

Deep convolutional neural network for Lampung character recognition

Panji Bintoro¹, Zulkifli Zulkifli², Fitriana Fitriana³, Sukarni Sukarni³

¹Department of Software Engineering, Faculty of Technology and Informatics, Aisyah University, Pringsewu, Indonesia

²Department of Informatics Engineering, Faculty of Technology and Informatics, Aisyah University, Pringsewu, Indonesia

³Department of Medical Records and Health Information, Faculty of Health, Aisyah University, Pringsewu, Indonesia

Article Info

Article history:

Received May 15, 2023

Revised Dec 20, 2023

Accepted Feb 24, 2024

Keywords:

Deep convolutional neural network

Lampung characters

Pattern recognition

Principal component analysis

Support vector machine

ABSTRACT

Recognition of document based, and handwritten characters has recently emerged as highly relevant field of study in the field of digital image processing. The ability to read and write Lampung script is a crucial competency as it helps preserve the language, which is a part of Indonesian culture. This research utilizes data obtained from classified documents and handwritten samples, categorized into eight types. To recognize Lampung characters, deep convolutional neural network (DCNN) architecture is proposed. The novelty of this architecture lies in optimizing document-based and handwritten character recognition to achieve the best performance in terms of accuracy and execution time. The proposed architecture will be compared to principal component analysis (PCA) combined with support vector machine (SVM) to evaluate its results. Experimental results using the DCNN architecture show an average accuracy of 99.3% and an execution time of 283 seconds for all data, while PCA and SVM exhibit an average accuracy of 92.9%. Furthermore, the recognition results for all data from documents and handwritten samples yield satisfactory accuracy of 98.6%. These results make the DCNN architecture suitable for use in recognizing Lampung characters and are expected to make it easier for Lampung people to recognize Lampung character.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Panji Bintoro

Department of Software Engineering, Faculty of Technology and Informatics, Aisyah University

Pringsewu, Lampung, Indonesia

Email: panjibintoro09@aisyahuniversity.ac.id

1. INTRODUCTION

The recognition of both document-based and handwritten characters has recently emerged as a highly relevant area of study within the realm of digital image processing. The vast range of recognizable characters highlights the ongoing need for research in character recognition [1]. Furthermore, recognizing characters in the Lampung script is particularly challenging due to the complexity of its geometric character shapes, many of which belong to multiple classes. The recognition of document images is further complicated by complex text layouts that affect characters with varying font sizes and styles. It is crucial to acknowledge that the Lampung script is a widely used language alphabet among the people of Lampung.

In contrast, deep neural networks (DNNs) have gained attention in the fields of machine learning and pattern recognition in recent years. DNNs consist of multiple layers and are capable of modeling complex functions that shallow networks cannot [2]. The development of advanced computing technologies and access to large-scale training data have made it feasible to train these deep networks, resulting in their widespread use across various domains. For instance, deep convolutional neural networks (DCNNs) have

demonstrated remarkable results in numerous image recognition fields, surpassing previous benchmark performances by significant margins [3], [4].

In the domains of computer vision and pattern recognition, deep learning techniques are widely considered to be effective tools for feature extraction and classification [5]. Consequently, this study proposes the use of a DCNN architecture to classify and extract features from images representing Lampung characters [6]-[8]. Furthermore, an efficient supervised statistical classifier, namely support vector machine (SVM), is utilized to categorize input patterns and separate the closest data points through the use of a hyperplane [9], [10]. This study primarily focuses on optimizing the recognition of both document-based and handwritten Lampung characters, with the aim of attaining the best possible performance results in terms of accuracy and execution time. Lampung characters originating from documents are shown in Figure 1, while Lampung characters originating from handwriting are shown in Figure 2.

