PeerJ Computer Science

Optical character recognition system for Baybayin scripts using support vector machine

Rodney Pino, Renier Mendoza and Rachelle Sambayan

Institute of Mathematics, University of the Philippines Diliman, Quezon City, Metro Manila, Philippines

ABSTRACT

In 2018, the Philippine Congress signed House Bill 1022 declaring the Baybayin script as the Philippines' national writing system. In this regard, it is highly probable that the Baybayin and Latin scripts would appear in a single document. In this work, we propose a system that discriminates the characters of both scripts. The proposed system considers the normalization of an individual character to identify if it belongs to Baybayin or Latin script and further classify them as to what unit they represent. This gives us four classification problems, namely: (1) Baybayin and Latin script recognition, (2) Baybayin character classification, (3) Latin character classification, and (4) Baybayin diacritical marks classification. To the best of our knowledge, this is the first study that makes use of Support Vector Machine (SVM) for Baybayin script recognition. This work also provides a new dataset for Baybayin, its diacritics, and Latin characters. Classification problems (1) and (4) use binary SVM while (2) and (3) apply the multiclass SVM classification. On average, our numerical experiments yield satisfactory results: (1) has 98.5% accuracy, 98.5% precision, 98.49% recall, and 98.5% F1 Score; (2) has 96.51% accuracy, 95.62% precision, 95.61% recall, and 95.62% F1 Score; (3) has 95.8% accuracy, 95.85% precision, 95.8% recall, and 95.83% F1 Score; and (4) has 100% accuracy, 100% precision, 100% recall, and 100% F1 Score.

Subjects Artificial Intelligence, Computer Vision, Natural Language and Speech, Optimization Theory and Computation, Scientific Computing and Simulation
Keywords Baybayin, Latin script identification, Baybayin script identification, Support vector machine, Optical character recognition

Submitted 22 July 2020
Accepted 23 December 2020
Published 15 February 2021

Corresponding author
Renier Mendoza,
rmendoza@math.upd.edu.ph

Academic editor
Arkaitz Zubiaga

Additional Information and Declarations can be found on page 20

DOI **10.7717/peerj-cs.360**

INTRODUCTION

Baybayin is one of the pre-Hispanic writing systems used in the Philippines (*Cabuay, 2009*). In 2018, the Philippine government is showing efforts to preserve and reintroduce this heritage, mandating all local government units to inscribe Baybayin script with its translation in their communication systems (e.g., signage), through House Bill 1022. Furthermore, local manufacturers are required to use Baybayin on labels, and the Education Department is tasked to promote the said writing system (*Lim & Manipon, 2019*).

Baybayin is an abugida, or alphasyllabary, primarily used by the Tagalogs in northern Philippines during the pre-colonial period. Baybayin consists of 17 unique characters: 14 (syllabic) consonants and three vowels (see Fig. 1A). In Baybayin, consonants are pronounced with an inherent vowel sound ' \a\', and diacritics are used to express the other vowels. For example, a diacritic written above a consonant character may represent

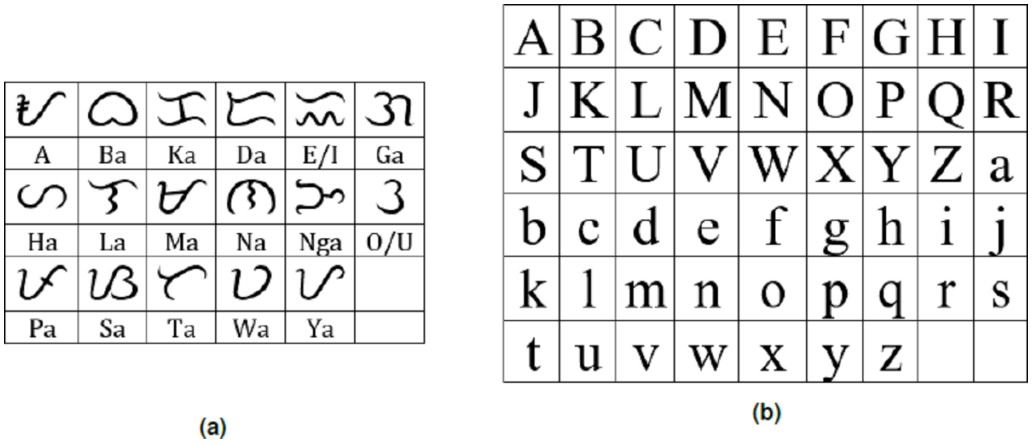

Figure 1 **(A) Baybay in characters (without diacritics) with equivalent Latin syllable; (B) Latin alphabet in upper- and lowercase.**

Full-size DOI: 10.7717/peerjcs.360/fig-1

an accompaniment vowel ‘ \e\’ or ‘ \i\’, while a diacritic written below may represent an ‘ \o\’ or ‘ \u\’ sound. Diacritics can also be used to silence the vowel sounds. Figure 2 shows an example of the phonetic distinction in a Baybayin consonant character using diacritical marks. While there are no standard accent symbols for Baybayin script, the most commonly used are bar, dot, cross, or x. A bar or a dot represents the vowels E/I or O/U based on their placement, while the cross or x symbol located below the character cancels the vowel “a” (*Cabuay, 2009*).

Optical Character Recognition (OCR) is a process of reading and recognizing handwritten or printed characters. It belongs to the family of machine recognition techniques where the system performs an automatic identification of scripts and characters (*Chaudhuri et al., 2017*).

Prior to choosing an appropriate character recognition algorithm, it is important to determine first the script used in writing a document. Most research studies in script identification have considered printed and handwritten scripts, and are based on several levels: page, line, word, and character level (*Ghosh, Dube & Shivaprasad, 2010*). With English as a global language and Latin as its standard script, there have been a lot of script identification techniques that discriminate the Latin script from other writing systems. *Jaeger, Ma & Doermann (2005)* have identified the Latin script from Arabic, Chinese, Hindi, and Korean scripts using modified document spectrum algorithm and four classifier systems at word level. At text block level script identification, *Zhou, Lu & Tan (2006)* have distinguished Bangla against Latin script using connected component analysis. Several studies by *Chanda, Pal & Kimura (2007)* about script recognition include: identifying Japanese from Latin script using a single text line scheme based on tree classifier, recognizing Thai from Latin script using a single text line scheme based on Support Vector Machine (SVM) (*Chanda, Terrades & Pal, 2007*), and recognizing Latin from all Han-based scripts (Chinese, Japanese, and Korean) via a multi-script OCR system that uses chain-code histogram-based directional features and SVM at character and block level (*Chanda et al.,*

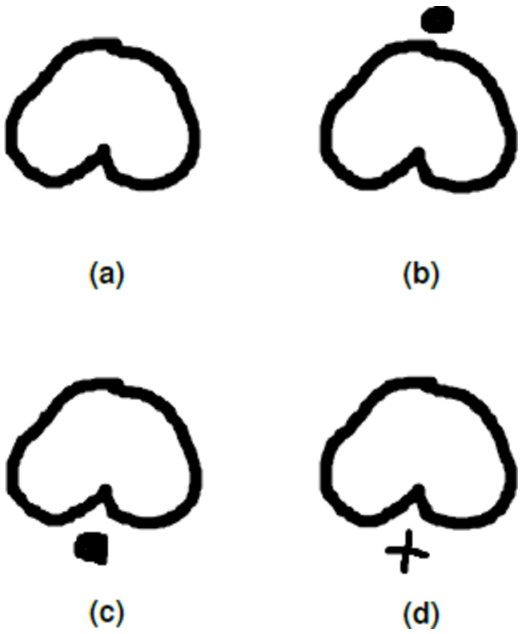

Figure 2 Incorporating diacritics in a Baybayin (consonant) character. The dot diacritic changes the 'Ba' character in A into 'Be' or 'Bi' character if written above the character (B), and into 'Bo' or 'Bu' if written below the character (C). A cross diacritic written below the 'Ba' character cancels the vowel 'a' (D).
Full-size DOI: 10.7717/peerjcs.360/fig-2

2010). At character level, *Zhu et al. (2009)* have reported recognition of Chinese script from Latin script using feature selection and cascade classifier, while *Rani, Dhir & Lehal (2013)* have proposed a technique using Gabor filter and gradient-based features using SVM classifier to recognize Gurumukhi and Latin scripts. At word level, on the other hand, *Hangarge, Santosh & Pardeshi (2013)* have reported script recognition based on directional discrete cosine transforms (D-DCT) to six Indic scripts, namely, Devanagari, Kannada, Telugu, Tamil, Malayalam, and Latin. *Rajput & Ummapure (2017)* have employed Scale Invariant Feature Transportation (SIFT) approach and $K-$Nearest Neighbor (KNN) classifier to identify Latin, Kannada, and Devanagari scripts. Multi-script identification at page level based on features extracted via Structural and Visual Appearance (SVA) and Directional Stroke Identification (DSI) has been proposed by *Obaidullah et al. (2017)*. Recognition of eleven Indian scripts together with the Latin script has been carried out using Multilayer Perceptron (MLP) and Simple Logistic (SL) classifiers. Script identification for Chinese, Kannada, Greek, Latin, and other scripts in natural scene text images and video scripts has been done by extracting global and local features, with the use of attention-based Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) Network (*Bhunia, Konwer & Bhunia, 2019*).

