neighbours of **x**.

Bagging generates multiple training sets (denoted as bootstrap sets) by sampling the original training set with replacement uniformly. The effect is that, in these sets, some of the original samples can be repeated while other may be left out. Each bootstrap set is used to train a different component classifier and the final classification decision is based on the vote of each component classifier.

Random Forest refers to a family of methods for building an ensemble of tree based classifiers.

Algorithm (applied to each tree)

Given a training set of N feature vectors, and each consisting of M features.

1. Draw N samples from the dataset at random with replacement. The resulting set will be the training set associated with the starting node of the tree.
2. Set a number K << M
3. At each node, randomly draw K features from the set of available ones.
4. For each of the K features drawn, consider its value in the training set and choose the best binary split value according to the Gini index [8.15], then select the feature with the best index value and generate two new nodes by splitting the samples associated with the original node according to such a value.
5. Grow the tree to its maximum size according to the stopping criterion chosen.
6. Leave the tree unpruned.

Once forest has been built, an unknown sample is labelled according to the Majority Vote rule: i.e., it is labelled with the most popular class among those provided by the ensemble trees. Random Forest does not over-fit as more trees are added, but rather its generalization error tends to a limiting value [21].

Generative adversarial network (GAN) is a well known adversarial learning model [9.10]. It is composed of a generator and a discriminator. The discriminator can guide the generator to adjust a complex data distribution to another specific distribution by adversarial learning. When GAN is performed on a handwritten character set, the generator could transfer noise vectors to realistic character images [2].

Adversarial feature learning (AFL) is a variant of GAN and composed of a feature extractor, a discriminator, and a classifier. The feature extractor is used to extract encoding features of handwritten and printed characters. The discriminator judges whether the extracted features come from handwritten or printed characters. With the prior knowledge provided by printed characters, it can guide the feature extractor to exploit writer-independent semantic features from handwritten characters automatically. Finally, the extracted features are fed into the classifier to recognize the handwritten characters [2].

The recognition process goes through two different stages: training and testing. A large number of samples are required at the training stage. The lack of labelled training data can be overcome by generating new training samples from the existing samples, with realist augmentations which reflect actual variations that are present in handwriting, by adding random controlled noise to their corresponding instantiation parameters [34]. Affine transformation [35] and distorted generation [36] are the other alternatives for data augmentation.

In the following, the classifier such as ELM, OS-ELM and SVM used in this study are discussed.

Extreme Learning Machine

Neural networks (NN) and its variants have been extensively used in the literature for solving various regression and classification tasks due to its generalization and universal approximation ability. Gradient based methods are the most commonly used methods for training NNs, however, these iterative methods have several major pitfalls such as slow convergence, local minima issue and overfitting problem [7]. Dropout is a widely used strategy to avoid overfitting and improve the generalization performance [37]. Randomized algorithms for training single hidden layer feedforward neural networks (SLFN) such as extreme learning machine (ELM) [38] and radial basis function network (RBFN) [39] have become a popular choice in recent years because of their generalization capability with faster learning speed. These methods use least square method (Moore-Penrose generalized inverse) to approximate the weights from hidden to output layer and require no parameter turning. They involve two phases for learning: 1) random feature mapping, and 2) output weights computation using an analytical method. These algorithms differ in feature mapping phase – ELM uses random feature mapping (weights from input to hidden layer are generated randomly), whereas RBFN uses distance based random feature mapping (centers of RBFs are generated randomly). However, RBFN obtain unsatisfactory solution for some cases and results in poor generalization [40]. In contrary, ELM provides effective solution for SLFNs with good generalization and extremely fast learning, thereby, has been widely applied in various applications like regression, classification, image segmentation, dimension reduction, etc. [7].

Several variants of ELM such as E-ELM, I-ELM, P-ELM, S-ELM, CI-ELM, CS-ELM, EI-ELM, EM-ELM, KS-ELM, OP-ELM, OS-ELM, TS-ELM, KOS-ELM and RCGA-ELM have been proposed in the literature [28]. The study conducted by Xu et al. [41] shows improvement in performance of ELM by optimizing the input weights and biases whilst requiring less number of hidden nodes. Das et al. have conducted a study to derive some best ELM configurations for designing ELM based classifiers [7]. The accuracy and generalization capability of ELM highly depends on the learning of output weights and minimization of the output weight norm. The approximation problem can be expressed as follows [38].