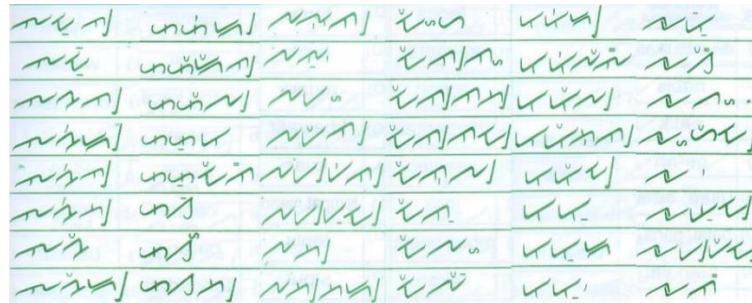


Figure 1. Lampung characters based on documents

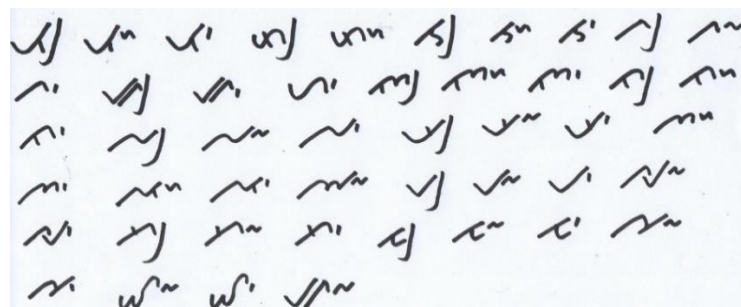


Figure 2. Lampung characters based on handwriting

Several recognition systems have been proposed for the recognition of document-based and handwritten characters, and a deep learning-based approach has been utilized in many cases, achieving highly promising results in terms of accuracy, Narasimhaiah and Rangarajan [11] conducted an extensive investigation into recognizing Kannada multi-characters. In this study, the dataset underwent classification utilizing the histogram of oriented gradients (HOG) feature descriptor, which was subsequently followed by dimensionality reduction techniques. Simple character combinations were found to perform well, while the other two classifications performed poorly. The DCNN model was found to be more suitable for all types of combined and compound data. The development of an artificial intelligence-based offline handwriting recognition system for Gujarati was studied in [12]. This study greatly contributed to data collection, collecting 10,000 images from 250 individuals of different ages and professions. The approach employed was based on a supervised classifier utilizing convolutional neural network (CNN) and multi-layer perceptron (MLP) algorithms, achieving success rates of 97.21% and 64.48% for CNN and MLP, respectively. Furthermore, handwriting recognition in Devanagari was performed using an image partitioning technique and HOG feature extraction [13]. The neural network was trained using a vector consisting of all partitions in the HOG, resulting in a maximum classification accuracy of 99.27% and an average accuracy of 97.06% in recognizing differences in Devanagari handwriting characters. Handwriting recognition was also developed [14] to recognize and convert images written in Bangla characters, which can be edited electronically. The data used was the BanglaLekha-Isolated dataset, and the proposed method was CNN. The model achieved an accuracy of 91.81% for the alphabet, consisting of 50-character classes. After expanding the number of images to 200,000 through data augmentation, the accuracy of the test set improved to 95.25%. Niharmin *et*

al. [15] utilized an optimized CNN method to recognize the handwritten Tifinagh characters. The CNN model suggested utilizing a multi-layer feed-forward neural network to extract features and characteristics directly from input image data. The aim was to enhance optical character recognition (OCR) to attain optimal performance in terms of accuracy and processing speed. Experimental outcomes demonstrated a commendable system accuracy of 99.27% along with an efficient classification execution time, surpassing previous studies. Arnia *et al.* [16] conducted to recognize Jawi characters based on invariant moment-based features. The research introduced an algorithm named tree root (TR), which was employed on two-character templates, specifically binary and thin, each containing 125 Jawi characters. In the binary template, TR successfully identified 90.4% of the characters, whereas in the thin template, the recognition rate was 87.2%. The proposed TR method achieved a higher recognition rate compared to SVM and Euclidean distance classifiers, which used the same test characters.

The subsequent sections of this paper are structured as: section 2 delineates the study's methodology and flow. Section 3 details the experiments conducted and their outcomes. Lastly, section 4 provides the conclusions drawn from the research.

2. METHOD

This research aims to compare some algorithm techniques to support the recognition of Lampung characters. The research begins by collecting data based on the data type, followed by augmentation processes and preprocessing using various methods. Feature extraction is performed before classification using the SVM algorithm. As a comparison, we combine the CNN and SVM algorithms, and finally, recognition steps are taken to measure the effectiveness of the optimal algorithm. Figure 3 shows the research framework for Lampung character recognition, the detailed steps are presented in the subsection.