Among the many varieties of the OCR algorithm, the SVM classifier is one of the most popular because of its high response speed, robustness, and good accuracy (*Thomé, 2012*). The SVM algorithm was first introduced by Vladimir Vapnik and Alexey Chervonenkis in 1963. It belongs to the family of supervised classification techniques based on statistical learning theory. SVM algorithm has been very well-developed for the past decade where it

seeks the optimal hyperplane that maximizes the margins between the borders of two classes (*Thomé, 2012*). For an extensive discussion on how SVM works, we refer the readers to *Cristianini & Shawe-Taylor (2000)*; *Shawe-Taylor & Cristianini (2004)*; *Bishop (2006)*, and *Schölkopf & Smola (2012)*. Several applications of SVM include face and object detection and recognition, image information retrieval, time series prediction (*Sapankevych & Sankar, 2009*), speech recognition (*Ganapathiraju, Hamaker & Picone, 2004*), data mining (*Nayak, Naik & Behera, 2015*), bioinformatics (*Yang, 2004*; *Rivero, Lemence & Kato, 2017*; *Rivero & Kato, 2018*; *Do & Le, 2019*), and genomics (*Le et al., 2019*; *Le, 2019*). Among these many applications, SVM is reported to greatly outperform most of the other learning algorithms for handwritten character and digit recognition problems (*Byun & Lee, 2003*).

Many script character recognition systems have been reported with SVM as classifiers. In their experiments, *Phangtriastu, Harefa & Tanoto (2017)* have shown that SVM can achieve an accuracy of 94.43%, and is better when compared to Artificial Neural Network. *Tautu & Leon (2012)* have studied how SVM can be used to classify handwritten Latin letters, and obtained over 90% precision when tests were implemented on small or capital Latin letters. *Sok & Taing (2014)* have proposed SVM for printed Khmer Characters and obtained 98.62% recognition accuracy. Using SVM models, *Shanthi & Duraiswamy (2009)* have reported recognition for Tamil characters and obtained 82.04% accuracy. With 96.79% recognition rate, identification of Arabic handwritten characters have been presented by *Althobaiti & Lu (2018)* with the use of SVM classifiers and Normalized Central Moments (NCM) and Local Binary Patterns (LBP) for feature extraction. *Aggarwal & Singh (2015)* have reported gradient and curvature approach in feature extraction and have made use of SVM models in recognizing Gurmukhi characters where they obtained an accuracy of 98.56%. Using Zernike invariants and SVM classifiers, *Kaushal, Khan & Varma (2014)* have proposed a technique in identifying Urdu characters and obtained a recognition accuracy of 96.29%. *Dong, Krzyak & Suen (2005)* have used directional histograms and SVM models to recognize Chinese characters. Their work yields a 99% recognition rate. An accuracy of 87.32% have been reported by *Kilic et al. (2008)* using SVM model for Ottoman character recognition. Recognition of Malayalam characters using wavelet transforms and SVM classifiers have been introduced by *John, Pramod & Balakrishnan (2012)*, where their empirical results obtained a 90.25% accuracy. *Gaur & Yadav (2015)* have performed K-means clustering and used SVM classifiers to recognize Hindi characters and obtained a result of 95.86% accuracy. Utilizing SVM and combining Zoning and Gabor filter in feature extraction yields a result of 92.99% recognition rate in classifying Bangla characters (*Pervin, Afroge & Huq, 2017*).

Baybayin OCR is still in its infancy in literature. There are only few mathematical studies on Baybayin script. *Recario et al. (2011)* have presented an automated reader for Baybayin scripts where the system outputs the equivalent syllables of the Baybayin character. They used the Line Angle Categorization and Freeman Chain Coding for classification, and obtained results of 51.96% and 66.47% recognition accuracy, respectively. *Nogra, Romana & Maravillas (2019)* and *Nogra, Romana & Balakrishnan (2020)* have proposed an LSTM neural network and CNN models, with 92.90% and 94.00% recognition rates, respectively, that translates Baybayin handwritten characters to their corresponding syllabic unit in Latin

alphabet. While their works were done at character level, they have greatly contributed in introducing Baybayin characters to computer vision. *Recio & Mendoza (2019)* have implemented an edge detection approach on recognizing old images containing Baybayin texts. The method has been generally shown to be effective in detecting the texts' edges and reducing the noise level of an old image.

In line with the restoration of the script, we propose an OCR system that distinguishes Baybayin from the Latin script at character level in either handwritten or computer-generated form. The 17 main Baybayin characters and 52 (26 each for upper and lower cases) Latin characters are shown in Fig. 1. In this paper, we introduce a technique in classifying the characters of both scripts with the aid of SVM.

The remainder of this paper is organized as follows: in 'A Review of Support Vector Machine', a brief review of how support vector machine works is presented. 'Collecting Baybayin Dataset' discusses how Baybayin, its accents, and Latin characters were gathered. The proposed OCR algorithm for Baybayin and Latin script and character recognition are presented in 'Proposed OCR System'. In 'Experimental Setup, Results and Discussions', the results and discussion of our proposed OCR algorithm are shown. Lastly, conclusions and future works are presented in 'Conclusions and Future Works'.

A REVIEW OF SUPPORT VECTOR MACHINE

Our proposed OCR system consists of four classification problems wherein SVM is used. SVM considers a training set of points $\vec{x}_i \in \mathbb{R}^n$,i=1 ,…,$N$, where n is the number of features in a particular training sample and N is the number of training points. In a two-class or binary classification problem, each of these points are labeled by an indicator variable $y_i \in \{-1, 1\}$, depending on the class in which the data point belongs. The points are separated by classes or categories by a hyperplane called a *linear classifier*. The linear classifier can be written with a set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} + b = 0, \tag{1}$$

where $\vec{w}$ and b are the weight vector and bias term, respectively. With vector $\vec{w}$ normal to the hyperplane (Eq. (1)), the distance of the hyperplane from the origin is $|b|/\|\vec{w}\|$.

In a linearly separable case (i.e., it is possible to draw a line that can separate the two classes), we can select two parallel hyperplanes (dashed lines in Fig. 3) that separate the two classes with maximum distance from each other, in a way that there are no data points in between. The linear classifier (Eq. (1)) lies halfway between these hyperplanes, and the region bounded by one of these hyperplanes and the linear classifier is called a *margin*. SVM finds the values for the weight and bias that gives the maximum-margin hyperplane that separates the two classes, so that any point on or above the hyperplane

$$\vec{w} \cdot \vec{x} + b = 1$$

is labeled 1 while points on or below

$$\vec{w} \cdot \vec{x} + b = -1$$

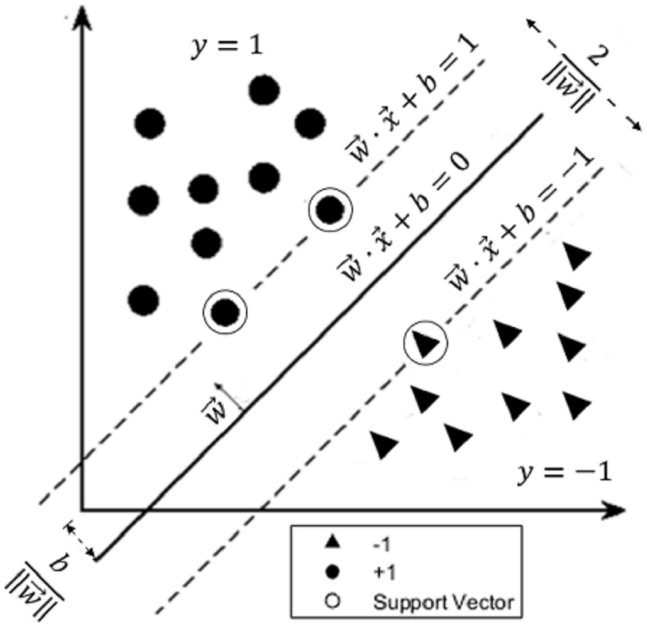

Figure 3 The maximum-margin hyperplane. The figure shows an example of a two-class data separated by a linear classifier. The parallel dashed lines represent the hyperplanes $\vec{w}\cdot\vec{x}+b=1$ and $\vec{w}\cdot\vec{x}+b=-1$ where the distance between them is $\frac{2}{\|\vec{w}\|}$. The circular-shaped data belong to the positive group (labeled with $y=1$) while the triangular-shaped data belong to the negative group (labeled with $y=-1$). The points on the dashed lines are called support vectors. SVM finds the optimal hyperplane (solid line) whose distance from the origin is $\frac{b}{\|\vec{w}\|}$ and lies halfway between the two (dashed) hyperplanes.
Full-size DOI: 10.7717/peerjcs.360/fig-3

is labeled -1. Hence, for any training data point $\vec{x}_i$ and its corresponding label y_i, we have

$$y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1, i=1,\ldots,N,$$

and the optimization problem is then given by

$$\underset{\vec{w},b}{\text{minimize}}\ \ \frac{1}{2}\|\vec{w}\|^2$$

$$\text{subject to}\ \ y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1\ \ \text{for all}\ i=1,\ldots,N. \tag{1}$$