For N arbitrary distinct samples ($\mathbf{x}_i$, $\mathbf{t}_i$), where $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in R^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in R^m$, SLFN with Ň hidden nodes and activation function g(x) are mathematically modelled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = o_j \qquad (1)$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector connecting the ith hidden node and input nodes, and $[\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and output neurons, and b_i is the threshold of ith hidden node. SLFN with Ň hidden nodes and activation function g(x) can approximate these N samples with zero error means that $\sum_{j=1}^{\tilde{N}} \left\| o_j - t_j \right\| = 0$, i.e., there exist β_i, $\mathbf{w}_i$, and b_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j, \quad j = 1, 2, ..., N \qquad (2)$$

The above N equations can be written compactly as

$$\mathbf{H}\beta = T \qquad (3)$$

where

$$H(w_1,...,w_{\tilde{N}}, b_1,...,b_{\tilde{N}}, x_1,...,x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \dots & \vdots \\ g(w_1 \cdot x_N + b_1) & \dots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{Nx\tilde{N}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N}xm} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{Nxm}$$

H is called hidden layer output matrix, its ith column is the ith hidden node output with respect to inputs $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$. If the activation function g is infinitely differentiable, we can prove that the required number of hidden nodes Ň ≤ N.

Algorithm

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, 2, ..., N\}$, activation function g(x), and number of hidden nodes Ň.
Step 1: Randomly assign input weights $\mathbf{w}_i$ and bias b_i, i = 1, 2, ..., Ň.
Step 2: Calculate the hidden layer output matrix **H**.
Step 3: calculate the output weight β,
$\beta = \mathbf{H} \quad \mathbf{T}$ where $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N]^T$

ELM tends to reach the smallest training error and smallest norm of output weights in one epoch. However, the generalization ability of the network depends on the choice of arbitrary weight initialization schemes, number of hidden nodes and the type of activation functions used. ELM initializes the hidden weights and biases randomly, and keeps them throughout the learning process; however, the generalization of the output weight depends on the previous layer initialization. Activation functions help to decide whether a node is activated or disabled. It also helps to limit the output to a finite range. In general, any continuously differentiable function that exists over a range (-inf, inf) can be used as the activation function. However, all activation functions results in different approximation results [7].

Online Sequential Extreme Learning Machine (OS-ELM)

The learning based on ELM requires all the data to be present at the beginning of training. However, the training data may arrive one-by-one or block-by-block in real applications. To cope with this situation, a sequential learning variant of ELM called OS-ELM can be used [42]. OS-ELM uses the ideas of batch learning ELM. It consists of two phases – initialization phase and sequential learning phase. In the initialization phase, matrix $\mathbf{H}_0$ is filled up for use in the learning phase.

Step 1) Initialization phase: Initialize the learning using a block of training data $\aleph_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$ from the given training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, 2, ..., N\}, N_0 \geq \check{N}$.

a) Assign random input weight $\mathbf{w}_i$ and bias $\mathbf{b}_i$ (for additive hidden nodes) or centre $\mathbf{w}_i$ and impact factor $\mathbf{b}_i$ (for RBF hidden nodes), i = 1, 2, ..., Ň.
b) Calculate initial hidden layer output matrix $\mathbf{H}_0$.
c) Estimate the initial output weight $\beta^{(0)}$.
d) Set k = 0.

Step 2) Sequential learning phase: Present (k+1)th block of new observations $\aleph_{k+1}$.

a) Calculate partial hidden layer output matrix $\mathbf{H}_{k+1}$ for the (k+1)th block of data $\aleph_{k+1}$.
b) Set T_{k+1}.
c) Calculate the output weight $\beta^{(k+1)}$.
d) Set k = k + 1. Repeat step 2.

