International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Optimized Recognition of Javanese Letters Images Using a Hybrid Principal Component Analysis and Support Vector Machine Model

Anggit Gusti Nugraheni [a]*, *Arya Samudra Mahardhika* [b], *Gunarso Wiwoho* [b] *and Bayu Setyo Wibowo* [c]*

[a] *Faculty of Science and Technology, Putra Bangsa University, Jl. Ronggowarsito No.18 Pejagoan, Kebumen, Central Java, Indonesia, 061060.*
[b] *Faculty of Economics and Business, Putra Bangsa University, Jl. Ronggowarsito No.18 Pejagoan, Kebumen, Central Java, Indonesia, 061060.*
[c] *Faculty of Engineering, Lambung Mangkurat University, Jl. Brig Gen. Hasan Basri, Banjarmasin, South Kalimantan, Indonesia*

A B S T R A C T

This study presents an optimized approach for recognizing Javanese letters images using a hybrid model that combines Principal Component Analysis (PCA) and Support Vector Machine (SVM) techniques. The research aims to enhance the accuracy and efficiency of Javanese letters recognition by leveraging the dimensionality reduction capabilities of PCA for feature extraction and the classification power of SVM. A comprehensive dataset of Javanese letters was utilized, and various preprocessing techniques were applied to improve image quality. The performance of the proposed model was evaluated using metrics such as accuracy and execution time. Results indicate that the hybrid PCA-SVM model significantly outperforms traditional recognition methods, achieving higher accuracy rates and faster processing times. This approach demonstrates the potential for effective Javanese script recognition, contributing to advancements in optical character recognition and preserving cultural heritage. The findings of this research can be summarized as follows: The Zoning-PCA-SVM scenario performed the best, achieving an average accuracy of 52.35% and the shortest execution time of 187.53 seconds. In comparison, the PCA feature extraction method was less effective for images where the background was dominant. The Zoning process was more successful in preparing data for PCA feature extraction. Additionally, lower accuracy was due to remaining noise after preprocessing, the introduction of new noise, and a limited amount of training data for each class.

Keywords: Javanese letter recognition, Hybrid PCA-SVM model, Feature extraction, Dimensionality reduction, Optical character recognition (OCR)

1. Introduction

Principal Component Analysis (PCA) has been widely utilized as an effective feature extraction technique in various studies, including lung cancer detection using CT scan images [1] and facial image recognition [2], [3], [4]. These studies demonstrate that PCA consistently achieves image recognition accuracy rates exceeding 70%. Further research by Syakhala et al. [5] reinforces PCA's advantages, showing it delivers faster recognition times between 1 to 1.5 seconds and superior accuracy of 86.6%, outperforming the Hidden Markov Model (HMM), which achieved recognition times of 2 to 7.5 seconds and an accuracy rate of 77.7%.

In addition to PCA, Support Vector Machine (SVM) is another widely-used method for data classification, originally developed for binary classification tasks. Its effectiveness is demonstrated in studies such as hepatitis diagnosis [6] and predicting student graduation timelines [7]. Despite its binary origins, SVM has also been successfully applied to multiclass problems, including the classification of five types of schizophrenia [8] and multiclass sentiment analysis [9]. Research has shown that SVM outperforms methods like Fuzzy-KNN, binary decision trees, and deep recurrent neural networks in various cases [10], [11].

Javanese script is an ancient writing system used in the Javanese language, comprising 20 basic letters and their variants, 10 numerals, 5 swara (vowel) characters, 8 murda (capital) characters and their variants, 5 partner characters and their variants, as well as several additional symbols [12]. Each character has a distinct shape, along with specific reading and writing rules. Previous research on Javanese script recognition has employed various methods, such as K-Nearest Neighbor (KNN) [13], Self-Organizing Maps [14], Back Propagation [15], and Learning Vector Quantization [16]. However, most studies have focused solely on recognizing the basic letters, while the inclusion of additional elements like letter pairs and supporting characters introduces greater complexity. To address this, more advanced techniques such as a combination of Principal Component Analysis (PCA) and Support Vector Machine (SVM) are needed to improve the accuracy and efficiency of Javanese letters recognition.

This research aims to evaluate the performance of combining Principal Component Analysis (PCA) and Support Vector Machine (SVM) for recognizing Javanese letters images. PCA will be applied for feature extraction, while SVM will serve as the classifier. The effectiveness of this hybrid approach will be assessed based on average accuracy and execution time (runtime), providing insights into how efficiently and accurately it can recognize and classify Javanese characters.

2. Methodology

2.1 Principal Component Analysis (PCA)

PCA is often used because of its ability to reduce data dimensions without losing important information. The feature values of the training image obtained through PCA can be visualized in the form of an eigen image. The feature extraction process using PCA involves several steps, namely [17]:

1. Converting Image Representation from Matrix to Vector

The image, initially in matrix form, is transformed into a vector by aligning its pixels sequentially. Each vectorized image is then consolidated into an image collection matrix.