The primal Lagrangian is constructed as

$$L(\vec{w},b,\vec{\alpha})=\frac{1}{2}\|\vec{w}\|^2-\sum_{i=1}^{N}\alpha_i\left[y_i(\vec{w}\cdot\vec{x}_i+b)-1\right], \tag{2}$$

where $\alpha_1,\alpha_2,\ldots,\alpha_N\geq 0$ are Lagrange multipliers. Differentiating (Eq. (3)) with respect to $\vec{w}$ and b and then equating to zero, yields

$$\vec{w}=\sum_{i=1}^{N}y_i\alpha_i\vec{x}_i \tag{3}$$

and

$$\sum_{i=1}^{N}\alpha_i y_i=0, \tag{4}$$

respectively. Substituting (Eqs. (4)) and ((5)) to ((3)), the primal Lagrangian (Eq. (3)) becomes

$$L(\vec{w},b,\vec{\alpha})=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j(\vec{x}_i\cdot\vec{x}_j). \tag{5}$$

The Karush–Kuhn–Tucker (KKT) conditions assert that the optimal solutions $\vec{\alpha}^*$, $\vec{w}^*$, and b^* must satisfy

$$\alpha_i^*\left[y_i\left(\vec{w}^*\cdot\vec{x}_i+b^*\right)-1\right]=0, \tag{6}$$

which implies that for all nonzero $\vec{\alpha}_i^*$,

$$y_i\left(\vec{w}^*\cdot\vec{x}_i+b^*\right)=1,$$

where the $\vec{x}_i$'s are precisely the data points or vectors on the margin. These vectors are called *support vectors* as they are the ones which determine or "support" the margins. From (Eqs. (1)) and ((4)), the maximum-margin hyperplane is now given by

$$\sum_{i\in S}y_i\alpha_i^*(\vec{x}_i\cdot\vec{x})+b^*=0, \tag{7}$$

where S is the set of indices of the support vectors. A new data point $\vec{x}\in\mathbb{R}^n$ can now be assigned a class through the decision function

$$f(\vec{x})=\operatorname{sign}\left(\sum_{i\in S}y_i\alpha_i^*(\vec{x}_i\cdot\vec{x})+b^*\right), \tag{8}$$

using only the support vectors.

Although there are datasets that can be linearly classified, this is not usually the case, as can be seen in Fig. 4A. As a solution, *Boser, Guyon & Vapnik (1992)* proposed that the data in the input space be mapped into a higher-dimensional space, called a *feature space*, where a linear separator could be found (see Fig. 4B). The computation of the optimal hyperplane in this scenario is not done explicitly on the feature space but rather by the use of a *kernel trick*. This is briefly discussed below.

Suppose that the data points in the input space are mapped into a feature space by using some nonlinear function ϕ. Then (Eq. (6)) becomes

$$L=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j\left(\phi(\vec{x}_i)\cdot\phi(\vec{x}_j)\right). \tag{9}$$

Similarly, (Eqs. (8)) and ((9)) become

$$\sum_{i\in S}y_i\alpha_i^*\left(\phi(\vec{x}_i)\cdot\phi(\vec{x}_j)\right)+b^*=0, \tag{10}$$

and

$$f(\vec{x})=\operatorname{sign}\left(\sum_{i\in S}y_i\alpha_i^*\left(\phi(\vec{x}_i)\cdot\phi(\vec{x}_j)\right)+b^*\right), \tag{11}$$

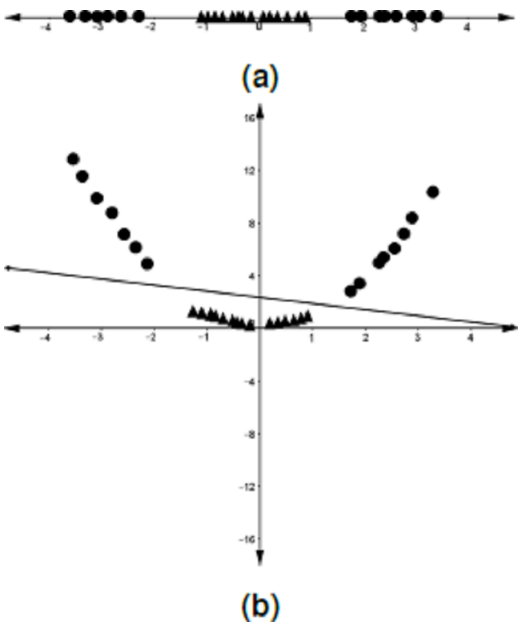

Figure 4 Mapping of the data points from the input space onto a feature space. (A) One-dimensional dataset that cannot be separated by a linear classifier. (B) The data points in A are mapped into a higher-dimensional space (feature space) where a linear classifier is found.
Full-size DOI: 10.7717/peerjcs.360/fig-4

respectively. Notice that in the above equations, the inner product happens only on the images of the inputs $\vec{x}_i$ and $\vec{x}_j$. Mercer's theorem states that if,

$$\kappa\left(\vec{x}_i,\vec{x}_j\right)=\phi(\vec{x}_i)\cdot\phi(\vec{x}_j) \tag{12}$$

is positive-definite, then the function κ in (Eq. (13)) is called a *kernel function*. Equation (13) also tells us that as long as the function κ is positive-definite, we are assured of some inner products of the images in the feature space. This follows that the coordinates of the images or mapped data points need not be explicitly computed, and the kernel trick allows us to write (Eqs. (10)), ((11)), and ((12)) as

$$L=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j\kappa\left(\vec{x}_i,\vec{x}_j\right), \tag{13}$$

$$\sum_{i\in S}y_i\alpha_i^*\kappa\left(\vec{x}_i,\vec{x}_j\right)+b^*=0, \quad \text{and} \tag{14}$$

$$f(\vec{x})=\text{sign}\left(\sum_{i\in S}y_i\alpha_i^*\kappa\left(\vec{x}_i,\vec{x}_j\right)+b^*\right), \tag{15}$$

respectively. Some commonly-used kernels are as follows:

- Linear kernel: $\kappa(\vec{u},\vec{v})=\vec{u}\cdot\vec{v}$.

- Polynomial kernel: $\kappa(\vec{u},\vec{v}) = (\vec{u}\cdot\vec{v}+r)^n$, where r and n are the polynomial coefficient and degree, respectively.
- Radial Basis Function (RBF) kernel: $\kappa(\vec{u},\vec{v}) = e^{-\frac{\|\vec{u}-\vec{v}\|^2}{2\sigma^2}}$, where σ is an arbitrary constant.

Among these three kernel functions, the RBF kernel is the most versatile and preferred, especially when not much of the data is known (*Sok & Taing, 2014*). The numerical results in *Tautu & Leon (2012)* showed that RBF fared well compared to other kernels in classifying handwritten characters.

COLLECTING BAYBAYIN DATASET

Data preparation is the main part of the system. In this section, we discuss how all the necessary dataset for Baybayin, its accents, and Latin characters were collected and compiled. In implementing essential functions to generate the dataset, MATLAB (vR2018a) Image Processing Toolbox was used.

For uniformity of dataset, each image was converted into binary data using a modified k-means clustering. K-means clustering provides a simple and flexible technique in grouping image intensities. It gives a good performance even in dark images and operates at a low computational complexity since it reduces the data dimension (*Pourmohammad, Soosahabi & Maida, 2013*). It chooses k centers so as to minimize the total squared distance between each point and its closest center. The number of centroids is equal to the number of clusters.