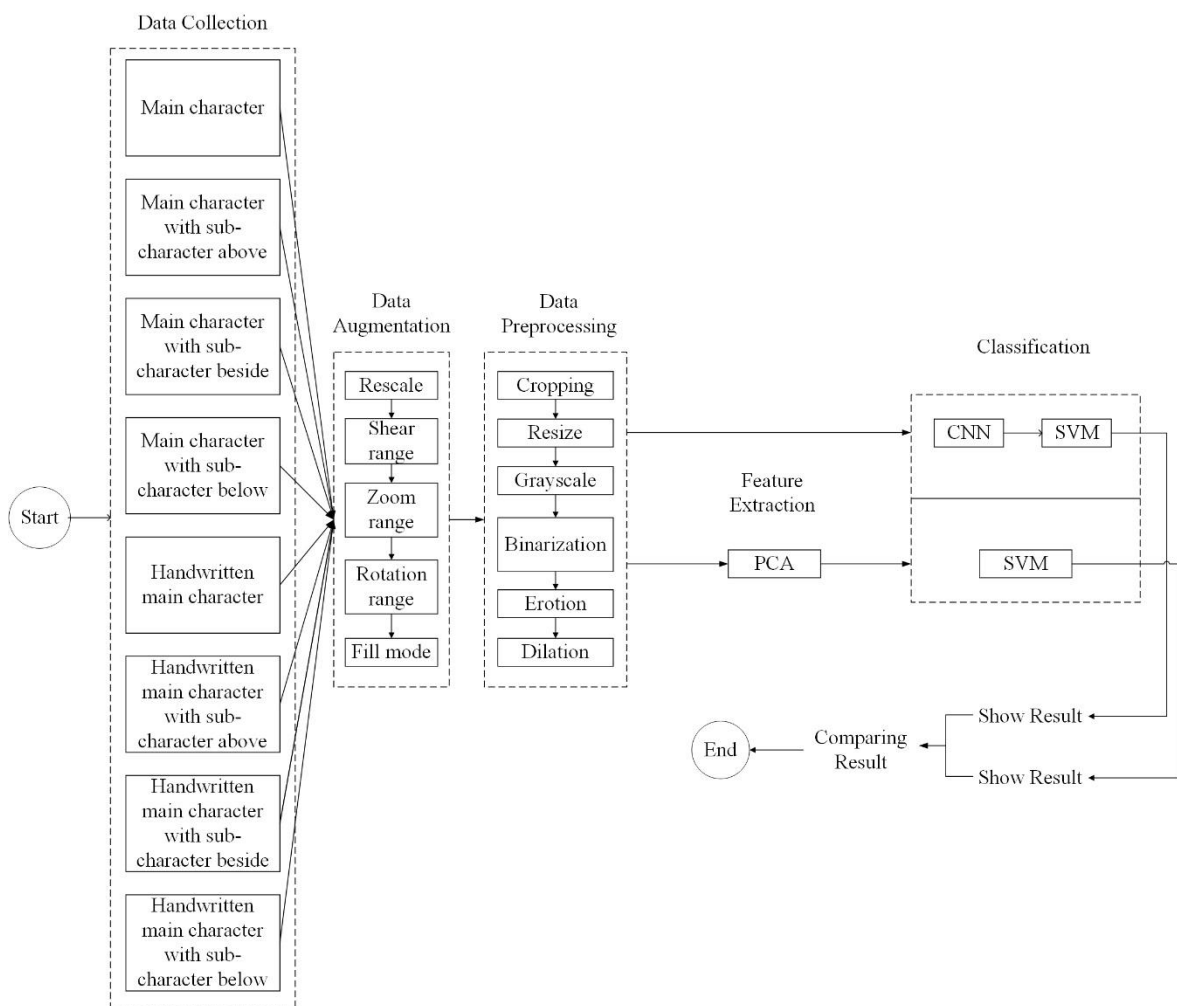


Figure 3. Research framework for recognition of Lampung characters

2.1. Data collection

In this present study, data on Lampung character images were collected from printed documents and handwriting and were divided into 4 categories. The process of collecting Lampung character data from documents and handwriting are respectively shown in Figure 4. The total data obtained for the main character is 520 images, which were divided into 20 classes, hence each has 26 main character images. For the main character with sub-characters above, a total of 2496 data images were acquired, which was classified into 96 classes, thereby amounting to 26 images. It was also observed that 1222 main characters with the sub-character on the side were obtained. The data is grouped into 47, hence each has 26 main character images with sub-character on the side. Regarding the main character with the sub-character below, 1482 images were collected. The data is sorted into 57 classes, each having 26 main character images with sub-characters below.

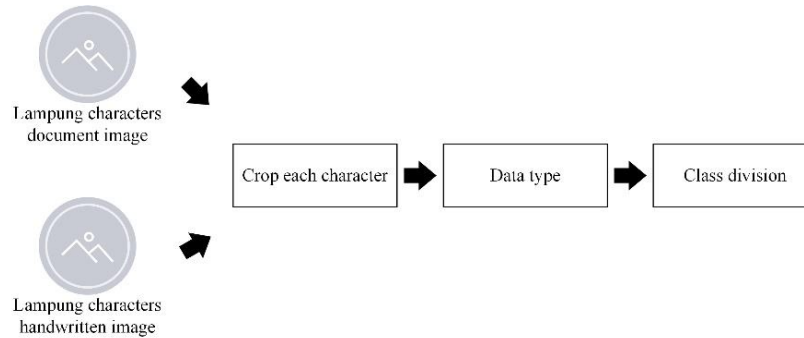


Figure 4. The data collection process

In the main handwritten character, 240 images were obtained, which were divided into 20 classes. Therefore, each has 26 images of the main handwritten character. A total of 1152 images were collected in the main handwritten character with the sub-character above. The data were categorized into 96 classes, and each has 12 handwritten pictures of the main character with the sub-character above. Concerning the main handwritten character with the sub-character on the side, 564 images were retrieved. The data were grouped into 47, each having 12 handwritten pictures of the main character with the sub-characters on the side. For the main handwritten character with the sub-character below, 684 images were collected. The data was arranged into 57 classes, hence each has 12 handwritten pictures of the main character with the sub-characters below.