2. Calculates the Overall Average of the Image

After constructing the image collection matrix, the next step is to compute the average value of each pixel across all images, referred to as the average image value (μ). The model for the image collection matrix is presented in Equation (2.1), while the calculation method for the average image value is detailed in Equation (2.2). This approach enables Principal Component Analysis (PCA) to achieve more efficient data compression, thereby streamlining the classification process.

$$\begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ X_d \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \cdots & \cdots & & \cdots \\ x_{d,1} & x_{d,2} & \ldots & x_{d,p} \end{bmatrix} \ldots\ldots (2.1)$$

$$\mu_p = \sum_{i=1}^{d} x_{i,p} \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

Where :

d : amount of training data

p : number of pixels

μ_p : average pixel to p

$x_{i,p}$: value of the pth pixel from the ith data

3. Computing the Zero Mean

The zero mean matrix (Φ) is generated by subtracting the average value from each data point in the image collection matrix, effectively centering the data around zero, as outlined in Equation (2.3). This step is crucial for enabling accurate eigenvector and eigenvalue calculations in the subsequent phase, ensuring that the data distribution is more representative and better prepared for further processing in PCA analysis.

$$\phi_{d,p} = x_{d,p} - \mu_p \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

Where:

$\Phi_{d,p}$: zero mean element of data to d, pixel to p

$x_{d,p}$: p pixel value of d data

μ_p : average pixel to p

4. Constructing a Covariance Matrix

The calculated zero mean matrix is then utilized to construct a covariance matrix, as outlined in Equation (2.4). This covariance matrix captures the linear relationships between variables in the data, serving as a crucial step in identifying the principal components during the PCA process.

$$K = \frac{1}{d-1} * \phi * \phi^T \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.4)$$

Where:

K : covariance matrix

d : amount of training data

Φ : zero mean matrix

5. Calculating the Eigenvectors and Eigenvalues of the Covariance Matrix

Once the covariance matrix is constructed, the next step is to calculate the eigenvectors and eigenvalues. Eigenvectors represent the primary directions of data variability, while eigenvalues indicate the extent of variability explained by each eigenvector. These two values are crucial in the PCA process for dimensionality reduction, as only the eigenvectors associated with the largest eigenvalues will be retained to form the principal components.

6. Ranking Eigenvalues and Their Corresponding Eigenvector Pairs in Descending Order

After calculating the eigenvalues and eigenvectors, the next step is to sort the eigenvalues in descending order, pairing each eigenvalue with its corresponding eigenvector. This ordering is crucial, as larger eigenvalues signify the most significant components in explaining data variability. Only the eigenvectors associated with the largest eigenvalues will be selected to form the principal components, facilitating effective dimensionality reduction while preserving essential information.

7. Projecting Data to Create an Eigen image

After selecting the principal eigenvector, the next step is to project the original data into the new space defined by this eigenvector. This process yields a simplified data representation while preserving essential information. The resulting projected image, known as an eigenimage, captures the primary features of the original data. This eigenimage is subsequently utilized in the classification or further analysis stages.

The data projection matrix will be utilized to extract feature values from the test data, while the eigenimage serves as the feature representation of the training image. To derive the projection matrix, calculations must be performed according to Equations (2.5) through (2.8). Meanwhile, the eigenimage can be computed using Equation (2.9). This process is crucial for ensuring that the features extracted from the test data align with the representation established in the training images.

$$S = (\phi^T * V)^2 \text{ .... (2.5)}$$

$$S = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,d} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,d} \\ \cdots & \cdots & & \cdots \\ s_{p,1} & s_{p,2} & \ldots & s_{p,d} \end{bmatrix}$$

$$R_d = \frac{1}{\sqrt{\sum_{i=1}^{p} s_{i,d}}} \text{ .... (2.6)}$$

$$R = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \cdots & \cdots & & \cdots \\ 0 & 0 & \ldots & R_d \end{bmatrix} \text{ .... (2.7)}$$

$$Projection = (S * R)^T \text{ .... (2.8)}$$

$$eigen\_image = X * Projection^T \text{ .... (2.9)}$$

Where:

Φ : zero mean matrix

V : eigenvector

p : number of pixels

d : amount of training data

X : matrix of the training image collection

8. Extracting Features from Test Images

The next step involves calculating the features of the test image using the previously obtained projection matrix. This process entails projecting the test image into the feature space defined by the principal eigenvector. By doing so, the features of the test image are extracted consistently, allowing for effective comparison with features from the training images. The results of this feature extraction will serve as the foundation for the subsequent classification stage, ensuring that relevant information is accurately identified and analyzed.

Test image features are obtained by multiplying the test image vector and the transpose of the projection matrix produced in step 7. This process allows the test image to be represented in the same feature space as the training image, thus facilitating further analysis and classification. In this way, relevant features from the test image can be extracted effectively, ensuring that important information is preserved in the pattern recognition process.