In this study, $k = 2$ was used in the modified k-means function for image binarization. The input image was clustered into two intensities, where lower intensities were set to 1s (white pixels), while higher intensities were set to 0s (black pixels). The MATLAB function `regionprops` was implemented to provide measurements for each connected component in the input binary image, and the following output were utilized: area, bounding box, and centroid. The image associated with the bounding box was then cropped using the `imcrop` function to limit the data into its significant features. The cropped image was then rescaled to 56×56 pixels using `imresize` function. To denoise the resized image, the command `bwareaopen` was used. This command removes all regions that have less pixels than the set pixel value. Finally, the image's feature vector was generated and then compiled for SVM training. Algorithm 1 and Fig. 5 illustrate the data preparation process succinctly.

For our dataset collection, the images of each Baybayin and Latin characters were searched online. A total number of 9,000+ images for Baybayin characters were taken from the dataset provided by *Nogra (2019)* in Kaggle. However, some characters in this dataset have less number of images than the other characters. Thus, to complete the dataset, additional images from various websites and books were collected through the use of a snipping tool. Computer-generated and (noisy) handwritten images were also added, as long as the character and its features are visible. The number of images per character was set to 1,000, and after binarization, Algorithm 1 was implemented to each of the binarized images.

Meanwhile, the images for Latin characters were taken from *Vial (2017)* and *Nano (2016)* in Kaggle. The generated Latin dataset was set to 700 images per character, with a

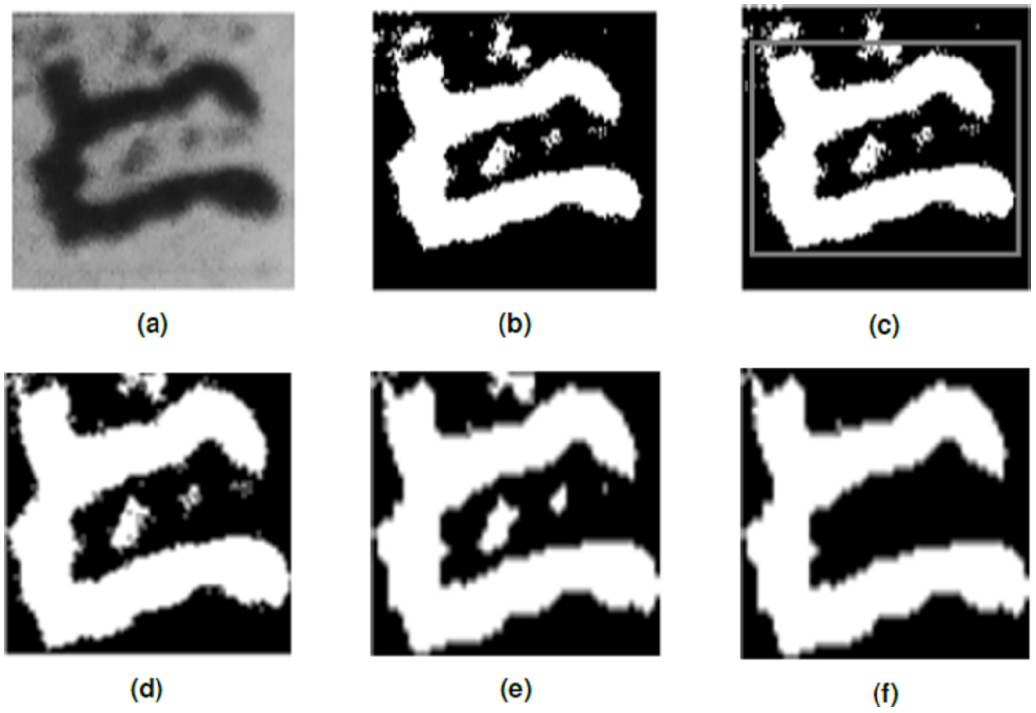

Figure 5 Feature extraction process of a raw character image. (A) Raw character image, 106 × 108; Binary image of A, 106 × 108. (C) Character bounding box, 106 × 108. (D) Cropped image, 86 × 99. (E) Resized image, 56 × 56. (F) Denoised image, 56 × 56.

Full-size DOI: 10.7717/peerjcs.360/fig-5

Algorithm 1 Feature Extraction Process

Require: Character binary image.

Ensure: Feature vector of size 3,136 with elements 0s and 1s.

1: Identify the character's significant features by computing its bounding box measurement.
2: Crop the image associated with the bounding box measurement to fit and obtain the character's essential area.
3: Rescale the cropped image into a size of 56 × 56 pixels.
4: Denoise unnecessary region/components that are not connected to the character.
5: Extract its feature vector by concatenating each row right next to each other to form a $1 \times 3,136$ array.

balanced number of upper and lower cases. The binarized images were also preprocessed using Algorithm 1.

Baybayin diacritic images were also gathered for classification training. Images for dots, bars, cross, and x were collected from *Nano (2016)* in Kaggle. The images for minus, plus, and times symbols, together with the letter "o", were modified to produce the diacritic symbols. The binarized images also underwent Algorithm 1 for preprocessing, but due to the characters' thinness, the `imdilate` function was applied after the `bwareaopen`

operation to dilate the binary input and provide a preferable data than the original image. A total of 500 images for each diacritic was produced.

The generated dataset for the Baybayin and Latin characters and Baybayin diacritics can be accessed in *Pino (2020a)*.

PROPOSED OCR SYSTEM

The proposed system assumes that the character input has the following properties:

- The character print has the lowest intensity (darkest) than any other part of the image.
- If an input is a Baybayin character with a diacritic or accent, the main body (character) has the largest number of pixels in the image, followed by the accent.
- The accent symbol should be written properly (i.e., the accent symbol should be above or below the main character, should not be touching the main character, and must be within the width of the main character).

From an input character image, the system starts with a preprocessing part where the original image is transformed into binary data. If the input has only one component (see Fig. 2A), the system proceeds as in Algorithm 1. On the other hand, if the input has an accent (Figs. 2B to 2D), we locate and separate the main component from the accent/diacritic component, then both components are preprocessed using Algorithm 1. With this, the rest of the structure of the system is constructed from location segmentation, component normalization, feature extraction, classification, and character recognition. Figure 6 shows the overall flow of the system.

Location segmentation and feature extraction

This is trivial for the one-component image. For the two-component image, after the binarization of the original image using modified k-means, the function `regionprops` was implemented to obtain the information on the image components. Then, the `imcrop` function was applied to extract the two highest area components (main body and accent). The two components were then isolated. Steps 3 –5 of Algorithm 1 were then implemented to the (isolated) components to obtain their respective $1 \times 3{,}136$ feature vectors, which will later be used for classification.

Character classification and recognition

For classification and recognition processes, four main SVM classifiers were considered. The main body feature vector X goes through two classification nodes as shown in Fig. 7. The first classifier discriminates the Baybayin from the Latin script. Afterwards, if the first classifier determines X to be in Baybayin alphabet, X proceeds to the Baybayin character classifier and identifies which character unit it represents. A similar method applies if X is recognized as a Latin character by the first classifier node. In the case where there is more than one component, the system assumes that X is a Baybayin accent (see Fig. 6), and its feature vector goes to the Baybayin diacritic SVM classifier.