2.2. Data augmentation

To enhance model quality and mitigate overfitting, machine learning necessitates substantial volumes of data. In this study, image augmentation is applied to increase data variation. To train the neural network architecture effectively, data augmentation is necessary with the following configuration: rescale (first, scale down all chest X-ray images for size reduction purposes), shear range and rotation range ($\pm 20^\circ$), zoom range ($\pm 15\%$), and fill mode. These enhancements are carried out to include each image in both the training and testing datasets. A depiction of the data augmentation procedure is presented in Figure 5.



Figure 5. The data collection process

2.3. Data preprocessing

Data preprocessing is a series of steps performed on data for it to be useful as input during training. The first data preprocessing carried out was cropping, which is for removing areas not needed because the Lampung characters image obtained still has a lot of space. The second is resizing, a process that ensures all data has the same size. The third is grayscale which assists in changing the Lampung character image into

gray. Fourth is binarization, which involves converting the gray image into a binary or black-and-white in order to see the areas with objects and the image background. The fifth is erosion and it is for reducing pixels at the boundaries of an object. Furthermore, the sixth is dilation, which is used to add pixels to the boundary of an object in the input image. The final result of a series of data preprocessing is shown in Figure 6.



Figure 6. The result of preprocessing data

2.4. Principal component analysis

Principal component analysis (PCA) is a method employed in various applications, including reducing dimensionality, compressing data, extracting features, and visualizing data. Additionally, it permits the transformation of a set of correlated variables denoted as X into a smaller number, y , specifically the uncorrelated variable known as the principal component, where $y < X$, while retaining the variability inherent in the original data. Among the features of PCA is its capability to compress images, reducing their size while preserving maximum quality [17], [18]. The steps involved in PCA are outlined as follows.

Standardization: involves calculating all dimensions of the dataset, excluding labels [19]. This process scales the data, ensuring that each variable contributes equally to the analysis. In (1), z denotes the scaled value, x represents the initial value, while μ and σ denote the mean and standard deviation, respectively.

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

Covariance matrix computation: calculation of the covariance matrix involves assessing the covariance between two dimensions [19]. However, in a three-dimensional dataset, denoted as A , B , and C , the relationships between dimensions A and B , B and C , and A and C are evaluated. The covariance between the two variables X and Y was determined using in (2):

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - X')(Y_i - Y') \quad (2)$$

Computation of eigenvectors and corresponding eigenvalues: The eigenvectors and their corresponding eigenvalues were computed based on the covariance matrix [19]. Additionally, the appropriateness of the eigenvalues determines the scaling factor for the eigenvectors. The eigenvector of matrix A is represented by the vector u , satisfying (3) provided.

$$Au = \lambda u \quad (3)$$

Here, λ denotes the eigenvalue. This signifies the definition of the linear transformation, and the formula can be expressed differently as shown in (4):

$$(A - \lambda I)u = 0 \quad (4)$$

Where I is the identity matrix.

2.5. Convolutional neural network

The CNN architecture's fundamental components were formed by employing sequences of convolution, normalization, and pooling layers. Meanwhile, the convolution layer was responsible for feature extraction and normalization. The pooling component was utilized to normalize and down-sample the relevant local features [20], [21]. The convolutional layer operates by segmenting the image into smaller sections, often referred to as receptive fields. This assists in extracting motifs and the output are shown in (5) [22]:

$$F_l^k = [f_l^k(1,1), \dots, f_l^k(p,q), \dots, f_l^k(P,Q)] \quad (5)$$

Where F_l^k is the input feature matrix for l^{th} layer and K^{th} kernel. The pooling layer was used for the feature map down-sampling, while average and maximum techniques were utilized for the process [20]. The operation process is defined as shown in (6):

$$Z_l^k = g_p(F_l^k) \quad (6)$$

Where Z_l^k is the combined feature map for l^{th} layer and K^{th} kernel. The fully connected layer (FC) establishes complete connections with all activations in the preceding layer. It also provides discriminatory features for classifying the input image into various classes. CNN training uses an optimization process, which is similar to traditional machine learning techniques [23]. Two widely recognized training techniques for neural networks are stochastic gradient descent with momentum (SGDM) and adaptive moment estimation (ADAM) [24], [25]. This current study utilized the CNN network and the two main functions performed include fire_inceptor and fire_squeezer. The flowchart of fire_inceptor is shown in Figure 7.