2.2 Support Vector Machine (MVM)

Support Vector Machine (SVM) is a powerful supervised learning method primarily used for binary classification tasks. However, it can be effectively extended to handle multiclass classification problems using the one-vs-one strategy. In this approach, SVM constructs individual classifiers for every possible pair of classes, comparing each class pairwise. This results in a more detailed and comprehensive model. By breaking the problem down into simpler two-class comparisons, SVM can tackle more complex classification challenges, ultimately improving accuracy in distinguishing between multiple classes and capturing intricate patterns in the data.

$$Y(x) = \sum_{i=1}^{p} (\alpha_i K(x, x_i) + \beta) \text{ .... (2.10)}$$

Where:

p : amount of data

K(x,xi) : kernel function

β : bias

A Support Vector Machine (SVM) fundamentally works by finding the optimal linear separator, known as a **hyperplane,** to distinguish between two classes. The key objective of SVM is to identify the hyperplane that maximizes the margin between the two classes, thereby enhancing classification accuracy. This margin represents the distance between the hyperplane and the closest data points from each class, known as support vectors. The wider the margin, the more robust the classification. The SVM function is mathematically expressed as in Equation (2.10), with the bias term (β) determined by the following equation.

$$\beta = -\frac{\sum_{i=1}^{p}\left(\alpha_i K(x^+, x_i)\right) + \sum_{i=1}^{p}\left(\alpha_i K(x^-, x_i)\right)}{2}$$

To obtain the αi value, the steps that need to be followed are as follows [7]:

1. Initialize the Values and Form the Hessian Matrix

The process begins with initializing the necessary parameter values, followed by constructing the Hessian matrix. The Hessian matrix captures the curvature of the objective function, which plays a critical role in the optimization process for determining the optimal value of αi.

Set the initial value of αi to 0, along with other key parameters, including lambda (λ), learning rate (γ), complexity (C), maximum epochs (maxEpoch), and minimum error threshold (ε). Afterward, compute the Hessian matrix using Equation (2.11). This step is essential in setting the stage for the upcoming optimization process.

$D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2)$............ (2.11)

2. Calculating the α value

The calculation of the αi value is performed through the following steps:

$$E_i = \sum_{j=1}^{n} \alpha_j D_{ij}$$

$\Delta\alpha_i = \min(\max(\gamma(1\text{-}E)\text{-}\alpha_i)C\text{-}\ \alpha_i)$

$$\alpha_i = \alpha_i + \Delta\alpha_i$$

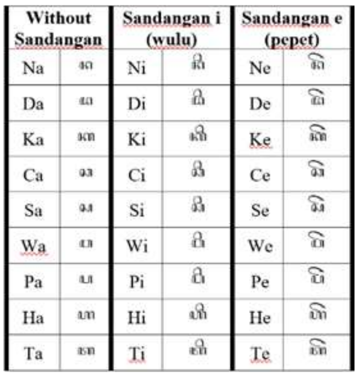
3. Repeat Step 2 until you reach the maximum number of epochs or |Δαi| ≤ ε

The process of calculating the value of αi is repeated continuously until it reaches the maximum number of epochs or until the difference in changes in αi (|Δαi|) is within the error tolerance limit (ε). This step is important to ensure convergence and accuracy in estimating the αi value

2.3 Scenario

In this research, Javanese letters recognition was performed using image data, comprising a total of 810 images divided into 27 classes (Table 2.1). Of these, 789 images were used for training and 81 for testing. The image preprocessing involved three treatment variations (scenarios). First, all images underwent noise removal, cropping, scaling, and grayscale conversion (PCA-SVM). Second, the images were also subjected to a thinning process (Thinning-PCA-SVM). Finally, in addition to noise removal, cropping, scaling, and grayscaling, a zoning process was applied to all images (Zoning-PCA-SVM). After preprocessing, the images were split into training and testing sets using 10-fold cross-validation. Feature extraction, training, and testing were then performed. Lastly, an evaluation was conducted to assess the performance of the PCA and SVM methods in recognizing Javanese characters.

Table 2.1 Image Data



Without Sandangan		Sandangan i (wulu)		Sandangan e (pepet)	
Na	ꦤ	Ni	ꦤꦶ	Ne	ꦤꦼ
Da	ꦢ	Di	ꦢꦶ	De	ꦢꦼ
Ka	ꦏ	Ki	ꦏꦶ	Ke	ꦏꦼ
Ca	ꦕ	Ci	ꦕꦶ	Ce	ꦕꦼ
Sa	ꦱ	Si	ꦱꦶ	Se	ꦱꦼ
Wa	ꦮ	Wi	ꦮꦶ	We	ꦮꦼ
Pa	ꦥ	Pi	ꦥꦶ	Pe	ꦥꦼ
Ha	ꦲ	Hi	ꦲꦶ	He	ꦲꦼ
Ta	ꦠ	Ti	ꦠꦶ	Te	ꦠꦼ

The feature extraction process yields image eigenvalues and a projection matrix through the application of the PCA method. The eigenimages serve as features for the training images, while the projection matrix is used to extract features from the test images. For both training and testing, the

SVM method is employed with a linear kernel and a one-vs-one multiclass strategy. This approach enables the model to effectively classify Javanese letters images based on the extracted features.