Support Vector Machine (SVM)

SVM is a system for efficiently training the linear learning machines in the kernel induced feature space. It is based on statistical learning theory and quadratic programming optimization. An SVM is basically a binary classifier and multiple SVMs can be combined to form a system for multiclass classification. There may be instance in which two classes are not linearly separable. In such cases, one prefers nonlinear mapping of data into some higher dimensional space called feature space where it is linearly separable. SVMs try to find the optimal decision boundary by maximizing the margin (using a central hyperplane) between the boundaries (bounding planes) of different classes. To do so, they identify those instances of each class that define the boundary of that class in the feature space. These instances are called support vectors. In fact, the novelty of SVMs is in their ability to use a kernel function for mapping into a high dimensional space, and find support vectors in that high dimensional space. What makes SVMs even more popular is the fact that they do it through a procedure affectionately known as kernel trick, which does not require any calculation in the high dimensional space.

The basic concept behind SVM is to search for a balance between the regularization term $(1/2)\mathbf{w}^T\mathbf{w}$ and the training errors. Minimization of $C\mathbf{e}^T\xi + (1/2)\, \mathbf{w}^T\mathbf{w}$ w.r to $\mathbf{w}$ and ξ causes maximum separation between bounding planes with minimum number of points crossing their respective bounding planes. So the formulation is $\min_{w,\xi,C} Ce^T\xi + \frac{1}{2}w^T w$

Since the objective function is quadratic, find the solution through quadratic programming. The formulation can also be written as $\min_{w,\xi,b} \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i$ (4)

subject to $t_i(w^T\varphi(x_i) - b) + \xi_i - 1 \geq 0, \quad \xi_i \geq 0, 1 \leq i \leq N$

There are two different approaches for multiclass classification by SVM – direct method and indirect method. In indirect method, several binary SVMs are constructed, and combined the outputs to find the final class, whereas the direct approach considers all the data in a single optimization formulation. Indirect methods are widely used, and they are classified into three types: one against one (1-v-1), one against all (1-v-r), and DAGSVM. In this study DAGSVM is used, which places 1-v-1 SVMs on the decision directed acyclic graph (DDAG) nodes and combines their results to find the class. Let $\mathbf{w}$ be the weight vector correctly splitting *i* and *j* classes at the *ij* node with threshold θ. The margin of *ij* node is defined as $\delta = \min_{c(x)=i,j}\{|\langle w, x\rangle - \theta|\}$, where c(x) is the class associated to training example $\mathbf{x}$. Maximizing the margin of all nodes in a DDAG will minimize a bound on the generalization error. Therefore, a DDAG has to be created whose nodes are maximum margin classifiers over a kernel induced feature space. Such a DDAG is obtained by training each *ij* node only on the subset of training points labelled by *i* or *j*. The DDAG partitions the input space into polytopic regions, each of which is mapped to a leaf node and assigned to a specific class [28].

V. EXPERIMENTAL RESULTS

In this character recognition problem, KUMCD database is used for classification experiments. It is divided into training set and test set in the ratio 3:1. From each normalized and binarized image, division point feature is extracted at level 2. The feature set is then evaluated using ELM, OS-ELM, and SVM classifiers. The number of hidden nodes is taken as 2000 and the size of initial and succeeding blocks of OS-ELM are 4,500 and 500 respectively. The generalization performance of classifiers tends to become worse when too few or too many nodes are randomly generated. The choice of kernel and the regularizing parameter C for SVM is determined via performance on a validation set. The generalization accuracy is estimated using different kernel parameters γ and cost parameter C: $\gamma = [2^4, 2^3, ..., 2^{-10}]$ and $C = [2^{12}, 2^{11}, ..., 2^{-2}]$. Based on the experiments RBF kernel is selected, and same (C, γ) is used for all k(k-1)/2 binary classifiers. The performance of all the three classifiers on KUMCD is given in Table 1. SVM is able to produce highest recognition accuracy 94.12% using division point feature.

Table 1: Recognition Accuracy (%) of Classifiers Using Division Point Feature

Classifier	Training time (s)	Testing time (s)	Accuracy (%)
ELM	213	1.69	92.95
OS-ELM	356	2.09	93.58
SVM	18	31.28	**94.12**

VI. CONCLUSION

HCR is a fundamental step towards document analysis and recognition. Even though many techniques used in different languages are same, domain specific feature extraction is also important. Among the available feature extraction methods and classifiers, it is necessary to find out the right combination for a specific problem. At the same time, more studies are required to develop new techniques for character recognition. It is intend to do the future study in this direction.