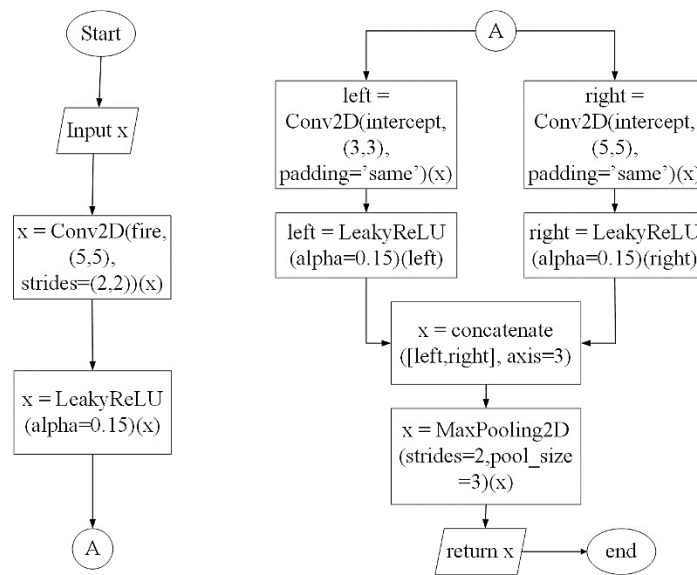


Figure 7. Fire inceptor flowchart

The input data for this function is denoted as variable x , which undergoes convolution using 5×5 convolutional layers and 2×2 shifts. The outcome is then fed into the LeakyReLU layer with an alpha value of 0.15. Subsequently, the output is handled in two different paths: x is convoluted separately on the left and right. The left-side processing entails a 3×3 convolution, yielding a result stored in the left variable for subsequent processing in the LeakyReLU layer with an alpha value of 0.15. Conversely, the right-side processing involves a 5×5 convolutional layer, producing a result stored in the right variable for processing in the LeakyReLU layer with an alpha value of 0.15. The outcomes of the left and right variables are then merged and substituted back into variable x . Finally, the results undergo processing in the MaxPooling layer with a shift of 2 and a size of 3.

The function of the fire_squeezer resembles that of the fire_inceptor, albeit with some parameter adjustments. Initially, variable x is introduced and convoluted with a size of 16×16 and a shift of 1×1 . The resulting outputs are subsequently passed through the LeakyReLU layer with an alpha value of 0.15. Moreover, the outcomes of the LeakyReLU layer undergo processing in two distinct paths, wherein x is convoluted separately on the left and right. On the left side, the processing entails employing a convolutional layer with a size of 1×1 , with the result stored in the left variable for further processing in the LeakyReLU layer with an alpha value of 0.15. Meanwhile, on the right side, the processing involves a convolutional layer with a size of 3×3 , the outcome of which is stored in the right variable for subsequent processing in the LeakyReLU layer with an alpha value of 0.15. The results obtained from the left and right variables are then merged and reintroduced into variable x . Additionally, the combined output is further processed in the

MaxPooling layer with a shift of 2 and a size of 3. These two functions collectively form the constructed CNN model, with its corresponding flowchart illustrated in Figure 8.

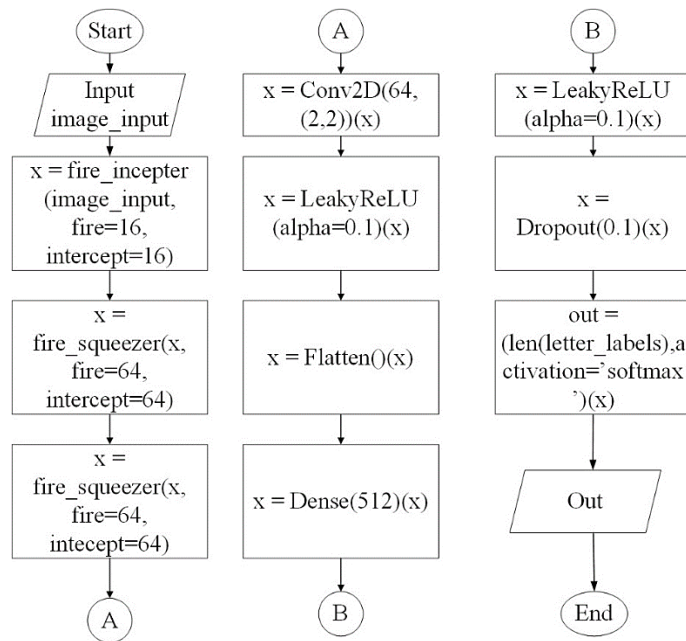


Figure 8. CNN model flowchart

2.6. Support vector machine

The SVM is a binary classification technique rooted in statistical learning theory. In an n-dimensional space, the input data xi (where i ranges from 1 to k) is categorized into either class 1 or class 2. For instance, the label -1 signifies class 2, whereas -1 and +1 denote classes 1 and 2, respectively [26]. When the input data is linearly separable, the separation of hyperplanes can be represented as shown in (7):