The one-vs-one strategy performs classification by comparing each pair of classes. As a result, multiple SVM models are generated in a single training process (Figure 2.1). The total number of models produced using this strategy can be determined by the combination formula, as simplified in Equation (2.12), where n represents the number of classes. In this research, the number of models is calculated as follows:

$$numbers\_of\_models = \frac{n(n-1)}{2} \text{ .... (2.12)}$$

$$numbers\_of\_models = \frac{27(27-1)}{2}$$

$$numbers\_of\_models = 351\ models$$

Test results from each treatment will be recorded, compared, and thoroughly analyzed. In this study, the comparison parameters include the average accuracy and execution time (runtime) for each scenario. The goal of this analysis is to determine which treatment delivers the best performance in recognizing Javanese characters.

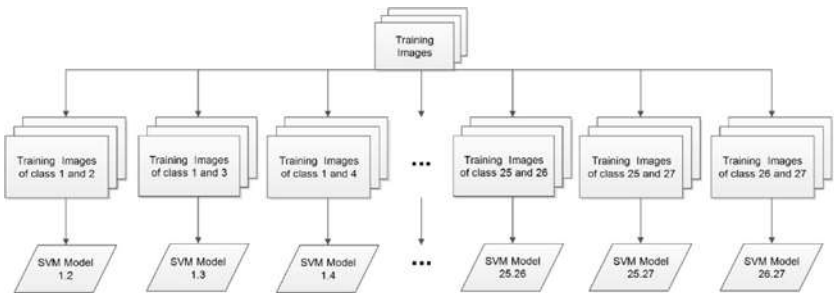

Training Images
Training Images of class 1 and 2
Training Images of class 1 and 3
Training Images of class 1 and 4
Training Images of class 25 and 26
Training Images of class 25 and 27
Training Images of class 26 and 27
SVM Model 1.2
SVM Model 1.3
SVM Model 1.4
SVM Model 25.26
SVM Model 25.27
SVM Model 26.27

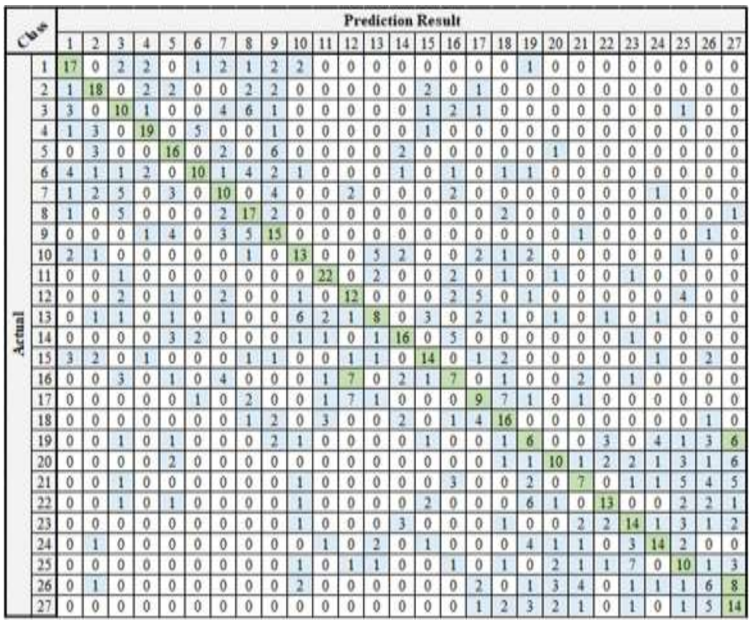
Figure 2.1 One vs one strategy SVM model

3. Results and Discussion

Figures 3.1 to 3.3 illustrate the results of test image recognition using the 10-fold cross-validation method for each scenario, presented in the form of confusion matrices. These figures reveal that several images in each class were still misclassified. For instance, in class 2 (Figure 3.1), 18 images were correctly recognized, while the remaining images were classified into other classes. In Figure 3.2, 17 test images were correctly identified as class 2, but the other 13 were misclassified into classes 4, 5, 8, and 9. In contrast, Figure 3.3 shows improved results, with 19 test images from class 2 correctly classified, while the rest were misclassified into classes 4, 5, 7, 8, 9, and 20.