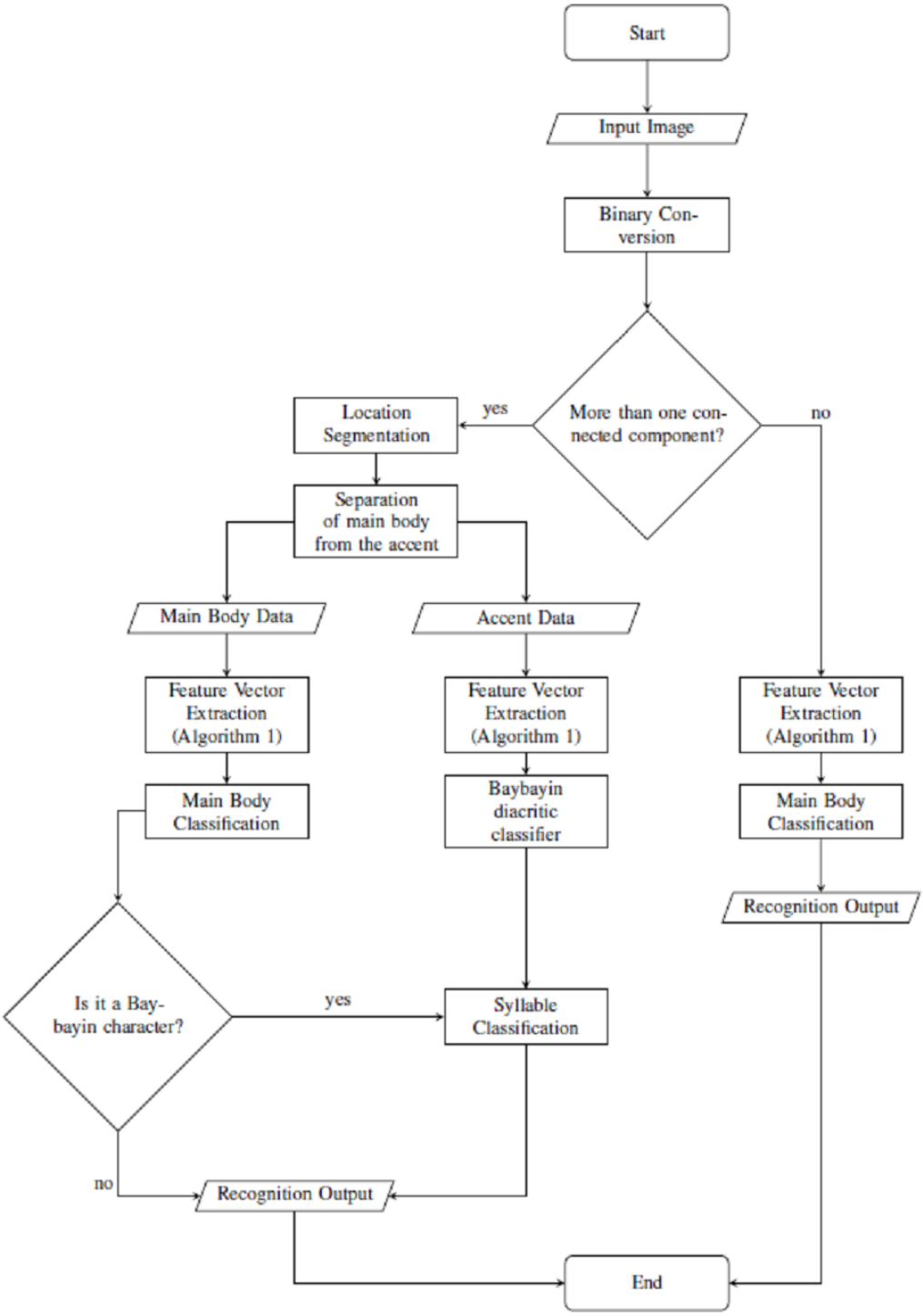

Figure 6 **Proposed OCR system.**

Full-size DOI: 10.7717/peerjcs.360/fig-6

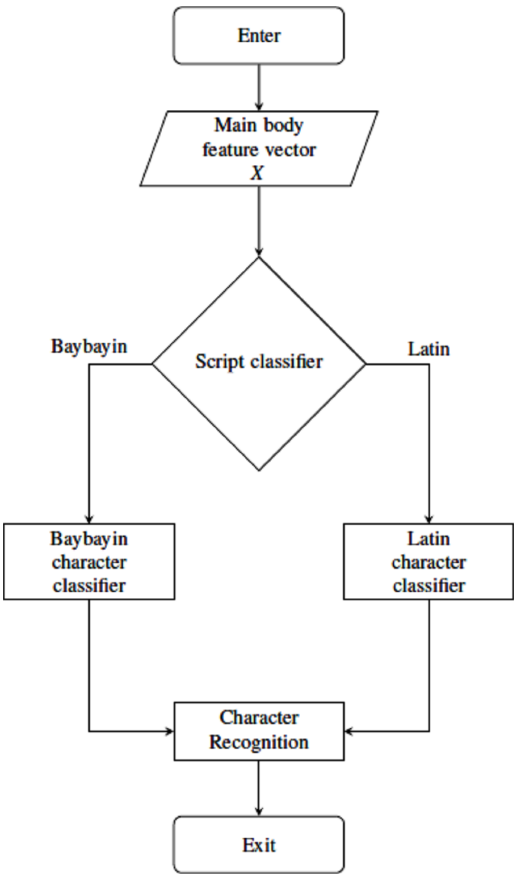

Figure 7 **Main body classification process using SVM classifiers.**
Full-size DOI: 10.7717/peerjcs.360/fig-7

One component case

From the result obtained in the main body classification, if the image is classified as a Baybayin character, the system prints out its Latin equivalent. On the other hand, if the image is classified as Latin, the Latin character itself is printed out.

Two components case

In the two components case, the system separates the main body of the character image from its diacritic (or accent), and identifies the main body as either Baybayin or Latin character.

If the main body is classified as a Baybayin character, the diacritic classifier result is combined with the Baybayin character for syllable classification. The system determines the correct placement of the accent through its ordinate values from the centroid's coordinates. That is, if the accent's centroid ordinate value is less than the main body's, then its placement is above the character (see Fig. 8). On the other hand, if the accent's centroid ordinate value is greater than the main body's, then its placement is below the character. The Baybayin diacritic classifier then discriminates between the dot-bar and

Figure 8 **Binary image with the centroid locations superimposed.**
Full-size DOI: 10.7717/peerjcs.360/fig-8

cross-x symbol. Finally, the system prints out the Latin equivalent of the Baybayin character with the recognized associated unit represented by the accent.

Meanwhile, if the main body is classified as a Latin character, then there is a chance that a noise was mistaken as another component or it could be that the character is either an 'i' or a 'j'. In the latter case, the system ignores the accent classifier and prints directly the identified Latin character.

OCR system main process

This section summarizes the main process of the proposed system. Given an individual Baybayin or Latin character input image, the system converts the input into a binary image. Afterwards, if the system detects only one region/component from the input, it will proceed directly to Algorithm 1 for feature extraction process which serves as input for the main body classification. The main body classification process contains SVM classifiers that will categorize the input vector as either Baybayin or Latin, and further recognize it as to what character it represents. The system then outputs the corresponding default Latin syllable for Baybayin characters and an equivalent Latin letter for Latin characters.

For the two component case, the system assumes that the other region represents the Baybayin diacritic symbol. After converting the raw input image into binary data, the system locates the two components' centroids and computes for their respective region of interest (bounding box). Each of these components then goes through a similar process as in Algorithm 1. The resulting feature vector for the main body proceeds to the main body classification while the accent's feature vector will be fed into the Baybayin diacritic SVM classifier. If the main body classification results to a Latin script, the system ignores the Baybayin diacritic classifier result and outputs directly its equivalent Latin letter. Otherwise, the Baybayin character is combined with the diacritic information, the placement of which is based on the centroid and the diacritic classifier result. The system then outputs the equivalent Latin syllable or unit it represents. Algorithm 2 summarizes the overall OCR process. The MATLAB code used in this work can be accessed in GitHub shared by *Pino (2020b)*.

Algorithm 2 Proposed OCR System.

Require: Isolated Baybayin or Latin character image.
Ensure: Equivalent Latin syllable/unit.

```
Convert the input image into binary data.
Count the number of components N from the binary image.
if N = 1 then
    Implement Algorithm1.
    Feed the feature vector X into character script classifier.
    if Baybayin then
        Feed X into Baybayin character classifier.
    else
        Feed X into Latin character classifier.
    end if
    Print the identified script and its default equivalent Latin unit.
else
    Locate the centroids of two components with highest pixel area.
    Identify the main body and the accent components.
    Implement Algorithm1 to both selected components.
    Feed the accent feature vector into Baybayin diacritic classifier.
    Feed the main body feature vector X into character script classifier
    if Latin then
        Discard Baybayin diacritic classifier result.
        Feed X into Latin character classifier.
    else
        Feed X into Baybayin character classifier.
        Combine the result with the Baybayin diacritic classifier outcome.
        if Accent is located above then
            E/I vowel concatenation.
        else
            if Bar or dot then
                O/U vowel concatenation.
            else
                Vowel cancellation.
            end if
        end if
    end if
    Print the identified script and its corresponding Latin unit.
end if
```

Table 1 Templates used in MATLAB SVM training.

Property Name/Set to	Setup Description
Kernel Function/'rbf'	The software makes use of Gaussian kernels in implementing the algorithm to generate a classification model.
Data Standardization/'true'	The software standardize the predictors before training the classifier for a potential decrease in classification error.
Box Constraint Parameter/'inf'	The software makes a strict classification which means there will be no points misclassified in training.
Kernel Scale Parameter/'auto'	The software applies an appropriate kernel norm to compute the Gram matrix that arises from kernel functions.