$$f(X) = w^T x + b \tag{7}$$

In (7), w represents the weight vector in n-dimensional space, while b stands for the scalar or bias value. This formula is employed to ascertain the maximum margin required for separating the positive and negative classes. The decision function is illustrated in (8):

$$f(x) = sgn(w^T x + b) \tag{8}$$

If the two classes are linearly separable, the hyperplane that achieves the maximum margin between them is represented as in (9):

$$\left\{ \begin{array}{l} \text{Minimize } \frac{1}{2} \|w\|^2 \\ \text{Subject to } y_i (w_i x_i + b) \geq 0 \end{array} \right\} \tag{9}$$

When the SVM parameters are appropriately configured, the classification performance experiences enhancement. The training of this model involved solving the optimization problem described by in (10):

$$\left\{ \begin{array}{l} \text{Maximize } L(\alpha) = \sum_{i=1}^k \alpha_i - \frac{1}{2} \sum_{i,j=0}^k \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{Subject to } \sum_{i=1}^k y_i \alpha_i = 0 \quad \alpha_i \geq 0 \text{ for } i = 1 \dots k \end{array} \right\} \tag{10}$$

In this context, $K(x_i, x_j)$ represents the kernel function, and α_i signifies the Lagrange multiplier. If the data cannot be linearly separated, the kernel function's mapping is altered as outlined in (11):

$$K(x_i, x_j) = K(x_i, x_j) + \frac{1}{C} \delta_{ij} \quad (11)$$

In this context, C represents the penalty parameter, and its value can enhance classification performance, while δ_{ij} denotes the Kronecker symbol. In this investigation, the kernel function utilized is radial based, as in (12):

$$K(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / (2\sigma^2)) \quad (12)$$

Where σ represents the kernel parameter influencing the distribution complexity of the data in the feature space. Following the utilization of CNN, the subsequent step involves applying SVM for Lampung character recognition. The design flowchart for SVM is depicted in Figure 9.

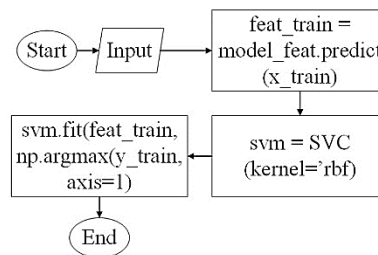


Figure 9. SVM model flowchart

The input data utilized for the SVM developed was sourced from the final layer of the CNN network. The subsequent step involves predicting the x_{train} data by initializing the SVM kernel utilized for Lampung character recognition. Following this, the SVM model that has been created is fitted.

2.7. Evaluation metrics

The performance metrics employed to assess the proposed method consist of accuracy, precision, recall, and F1-score [27], [28]. Additionally, comparisons were made with various proposed models, including DCNN (CNN+SVM) and PCA with SVM.

$$\frac{TP + TN}{TP + FP + FN + TN} \quad (13)$$

$$\frac{TP}{TP + FP} \quad (14)$$

$$\frac{TP}{TP + FN} \quad (15)$$

$$2 * \frac{(Recall * Precision)}{(Recall + Precision)} \quad (16)$$

The accuracy and precision of the model are determined with in (13) and (14). Moreover, in (15) was utilized for evaluating sensitivity or recall, while in (16) was employed to calculate the F1-Score.

3. RESULTS AND DISCUSSION

3.1. Experimental setup

In the experiment conducted, the dataset was split into 90% for training and 10% for testing purposes. The proposed network consists of 10 convolutional layers, as indicated in Table 1. The learning rate was established at 0.001, and based on experimental analysis, the maximum number of epochs was set to 30. The proposed architecture was implemented using Python and the Keras and Tensorflow libraries on an Intel® Core™ i7-2.8 GHz processor. Additionally, testing was performed using the NVIDIA GTX 1060 graphics processing unit (GPU) with 6 GB and 16 GB of RAM, respectively.