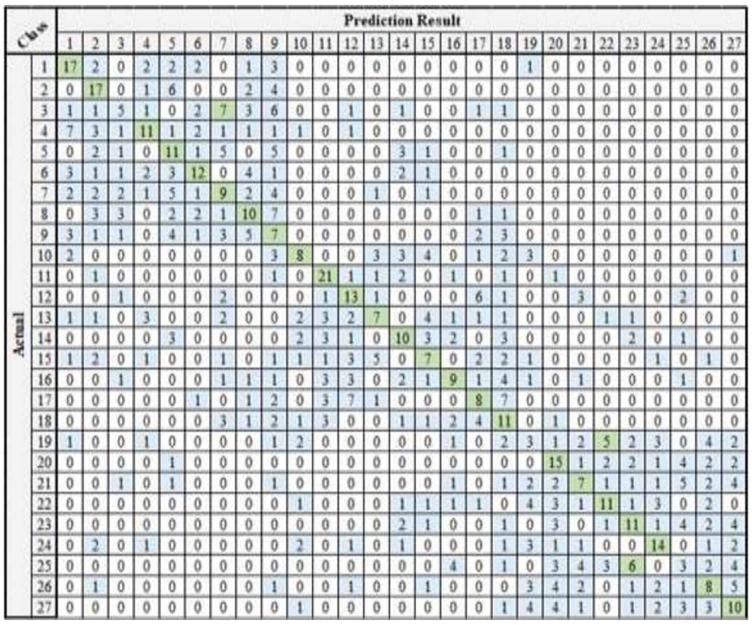
Recognition errors may arise due to similarities in the shapes of Javanese script characters across different classes. Many characters share significant visual similarities, making accurate classification challenging. Additionally, the smaller size of the character relative to the background contributes to these errors (see Figure 3.4). In the Thinning-PCA-SVM scenario, the background appears more prominent as the characters undergo the thinning process. Furthermore, errors in the initial writing of the characters at the beginning of the image can also lead to recognition inaccuracies (Figure 3.4).

Recognition errors may arise due to similarities in the shapes of Javanese letters characters across different classes. Many characters share significant visual similarities, making accurate classification challenging. Additionally, the smaller size of the character relative to the background contributes to these errors (see Figure 3.4). In the Thinning-PCA-SVM scenario, the background appears more prominent as the characters undergo the thinning process. Furthermore, errors in the initial writing of the characters at the beginning of the image can also lead to recognition inaccuracies (Figure 3.4).



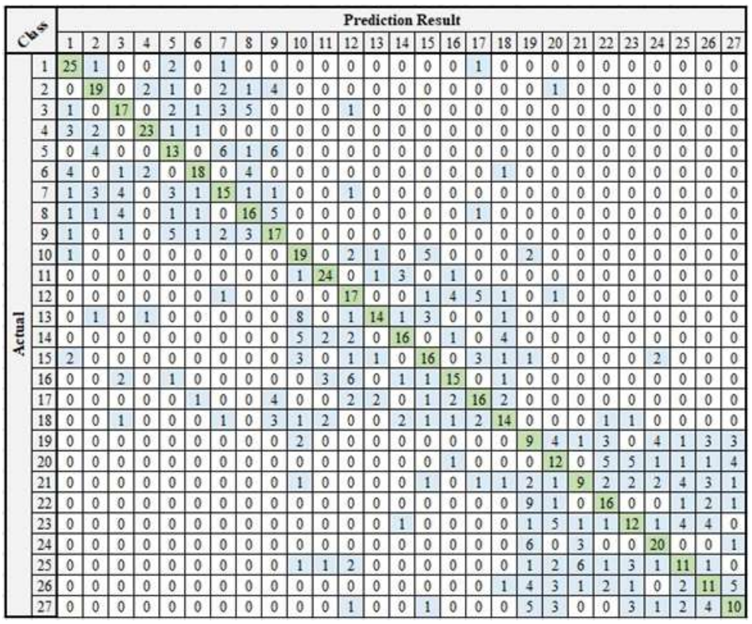
	Class	Prediction Result																										
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Actual	1	17	0	2	2	0	1	2	1	2	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	2	1	18	0	2	2	0	0	2	2	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0
	3	3	0	10	1	0	0	4	6	1	0	0	0	0	0	1	2	1	0	0	0	0	0	0	0	1	0	0
	4	1	3	0	19	0	5	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	3	0	0	16	0	2	0	6	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0
	6	4	1	1	2	0	10	1	4	2	1	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
	7	1	2	5	0	3	0	10	0	4	0	0	2	0	0	0	2	0	0	0	0	0	0	0	1	0	0	0
	8	1	0	5	0	0	0	2	17	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1
	9	0	0	0	1	4	0	3	5	15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
	10	2	1	0	0	0	0	0	1	0	13	0	0	5	2	0	0	2	1	2	0	0	0	0	0	1	0	0
	11	0	0	1	0	0	0	0	0	0	0	22	0	2	0	0	2	0	1	0	1	0	0	1	0	0	0	0
	12	0	0	2	0	1	0	2	0	0	1	0	12	0	0	0	2	5	0	1	0	0	0	0	0	4	0	0
	13	0	1	1	0	1	0	1	0	0	6	2	1	8	0	3	0	2	1	0	1	0	1	0	1	0	0	0
	14	0	0	0	0	3	2	0	0	0	1	1	0	1	16	0	5	0	0	0	0	0	0	1	0	0	0	0
	15	3	2	0	1	0	0	0	1	1	0	0	1	1	0	14	0	1	2	0	0	0	0	0	1	0	2	0
	16	0	0	3	0	1	0	4	0	0	0	1	7	0	2	1	7	0	1	0	0	2	0	1	0	0	0	0
	17	0	0	0	0	0	1	0	2	0	0	1	7	1	0	0	0	9	7	1	0	1	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	1	2	0	3	0	0	2	0	1	4	16	0	0	0	0	0	0	0	1	0
	19	0	0	1	0	1	0	0	0	2	1	0	0	0	0	1	0	0	1	6	0	0	3	0	4	1	3	6
	20	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	10	1	2	2	1	3	1	6
	21	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	2	0	7	0	1	1	5	4	5
	22	0	0	1	0	1	0	0	0	0	1	0	0	0	0	2	0	0	0	6	1	0	13	0	0	2	2	1
	23	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0	1	0	0	2	2	14	1	3	1	2
	24	0	1	0	0	0	0	0	0	0	0	1	0	2	0	1	0	0	0	4	1	1	0	3	14	2	0	0
	25	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	2	1	1	7	0	10	1	3
	26	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2	0	1	3	4	0	1	1	1	6	8
	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	2	1	0	1	0	1	5	14