EXPERIMENTAL SETUP, RESULTS AND DISCUSSIONS

The SVM models presented here were obtained using the MATLAB (vR2018a) Statistics and Machine Learning Toolbox. The two main functions used were `fitcsvm` for binary classification and `fitcecoc` for multiclass classification. Both tools support predictor data mapping with the use of kernel functions, and can employ SVM solvers like Sequential Minimal Optimization (SMO) which is a fast algorithm for training SVMs (*Platt, 1998*). Moreover, `fitcecoc` also provides multiclass learning by producing an error-correcting output codes (ECOC) classifier that combines binary classifiers in order to solve a multiclass problem (*Escalera, Pujol & Radeva, 2010*).

Each SVM model was produced using the template properties shown in Table 1. From the dataset collected, the function `fitcsvm` was implemented to train an SVM classifier for discriminating Latin and Babayin characters. Training an SVM classifier to differentiate dot and bar from cross and x for the Baybayin diacritics classifier was similarly carried out. An SVM multiclass classification model was trained for each script (Baybayin and Latin characters) using the `fitcecoc` function. The experiments were carried out with 20% and 30% holdout option for testing, i.e., 80% and 70% of the gathered data were used to train the models.

In the multiclass setting, the model's performance on each character was measured. Figure 9 shows the performance results (accuracy, precision, recall, and F1 score) on each of the ten independent runs for each classifier (script, Latin character, and Baybayin character) in the testing phase, for 20% and 30% holdout samples.

Meanwhile, the generalization performance for the Baybayin diacritics showed 0% classification error in both holdout options. This result can be attributed to the symbols' unique structures compared to other characters.

The mean and standard deviation of the generalization performance results were also calculated for each holdout options (see Table 2). Empirical results show that the 20% holdout percentage yields better generalization performance than the 30% holdout percentage.

It is noteworthy that most of the errors came from the characters with similar structures. For example, small letter 'm' and capital letters 'I' and 'T' from Latin were sometimes recognized as Baybayin characters 'Na', 'Ka', and 'La', respectively (see Fig. 1). Also, in

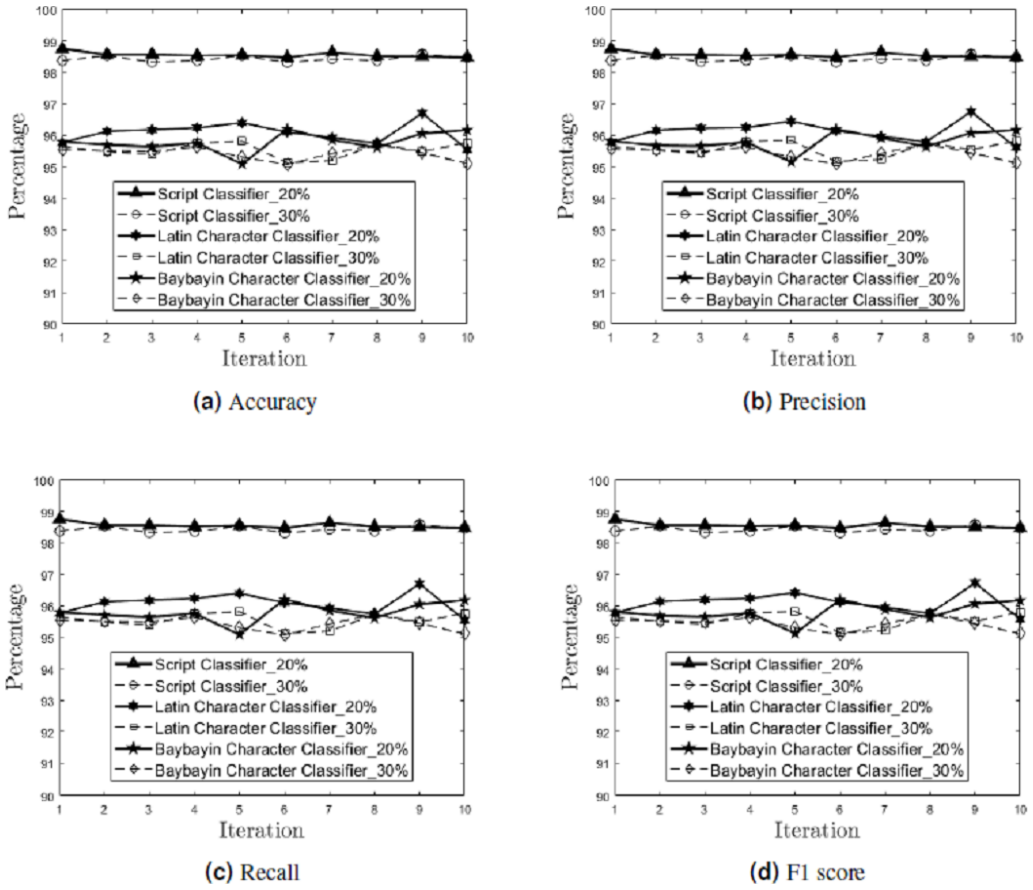

Figure 9 Generalization Performance Results. The generalization performance (accuracy (A), precision (B), recall (C), and F1 score (D)) of each classifier (script, Latin character, and Baybayin character) is measured per iteration, for 20% and 30% holdout samples.

Full-size DOI: 10.7717/peerjcs.360/fig-9

Table 2 Mean and standard deviation (SD) of the generalization performance results for 10 iterations in the testing phase with 20% and 30% holdout options. Better performance results between the two holdout options are styled in bold.

Classifiers	Holdout	Accuracy		Precision		Recall		F1 Score	
	Percentage	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Script	20%	**98.56**	0.08	**98.56**	0.08	**98.55**	0.08	**98.56**	0.08
	30%	98.44	0.09	98.44	0.09	98.43	0.09	98.43	0.09
Baybayin Characters	20%	**95.79**	0.32	**95.81**	0.31	**95.79**	0.32	**95.80**	0.32
	30%	95.43	0.21	95.43	0.20	95.43	0.21	95.43	0.21
Latin Characters	20%	**96.07**	0.34	**96.11**	0.34	**96.07**	0.34	**96.10**	0.34
	30%	95.54	0.24	95.58	0.24	95.54	0.24	95.56	0.24

Latin character identification, most errors came from letter 'I' being identified as either 'L' or 'J'.

Figure 10 reports the model's accuracy (diagonal cells), precision (rightmost column), and recall (bottom row) for each Baybayin character. It can be observed that

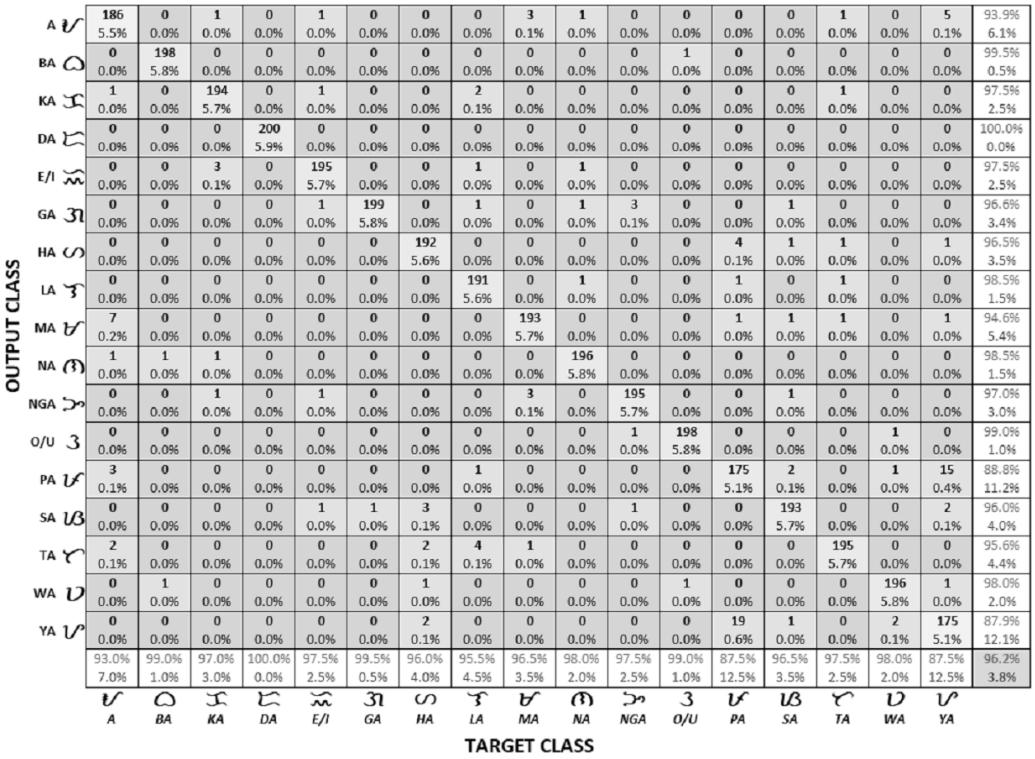

Figure 10 Generated confusion matrix from the best Baybayin character classifier. The diagonal entries correspond to the number of correctly classified characters, while the off-diagonal entries correspond to the number of incorrectly classfied characters. The last column and the last row correspond to the precision and recall of each character, respectively. The bottom right entry is the overall accuracy of the model.