REFERENCE

[1] X Y Zhang, Y Bengio, C L Liu, “Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark”, Pattern Recognition, 61, pp. 348-360, 2017.

[2] Y Zhang, S Liang, S Nie, W Liu and S Peng, “Robust offline handwritten character recognition through exploring writer independent features under the guidance of printed data”, Pattern recognition letters, 106, pp. 20-26. 2018.

[3] S Connel and A Jain, “Writer adaptation for online handwriting recognition”, IEEE Trans. Pattern analysis and machine learning, 24(3), pp. 329-346, 2002.

[4] K S Younis, “Arabic handwritten character recognition based on deep convolutional neural networks”, Jordanian J. Of Computers and Information Technology, 3(3), pp. 186-198, 2017

[5] Z Rao et al., “Research on handwritten character recognition algorithm based on extended nonlinear kernel residual network”, KSII Trans. Internet and information systems, 12(1), pp. 413-432, 2018.

[6] C L Liu, F Yin, D H Wang and Q F Wang, “Online and Offline handwritten Chinese character recognition: Benchmarking on new databases”, Pattern recognition, 46, pp. 155-162, 2013.

[7] D Das, D R Nayak, R Dash and B Majhi, “An empirical evaluation of extreme learning machine: Application to handwritten character recognition”, Multimedia tools appl, 78, pp. 19495-19523, 2019.

[8] H Kaur and S Rani, “Handwritten Gurumukhi character recognition using CNN”, Int. J. Computational intelligence research 13(5), pp. 933-943, 2017.

[9] J Tsukumo and H Tanaka, “Classification of hand printed Chinese characters using nonlinear normalization and correlation methods”, Proc. Int. Conf. Pattern recognition, pp. 168-171, 1988.

[10] C L Liu, H Sako and H Fujisava, “Handwritten Chinese character recognition: alternatives to nonlinear normalization”, Proc. Int. Conf. Document analysis and recognition, pp. 524-528, 2003.

[11] C L Liu and K Marukawa, “Pseudo two dimensional shape normalization methods for handwritten Chinese character recognition”, Pattern recognition, 38(12), pp. 2242-2255, 2005.

[12] Y Wen and L He, “A classifier for Bangla handwritten numeral recognition”, Expert systems with applications, 39(1), pp. 948-953, 2012.

[13] Y Wen, Y Liu and P Shi, “Handwritten Bangla numberal recognition system and its application to postal automation”, Pattern recognition, 40(1), pp. 99-107, 2007.

[14] C L Liu and C Y Suen, “A new benchmark on the recognition of handwritten Bangla and Farsi and numeral characters”, Pattern recognition, 42(12), pp. 3287-3295, 2009.

[15] M Bhalerao, S Bonde, A Nandedkar and S Pilawan, “Combined classifier approach for offline handwritten Devanagari character recognition using multiple features”, Computer vision and bio-inspired computing, pp. 45-54, 2018.

[16] Dash K S, Puhan N and Panda G, “A hybrid feature and discriminant classifier for high accuracy handwritten Odia numeral recognition”, IEEE region 10 Sym., pp. 531-535, 2014.

[17] Dash K S, Puhan N and Panda G, “Non-redundant stockwell transform based feature extraction for handwritten digit recognition”, Proc. Int. Conf. Signal processing and communication, pp. 1-4. 2014.

[18] Mishra T K, Majhi B, Sa P K and Panda S, “Model based Odia numeral recognition using fuzzy aggregated features”, Frontiers of computer science, 8(6), pp. 916-922, 2014.

[19] Sethy A, Patra P K and Nayak D R, “Gray level co-occurrence matrix and random forest based offline Odia handwritten character recognition”, Recent patents on engineering, 2018.

[20] Sethy A, Patra P K and Nayak D R, “Offline handwritten Odia character recognition using DWT and PCA”, Progress in advanced computing and intelligent engineering, pp. 187-195, 2018.