Figure 0.1 *Confusion Matrix* PCA-SVM



	Class	Prediction Result																										
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Actual	1	17	2	0	2	2	2	0	1	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	2	0	17	0	1	6	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	1	1	5	1	0	2	7	3	6	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
	4	7	3	1	11	1	2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	2	1	0	11	1	5	0	5	0	0	0	0	3	1	0	0	1	0	0	0	0	0	0	0	0	0
	6	3	1	1	2	3	12	0	4	1	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0
	7	2	2	2	1	5	1	9	2	4	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	3	3	0	2	2	1	10	7	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
	9	3	1	1	0	4	1	3	5	7	0	0	0	0	0	0	0	2	3	0	0	0	0	0	0	0	0	0
	10	2	0	0	0	0	0	0	0	3	8	0	0	3	3	4	0	1	2	3	0	0	0	0	0	0	0	1
	11	0	1	0	0	0	0	0	0	1	0	21	1	1	2	0	1	0	1	0	1	0	0	0	0	0	0	0
	12	0	0	1	0	0	0	2	0	0	0	1	13	1	0	0	0	6	1	0	0	3	0	0	0	2	0	0
	13	1	1	0	3	0	0	2	0	0	2	3	2	7	0	4	1	1	1	0	0	0	1	1	0	0	0	0
	14	0	0	0	0	3	0	0	0	0	2	3	1	0	10	3	2	0	3	0	0	0	0	2	0	1	0	0
	15	1	2	0	1	0	0	1	0	1	1	1	3	5	0	7	0	2	2	1	0	0	0	0	1	0	1	0
	16	0	0	1	0	0	0	1	1	1	0	3	3	0	2	1	9	1	4	1	0	1	0	0	0	1	0	0
	17	0	0	0	0	0	1	0	1	2	0	3	7	1	0	0	0	8	7	0	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	0	3	1	2	1	3	0	0	1	1	2	4	11	0	1	0	0	0	0	0	0	0
	19	1	0	0	1	0	0	0	0	1	2	0	0	0	0	0	1	0	2	3	1	2	5	2	3	0	4	2
	20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	1	2	2	1	4	2	2
	21	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	2	2	7	1	1	1	5	2	4
	22	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	4	3	1	11	1	3	0	2	0
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	1	0	3	0	1	11	1	4	2	4
	24	0	2	0	1	0	0	0	0	0	2	0	1	0	1	0	0	0	1	3	1	1	0	0	14	0	1	2
	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	3	4	3	6	0	3	2	4
	26	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	3	4	2	0	1	2	1	8	5
	27	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	4	4	1	0	1	2	3	3	10