Full-size DOI: 10.7717/peerjcs.360/fig-10

misclassifications occured among the characters 'Pa', 'Ma', 'Ya', and 'A', since they have very similar forms (see Fig. 1). To solve this problem, a reconsideration classifier was added to the main body classification process for these ambiguous characters. The best models were obtained with their respective accuracy: 'A' vs. 'Ma' –97.75%; 'Ka' vs. 'E'/'I' –99%; 'Ha' vs. 'Sa' –99.5%; 'La' vs. 'Ta' –99%; and 'Pa' vs. 'Ya' –94.50%. Figure 11 shows an example of Baybayin and Latin character images that are fed into the proposed system and produced a correct classification result (see Figs. 1 and 2 for verification).

To further test the performance of the proposed system on Baybayin characters, the images that satisfy the system's assumptions (see 'Proposed OCR System') were selected from the gathered dataset in *Nogra (2019)*. Among the selected images, 1,100 were randomly chosen for the test runs and were fed into the proposed OCR system for recognition. The conducted experiment obtained an overall average of 98.41%, 98.68%, 98.45%, and 98.57% for accuracy, recall, precision, and F1 score, respectively.

CONCLUSIONS AND FUTURE WORKS

This study has largely contributed to the script and character recognition community by providing a new set of data for Baybayin characters, its diacritics, and Latin characters

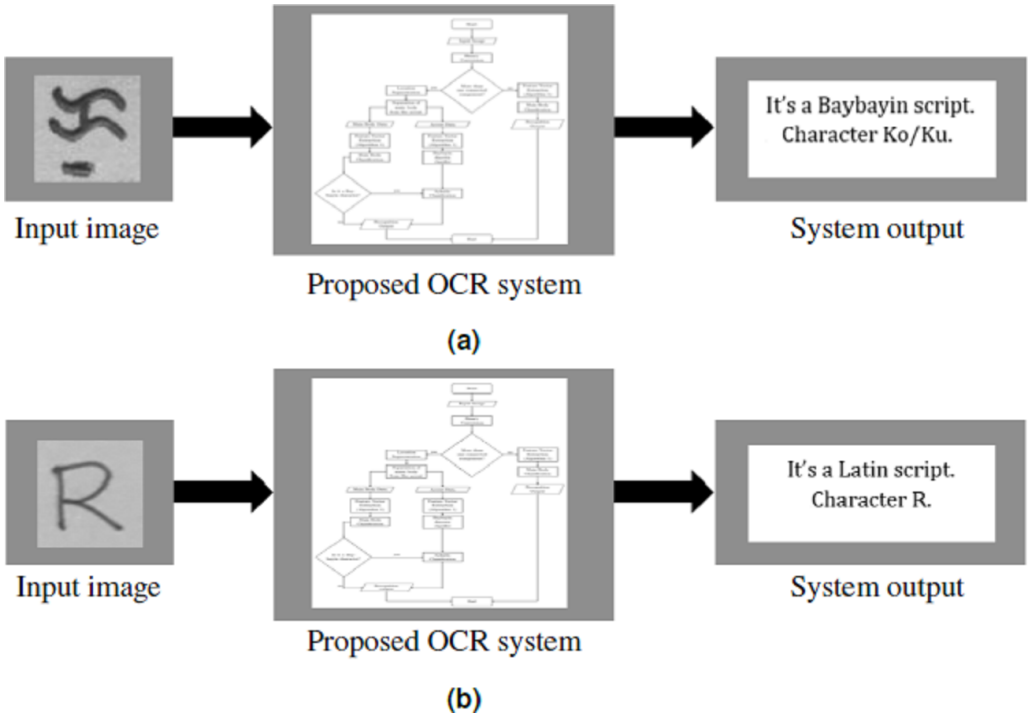

Figure 11 System framework for Baybayin and Latin characters. (A) Character Ko/Ku enters the system. (B) Letter R enters the system.

Full-size DOI: 10.7717/peerjcs.360/fig-11

that can be used in future related studies. This paper also introduced an OCR system that utilizes SVM in discriminating Baybayin script from Latin, and in classifying each of the writing system's characters. A limitation of this study is that the Baybayin characters written in a manner where the diacritics are attached to the main body were not considered. This setting is sensible since Baybayin characters are generally written with the diacritics detached from the main body. An advantage of this limitation though is that the number of classes in the SVM multiclass algorithm was significantly reduced.

Another strong point of this study is that the proposed system accommodates the highly similar structures among the Baybayin characters, yielding higher generalization performance. Experimental results also show that SVM can be an effective tool in Baybayin character recognition.

For future works, one can explore the use of multiclass SVM in classifying Baybayin characters where the characters with accents are treated as separate classes. One can also explore other machine learning algorithms to solve the classification problems arising from the proposed OCR. A comprehensive comparative study of different classification algorithms applied to Baybayin scripts is also an interesting research direction. Similar to what was shown in *Phangtriastu, Harefa & Tanoto (2017)*, other feature extraction algorithms can be combined with SVM to identify which will work well in classifying Baybayin scripts.

Improving the number of images in the dataset may also significantly decrease the classification error. The Baybayin characters considered in this study are based on the traditional scripts only. Modern Baybayin characters are being proposed recently to conform with the modern Filipino alphabet. Inclusion of these modern Baybayin characters in the proposed OCR system is another interesting extension of this study.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Office of the Chancellor of the University of the Philippines, through the Office of Vice Chancellor for Research and Development, through the Outright Research Grant (Project No. 202025 ORG). R. Pino acknowledges the Department of Science and Technology - Science Education Institute (DOST-SEI) through the Accelerated Science and Technology Human Resources Development Program (ASTHRDP) Scholarship, for funding support. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
The Office of the Chancellor of the University of the Philippines, through the Office of Vice Chancellor for Research and Development, through the Outright Research Grant: 202025 ORG.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Rodney Pino conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Renier Mendoza and Rachelle Sambayan conceived and designed the experiments, analyzed the data, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The data used in this article can be found at Kaggle. Specifically:

- Baybayin images: https://www.kaggle.com/rodneypino/baybayin-and-latin-binary-images-in-mat-format?select=Baybayin

- Latin images: https://www.kaggle.com/rodneypino/baybayin-and-latin-binary-images-in-mat-format?select=Latin

- Baybayin diacritics images: https://www.kaggle.com/rodneypino/baybayin-and-latin-binary-images-in-mat-format?select=Baybayin+Diacritics

The MATLAB code is available in GitHub:
https://github.com/rbp0803/An-OCR-System-for-Baybayin-Scripts-using-SVM.

REFERENCES

Aggarwal A, Singh K. 2015. Handwritten Gurmukhi character recognition. In: *2015 international conference on computer, communication and control (IC4)*. 1–5.

Althobaiti H, Lu C. 2018. Arabic handwritten characters recognition using support vector machine , normalized central moments , and local binary patterns. In: *Proceedings of the International Conferences of Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. Athens.

Bhunia AK, Konwer A, Bhunia AB. 2019. Script identification in natural scene image and video frames using an attention based Convolutional-LSTM network. *Pattern Recognition* **85**:172–184 DOI 10.1016/j.patcog.2018.07.034.

Bishop C. 2006. *Pattern recognition and machine learning (information science and statistics)*. Heidelberg: Springer-Verlag,Berlin.

Boser B, Guyon I, Vapnik V. 1992. A training algorithm for optimal margin classifiers. In: *Proceedings of the fifth annual workshop on computational learning theory*. New York: ACM, 144–152.

Byun H, Lee S-W. 2003. A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence* **17(3)**:459–486 DOI 10.1142/S0218001403002460.

Cabuay C. 2009. *An introduction to baybayin*. Raleigh: Lulu Press, Inc.

Chanda S, Pal U, Franke K, Kimura F. 2010. Script identification a han and roman script perspective. In: *2010 20th international conference on pattern recognition*. 2708–2711.

Chanda S, Pal U, Kimura F. 2007. Identification of Japanese and English script from a single document page. In: *7th IEEE international conference on computer and information technology (CIT 2007)*. Piscataway: IEEE, 656–661.

Chanda S, Terrades OR, Pal U. 2007. SVM based scheme for thai and english script identification. In: *Ninth international conference on document analysis and recognition (ICDAR 2007), vol. 1*. 551–555.