Table 1. A complete summary of the DCNN

Layer	Type	Output shape	Param #	Connected to
1	Input	69x69x4	-	-
2	Convolution2D	33x33x16	1616	Input
3	LeakyReLU	33x33x16	0	Convolution2D
4	Convolution2D_1	33x33x16	2320	LeakyReLU
5	Convolution2D_2	33x33x16	6416	LeakyReLU
6	LeakyReLU_1	33x33x16	0	Convolution2D_1
7	LeakyReLU_2	33x33x16	0	Convolution2D_2
8	Concatenate	33x33x16	0	LeakyReLU_1 LeakyReLU_2
9	MaxPooling2D	16x16x32	0	Concatenate
10	Convolution2D_3	16x16x64	2112	MaxPooling2D
11	LeakyReLU_3	16x16x64	0	Convolution2D_3
12	Convolution2D_4	16x16x64	4160	LeakyReLU_3
13	Convolution2D_5	16x16x64	36928	LeakyReLU_3
14	LeakyReLU_4	16x16x64	0	Convolution2D_4
15	LeakyReLU_5	16x16x64	0	Convolution2D_5
16	Concatenate_1	16x16x128	0	LeakyReLU_4 LeakyReLU_5
17	MaxPooling2D_1	8x8x128	0	Concatenate_1
18	Convolution2D_6	8x8x64	8256	MaxPooling2D_1
19	LeakyReLU_6	8x8x64	0	Convolution2D_6
20	Convolution2D_7	8x8x64	4160	LeakyReLU_6
21	Convolution2D_8	8x8x64	36928	LeakyReLU_6
22	LeakyReLU_7	8x8x64	0	Convolution2D_7
23	LeakyReLU_8	8x8x64	0	Convolution2D_8
24	Concatenate_2	8x8x128	0	LeakyReLU_7 LeakyReLU_8
25	MaxPooling2D_2	4x4x128	0	Concatenate_2
26	Convolution2D_9	3x3x64	32832	MaxPooling2D_2
27	LeakyReLU_9	3x3x64	0	Convolution2D_9
28	Flatten	576	0	LeakyReLU_9
29	Dense	512	295424	Flatten
30	LeakyReLU_10	512	0	Dense
31	Dropout	512	0	LeakyReLU_10
32	SVM	20	10260	Dropout

3.2. Data preparation

The dataset for Lampung characters obtained from documents and handwriting was converted into 69x69 pixels images and are saved in CSV format. The objective of this procedure is to organize data for both training and testing purposes. The division of the dataset is illustrated in Table 2.

Table 2. Data partition

No	Data	Partition	Number of characters	Total
1	Main character	Training images (90%)	468	520
		Testing images (10%)	52	
2	Main character with the sub-character above	Training images (90%)	2246	2496
		Testing images (10%)	250	
3	Main character with the sub-character side	Training images (90%)	1100	1222
		Testing images (10%)	122	
4	Main character with the sub-character below	Training images (90%)	1334	1482
		Testing images (10%)	148	
5	Handwritten main character	Training images (90%)	216	240
		Testing images (10%)	24	
6	Handwritten main character with the sub-character above	Training images (90%)	1037	1152
		Testing images (10%)	115	
7	Handwritten main character with the sub-character side	Training images (90%)	508	564
		Testing images (10%)	56	
8	Handwritten main character with the sub-character below	Training images (90%)	583	648
		Testing images (10%)	65	

3.3. Model training and validation

The system was developed utilizing the Keras CNN model with Tensorflow serving as the backend. Approximately 30 epochs were utilized in training the DCNN. Based on the new approach, the training process from one of the data achieved the best performance. This training requires approximately less than 360 seconds to reach a maximum accuracy of 99.37% in the 10th epochs as shown in Figures 10 and 11.

Hence, it can be inferred that the model achieved an exceptionally high accuracy level and an exceptionally low loss rate.