Figure 0.2 *Confusion Matrix Thining*-PCA-SVM



Class		Prediction Result																										
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Actual	1	25	1	0	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	2	0	19	0	2	1	0	2	1	4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	3	1	0	17	0	2	1	3	5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	3	2	0	23	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	4	0	0	13	0	6	1	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	4	0	1	2	0	18	0	4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	7	1	3	4	0	3	1	15	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	1	1	4	0	1	1	0	16	5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	9	1	0	1	0	5	1	2	3	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	1	0	0	0	0	0	0	0	0	19	0	2	1	0	5	0	0	0	2	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	1	24	0	1	3	0	1	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	1	0	0	0	0	17	0	0	1	4	5	1	0	1	0	0	0	0	0	0	0
	13	0	1	0	1	0	0	0	0	0	8	0	1	14	1	3	0	0	1	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	5	2	2	0	16	0	1	0	4	0	0	0	0	0	0	0	0	0
	15	2	0	0	0	0	0	0	0	0	3	0	1	1	0	16	0	3	1	1	0	0	0	0	2	0	0	0
	16	0	0	2	0	1	0	0	0	0	0	3	6	0	1	1	15	0	1	0	0	0	0	0	0	0	0	0
	17	0	0	0	0	0	1	0	0	4	0	0	2	2	0	1	2	16	2	0	0	0	0	0	0	0	0	0
	18	0	0	1	0	0	0	1	0	3	1	2	0	0	2	1	1	2	14	0	0	0	1	1	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	9	4	1	3	0	4	1	3	3
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	12	0	5	5	1	1	1	4
	21	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	2	1	9	2	2	2	4	3	1
	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	0	16	0	0	1	2	1
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	5	1	1	12	1	4	4	0
	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	3	0	0	20	0	0	1
	25	0	0	0	0	0	0	0	0	0	1	1	2	0	0	0	0	0	0	1	2	6	1	3	1	11	1	0
	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	3	1	2	1	0	2	11	5
	27	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	5	3	0	0	3	1	2	4	10

Figure 0.3 *Confusion Matrix Zoning*-PCA-SVM

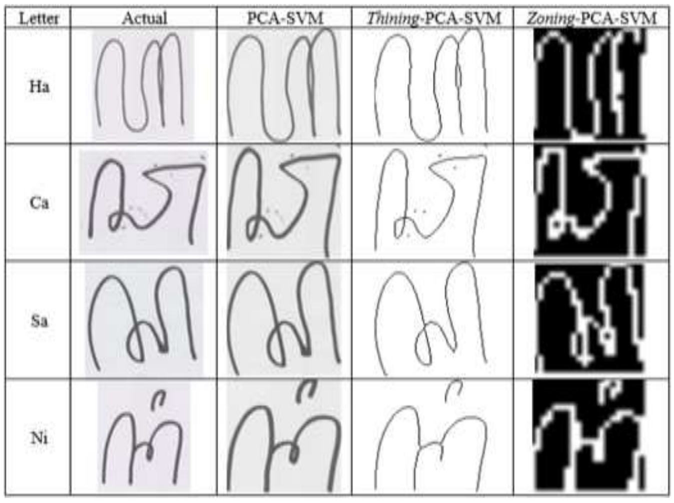


Figure 0.4 Example of Pre-processed Image Results

Table 3.1 Accuracy and Runtime

FOLD-	SCENARIO		
	PCA-SVM	***Thining*-PCA-SVM**	***Zoning*-PCA-SVM**
1	50.6173	34.5679	45.6790
2	49.3827	44.4444	65.4321
3	46.9136	33.3333	64.1975
4	59.2593	49.3827	69.1358
5	34.5679	28.3951	39.5062
6	30.8642	24.6914	29.6296
7	39.5062	34.5679	45.6790
8	34.5679	24.6914	54.3210
9	41.9753	34.5679	60.4938

FOLD-	**SCENARIO**		
	PCA-SVM	***Thining*-PCA-SVM**	***Zoning*-PCA-SVM**
10	35.8025	30.8642	49.3827
MEAN ACCURACY (%)	42.3457	33.9506	52.3457
RUNTIME (second)	253.561	254.342	191.407

Based on the confusion matrix analysis, the average accuracy for each scenario is shown in Table 3.1. The Zoning-PCA-SVM scenario achieved the highest average accuracy at 52.35%, with the shortest execution time of 191.41 seconds. In comparison, the PCA-SVM scenario had an average accuracy of 42.35% and a runtime of 253.56 seconds. The Thinning-PCA-SVM scenario recorded the lowest accuracy at 33.95%, with a runtime of 254.34 seconds. The advantage of the Zoning-PCA-SVM scenario in terms of shorter runtime is attributed to the reduced image dimensions resulting from the Zoning process, which compresses the images to only 400 pixels.

The use of the Thinning method during the preprocessing stage results in suboptimal images for feature extraction using PCA. This is evident from the accuracy values across all scenarios in the 10-fold cross-validation, as shown in Table 3.1. The Thinning-PCA-SVM scenario consistently produced the lowest accuracy among the three scenarios. Although in some folds the Zoning-PCA-SVM scenario had lower accuracy than the PCA-SVM scenario, overall, the Zoning-PCA-SVM scenario achieved the highest accuracy. Accuracy is calculated by dividing the number of correctly predicted images in each scenario and fold by the total of 81 test images.