Chaudhuri A, Mandivaya K, Badelia P, Ghosh S. 2017. *Optical character recognition systems for different languages with soft computing*. 1st edition. Cham, Switzerland: Springer.

Cristianini N, Shawe-Taylor J. 2000. *An introduction to support vector machines*. Cambridge: Cambridge University Press.

Do DT, Le NQK. 2019. A sequence-based approach for identifying recombination spots in Saccharomyces cerevisiae by using hyper-parameter optimization in FastText and support vector machine. *Chemometrics and Intelligent Laboratory Systems* **194**:103855 DOI 10.1016/j.chemolab.2019.103855.

Dong J, Krzyak A, Suen CY. 2005. An improved handwritten chinese character recognition system using support vector machine. *Pattern Recognition Letters* **26(12)**:1849–1856.

Escalera S, Pujol O, Radeva P. 2010. On the decoding process in ternary error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32(1)**:120–134 DOI 10.1109/TPAMI.2008.266.

Ganapathiraju A, Hamaker JE, Picone J. 2004. Applications of support vector machines to speech recognition. *IEEE Transactions on Signal Processing* **52(8)**:2348–2355 DOI 10.1109/TSP.2004.831018.

Gaur A, Yadav S. 2015. Handwritten Hindi character recognition using k-means clustering and SVM. In: *2015 4th international symposium on emerging trends and technologies in libraries and information services*. 65–70.

Ghosh D, Dube T, Shivaprasad A. 2010. Script recognition—a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32(12)**:2142–2161 DOI 10.1109/TPAMI.2010.30.

Hangarge M, Santosh KC, Pardeshi R. 2013. Directional discrete cosine transform for handwritten script identification. In: *2013 12th international conference on document analysis and recognition*. 344–348.

Jaeger S, Ma H, Doermann D. 2005. Identifying script on word-level with informational confidence. In: *Eighth international conference on document analysis and recognition (ICDAR'05), vol. 1*. 416–420.

John J, Pramod KV, Balakrishnan K. 2012. Unconstrained handwritten malayalam character recognition using wavelet transform and support vector machine classifier. *Procedia Engineering* **30**:598–605 DOI 10.1016/j.proeng.2012.01.904.

Kaushal DS, Khan Y, Varma S. 2014. Handwritten urdu character recognition using zernike mi's feature extraction and support vector machine classifier. *International Journal of Research* **1(7)**:1084–1089.

Kilic N, Gorgel P, Ucan ON, Kala A. 2008. Multifont Ottoman character recognition using support vector machine. In: *2008 3rd international symposium on communications, control and signal processing*. 328–333.

Le NQK. 2019. iN6-methylat (5-step): identifying DNA N6-methyladenine sites in rice genome using continuous bag of nucleobases via Chou's 5-step rule. *Molecular Genetics and Genomics* **294**:1–10 DOI 10.1007/s00438-019-01570-y.

Le NQK, Yapp EKY, Ho QT, Nagasundaram N, Ou YY, Yeh HY. 2019. iEnhancer-5Step: Identifying enhancers using hidden information of DNA sequences via Chou's 5-step rule and word embedding. *Analytical Biochemistry* **571**:53–61 DOI 10.1016/j.ab.2019.02.017.

Lim MK, Manipon RH (eds.) 2019. *Bilangan 2: selected papers from the 2019 international conference on cultural statistics and creative economy*. NCCA,Intramuros, Manila, Philippines.

Nano X. 2016. Handwritten math symbols dataset. *Available at https://www.kaggle.com/xainano/handwrittenmathsymbols/data* (accessed on 9 April 2020).

Nayak J, Naik B, Behera HS. 2015. A comprehensive survey on support vector machine in data mining tasks: applications & challenges. *International Journal of Database Theory and Application* **8(1)**:169–186.

Nogra J. 2019. Baybayn (Baybayin) Handwritten Images. *Available at https://www.kaggle.com/jamesnogra/baybayn-baybayin-handwritten-images%20* (accessed on 14 May 2020).

Nogra JA, Romana CLS, Balakrishnan E. 2020. Baybáyin character recognition using convolutional neural network. *International Journal of Machine Learning and Computing* **10(2)**:169–186.

Nogra JA, Romana CLS, Maravillas E. 2019. LSTM neural networks for baybyin handwriting recognition. In: *2019 IEEE 4th international conference on computer and communication systems (ICCCS)*. Piscataway: IEEE, 62–66.

Obaidullah SM, Halder C, Santosh KC, Das N, Roy K. 2017. PHDIndic_11: page-level handwritten document image dataset of 11 official Indic scripts for script identification. *Multimedia Tools and Applications* **77**:1643–1678.

Pervin MT, Afroge S, Huq A. 2017. A feature fusion based optical character recognition of Bangla characters using support vector machine. In: *2017 3rd international conference on electrical information and communication technology (EICT)*. 1–6.

Phangtriastu MR, Harefa J, Tanoto DF. 2017. Comparison between neural network and support vector machine in optical character recognition. *Procedia Computer Science* **116**:351–357 DOI 10.1016/j.procs.2017.10.061.

Pino R. 2020a. Baybayin and Latin (Binary) Images in .mat Format. *Available at https://tinyurl.com/y4yuc5yj* (accessed on 8 June 2020).

Pino R. 2020b. An OCR system for baybayin scripts using SVM. *Available at https://tinyurl.com/y4sdzd9j* (accessed on 28 October 2020).

Platt J. 1998. Sequential minimal optimization: a fast algorithm for training support vector machines. *Available at https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/*.

Pourmohammad S, Soosahabi R, Maida AS. 2013. An efficient character recognition scheme based on k-means clustering. In: *2013 5th international conference on modeling, simulation and applied optimization (ICMSAO)*. 1–6.

Rajput GG, Ummapure SB. 2017. Script identification from handwritten documents using SIFT method. In: *2017 IEEE international conference on power, control, signals and instrumentation engineering (ICPCSI)*. Piscataway: IEEE, 520–526.

Rani R, Dhir R, Lehal GS. 2013. Script identification of pre-segmented multi-font characters and digits. In: *2013 12th international conference on document analysis and recognition*. 1150–1154.

Recario RN, Mariano V, Galvez DA, Lajara CM. 2011. An automated reader philippine baybayin scripting image processing methods. In: *ICCC international digital design invitation exhibition*. 75–76.

Recio K, Mendoza R. 2019. Three-step approach to edge detection of texts. *Philippine Journal of Science* **148(1)**:193–211.

Rivero R, Kato T. 2018. Parametric models for mutual kernel matrix completion. *IEICE Transactions on Information and Systems* **E101.D(12)**:2976–2983 DOI 10.1587/transinf.2018EDP7139.

Rivero R, Lemence R, Kato T. 2017. Mutual kernel matrix completion. *IEICE Transactions on Information and Systems* **E100.D(8)**:1844–1851 DOI 10.1587/transinf.2017EDP7059.

Sapankevych NI, Sankar R. 2009. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine* **4(2)**:24–38.

Schölkopf B, Smola A. 2012. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Cambridge: MIT Press.

Shanthi N, Duraiswamy K. 2009. A novel SVM-based handwritten Tamil character recognition system. *Pattern Analysis and Applications* **13**:173–180.

Shawe-Taylor J, Cristianini N. 2004. *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Sok P, Taing N. 2014. Support Vector Machine (SVM) based classifier for Khmer Printed Character-set Recognition. In: *Signal and information processing association annual summit and conference (APSIPA), 2014 Asia-Pacific*. 1–9.

Tautu E-D, Leon F. 2012. Optical character recognition using support vector machine. *Bulletin of the Polytechnic Institute of Jassy* **Tomul LVIII (LXII), Fasc. 2**:31–43.

Thomé A. 2012. *SVM classifiers—concepts and applications to character recognition*. London, United Kingdom: InTech.

Vial G. 2017. Cyrillic-oriented MNIST. *Available at https://www.kaggle.com/gregvial/comnist/data* (accessed on 3 April 2020).

Yang ZR. 2004. Biological applications of support vector machines. *Briefings in Bioinformatics* **5(4)**:328–338 DOI 10.1093/bib/5.4.328.

Zhou L, Lu Y, Tan CL. 2006. Bangla/English script identification based on analysis of connected component profiles. In: *Lecture notes in computer science: International workshop document analysis systems*. 243–254.

Zhu Y, Sun J, Minagawa A, Hotta Y, Naoi S. 2009. Separate Chinese Character and English Character by cascade classifier and feature selection. In: *2009 10th international conference on document analysis and recognition*. 1191–1195.