[21] Cilia N D, Stephano C D, Fontanella F and Freca A S, “A ranking based feature selection approach for handwritten character recognition”, 121, pp. 77-86. 2019.

[22] Liu C L, Yin F, Wang D H and Wang Q F, “CASIA online and offline Chinese handwriting databases”, Proc. ICDAR, pp. 37-41, 2011.

[23] Torki M, M E Hussein, A Elsallamy, M Fayyaz and S Yasar, “Window based descriptors for Arabic handwritten alphabet recognition: A comprehensive study on a novel dataset”, arXiv: 1411.3519, 2014.

[24] Lawgali A et al, “DACDB: Handwritten Arabic characters’ database for automatic character recognition”, European Work. visual information processing, pp. 255-259, 2013.

[25] El-Sawy A, Loey M and El-Bakry H, “Arabic handwritten character recognition using convolution neural network”, WSEAS Trans. Computer research, 5, pp. 11-19, 2017.

[26] Bhattacharya U and Chaudhuri B B, “Handwritten numeral databases for Indian scripts and multistage recognition of mixed numerals”, IEEE Trans. Pattern analysis and machine intelligence, 31(3), pp. 444-457, 2009.
[27] Bhattacharya U and Chaudhuri B B, “Databases for research on recognition handwritten characters of Indian scripts”, Proc. ICDAR, pp. 789-793, 2005.
[28] Binu P Chacko, ICR: A study and analysis of ELM and SVM using division point and wavelet features, Ph.D thesis, 2011.
[29] G Vamvakas, B Gatos and S J Perantonis, “Handwritten character recognition through two stage foreground sub-sampling”, Pattern recognition, 43, pp. 2807-2816, 2010.
[30] T F Gao and C L Liu, “High accuracy Chinese character recognition using LDA based compound distances”, Pattern recognition, 42(11), pp. 3442-3451, 2018.
[31] T Long and L Jin, “Building compact MQDF classifier for large character set recognition by subspace distribution sharing”, Pattern recognition, 41(9), pp. 2916-2925, 2008.
[32] Y wang and Q Huo, “Building compact recognizers of handwritten Chinese characters using precision contrained Guassian model, minimum classification error training and parameter compression”, Int. J. Document analysis and recognition, 14(3), pp. 255-262, 2011.
[33] Y Wang, X Ding and C Liu, “MQDF discriminative learning based offline handwritten Chinese character recognition”, Proc. ICDAR, pp. 1100-1104, 2011.
[34] V. Jayasundara, S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne and R. Rodrigo, "TextCaps: Handwritten Character Recognition With Very Small Datasets", IEEE Winter Conf. Applications of Computer Vision, pp. 254-262, 2019.
[35] H Miyao and M Maruyama, “Virtual example synthesis based on PCA for offline handwritten character recognition”, Document analysis systems, 7, pp. 96-105, 2006.
[36] K Leung and C H Leung, “Recognition of handwritten Chinese characters by combining regularization, fisher’s discriminant and distorted sample generation”, Proc. ICDAR, pp. 1026-30, 2009.
[37] N Srivastava, G Hinton, A Krishevesky, I Sutskever and R Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, J. Machine learning and research, 15(1), pp. 1929-1958, 2014.
[38] G B Huang, QY Zhu and C K Siew, “Extreme learning machine: theory and application”, Neurocomputing, 70(1-3), pp. 489-501, 2006.
[39] D S Broomhead and D Lowe, “Radial basis functions, multivariable functional interpolation and adaptive networks”, Royal signals and radar establishment Malvern, 1988.
[40] D Wang, “Randomized algorithms for training neural networks”, Information sciences, pp. 364-365, 2016.
[41] Y Xu and Y Shu, “Evolutionary extreme learning machine based on particle swam optimization”, Proc. Int. Sym. Neural networks, pp. 644-652, 2006.
[42] N Y Liang, G B Huang, P Saratchandran and N Sundararajan, “A fast and accurate online sequential learning algorithm for feedforward networks”, IEEE Trans. Neural Networks, 17(6), pp. 1411-1423, 2006.