4. Conclusions

The findings of this research can be concluded as follows:

1. The Zoning-PCA-SVM scenario demonstrated the best performance, achieving the highest average accuracy of 52.35% and the shortest execution time of 187.53 seconds.
2. The PCA feature extraction method proved less effective in handling images where the background dominates the object.
3. The Zoning process was more successful in preparing data suitable for feature extraction using PCA.
4. Remaining noise after preprocessing, the introduction of new noise, and the limited amount of training data per class contributed to the lower accuracy observed.

5. Suggestions

In future research, several steps can be taken to improve the results:

1. Implement more effective noise reduction methods.
2. Explore alternative classification methods beyond SVM that are better suited for multi-class recognition.
3. Increase the amount of training data to enhance model performance.
4. Apply feature extraction techniques other than PCA.
5. Introduce additional preprocessing treatments beyond Thinning and Zoning.

References

[1] Ada and R. Kaur, 2013. Feature Extraction and Principal Component Analysis for Lung Cancer Detection in CT scan Images. IJARCSSE, vol. 3, no. 3, pp. 187-190.

[2] Pratiwi, 2014. 2DPCA Feature Extraction Method for Facial Image Recognition with Matlab (In Indonesian) Jurnal Teknologi, vol. 7, no. 1, pp. 1-5.

[3] Fu, Q. 2015. Research and Implementation of PCA Face Recognition Algorithm Based on Matlab (in Indonesian) EDP Sciences.

[4] Budi, S.A., Suma'inna and H. Maulana, 2016. Facial Image Recognition as an Identifier Using the Principal Component Analysis (PCA) Method. (in Indonesian) JURNAL TEKNIK INFORMATIKA, vol. 9, pp. 166-175.

[5] Syakhala,A.R., D., Puspitaningrum and E. P. Purwandari, 2015. Comparison of the Principal Component Analysis (PCA) Method with the Hidden Markov Model (HMM) Method in Recognizing Someone's Identity through Faces (in Indonesian). Jurnal Rekursif, vol. 3, no. 2.

[6] Munawarah, R., O. Soesanto and M. R. Faisal, 2017. Application of the Support Vector Machine Method in Hepatitis Diagnosis. (in Indonesian). Kumpulan jurnaL Ilmu Komputer (KLIK). vol. 4, no. 1.

[7] Pratama, A., R. C. Wihandika and D. E. Ratnawati, 2018. Implementation of the Support Vector Machine (SVM) Algorithm for Predicting Student Graduation Timeliness. (in Indonesian). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 4, pp. 1704-1708.

[8] Perdana, A., M. T. Furqon and Indriati, 2018. Application of the Support Vector Machine (SVM) Algorithm in the Classification of Schizophrenia Mental Illness (Case Study: RSJ. Radjiman Wediodiningrat, Lawang).(in Indonesian). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 9, pp. 3162-3167.

[9] Liu, Y., J.-W. Bi and Z.-P. Fan, 2017. A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm. Elsevier, p. 38–52.

[10] Hasanah, U., L. R. M, A. Pratama and I. Cholissodin, 2016. Comparison of SVM, Fuzzy-KNN, and BDT-SVM Methods for Classifying Heart Rate Results from Electrocardiography (in Indonesian). Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), pp. 201-207.

[11] Al-Smadi, M., O. Qawasmeh, M. Al-Ayyoub, Y. Jararweh and B. Gupta, 2017. Deep Recurrent Neural Network vs. Support Vector Machine for Aspect-based Sentiment Analysis of Arabic Hotels' Reviews. Elsevier, p. 386–393.

[12] Piyono, J., 2005. Kawruh Pepak Basa Jawa Anyar, Surakarta: Pustaka Mandiri.

[13] Mukhoyyar, Z., 2015. Recognition of Javanese Script Words Using the K-Nearest Neighbor Algorithm (in Indonesian).

[14] Hidayat, A. and R. N. Shofa, 2016. Self Organizing Maps (SOM) a Method for Recognizing Javanese Letters (in Indonesian). Jurnal Siliwangi, vol. 2, no. 1.

[15] Nurmila, N., A. Sugiharto and E. A. Sarwoko, 2010. Back Propagation Neural Network Algorithm for Recognizing Javanese Letter Character Patterns (in Indonesian). Jurnal Masyarakat Informatika, vol. 1, no. 1.

[16] Mafrur, R., M. Andestoni, M. S. Ahdi, N. S. Fajri and A. Muhantini, 2011. Introduction to Javanese Letters Using the Learning Vector Quantization (LVQ) Method (in Indonesian). Research Gate, Yogyakarta.

[17] Purnomo, M.H., and A. Muntasa, 2010. Concepts of Digital Image Processing and Feature Extraction, First ed (in Indonesian). Graha Ilmu, Yogyakarta:.