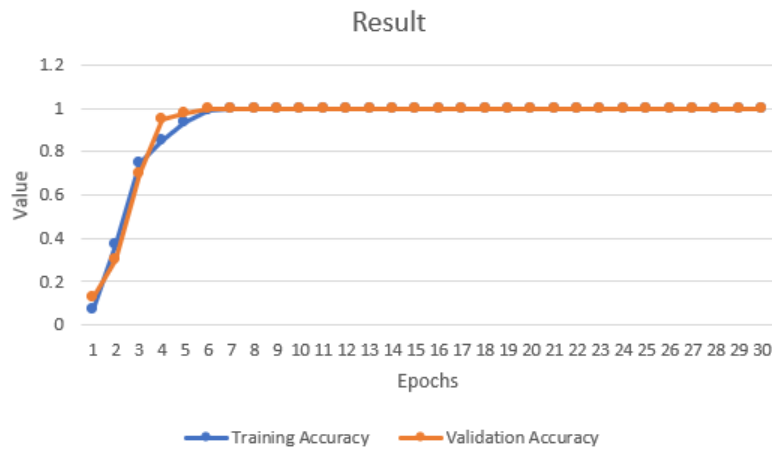


Figure 10. Accuracy curve of the proposed method

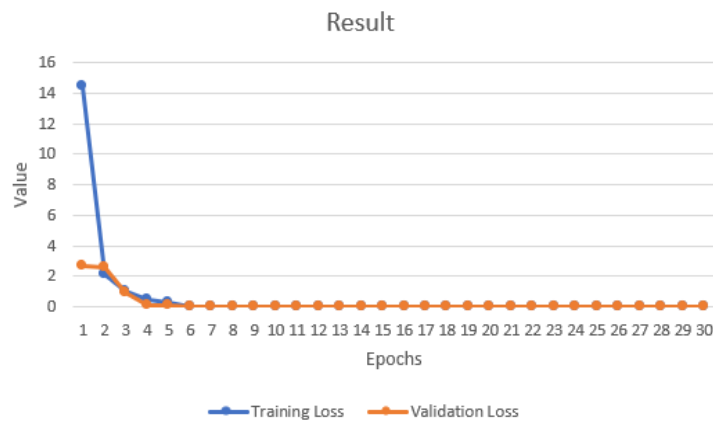


Figure 11. Loss curve of the proposed method

The confusion matrix displayed in Figure 12 served to consolidate the performance evaluation of the classification algorithm. It provides a summary of the prediction outcomes, detailing the count of correctly and incorrectly classified characters across each class. As per the summary within the matrix table, there were no incorrectly classified characters.

3.3. Comparison of result for all data

All data were successfully trained and tested. The proposed DCNN architecture outperformed PCA+SVM in training and testing for all data. Table 3 presents a comparison of the outcomes obtained from the proposed architecture. Based on the results displayed in Table 2, the DCNN demonstrated a commendable accuracy level, averaging 98.0% for training and 99.3% for testing. In contrast, PCA+SVM achieved average accuracy rates of 93.6% for training and 92.9% for testing, respectively.

3.4. Recognition results for all data

After the model has undergone several stages of training and testing, it achieves the best accuracy results. These results serve as the knowledge base used by the computer to recognize Lampung characters. The recognition results for all data were documented using the DCNN as shown in Figure 13. Table 4 shows the average accuracy of Lampung character recognition from the results of the proposed architecture.

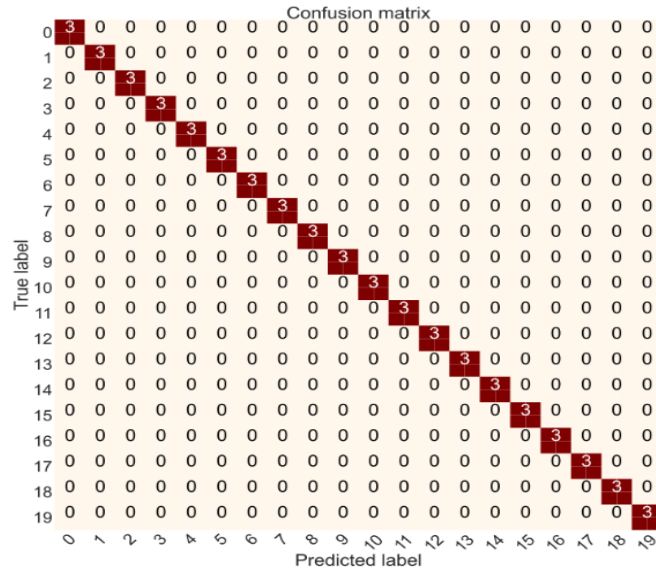


Figure 12. The confusion matrix of the proposed architecture

Table 3. Comparison of DCNN and PCA+SVM architecture for all data

No	Data	DCNN			PCA+SVM		
		Training (%)	Testing (%)	Time (s)	Training (%)	Testing (%)	Time (s)
1	Main character	99.8	99.3	160	95.0	93.0	-
2	Main character with the sub-character above	99.1	99.0	459	97.7	96.0	-
3	Main character with the sub-character side	99.4	99.2	309	92.8	90.6	-
4	Main character with the sub-character below	99.2	99.9	281	96.6	97.2	-
5	Handwritten main character	95.0	99.6	133	90.9	89.9	-
6	Handwritten main character with the sub-character above	99.8	98.4	366	93.3	90.8	-
7	Handwritten main character with the sub-character side	94.5	98.9	294	91.1	87.3	-
8	Handwritten main character with the sub-character below	96.7	99.8	264	92.6	93.3	-
	Average	98.0	99.3	283	93.6	92.9	-

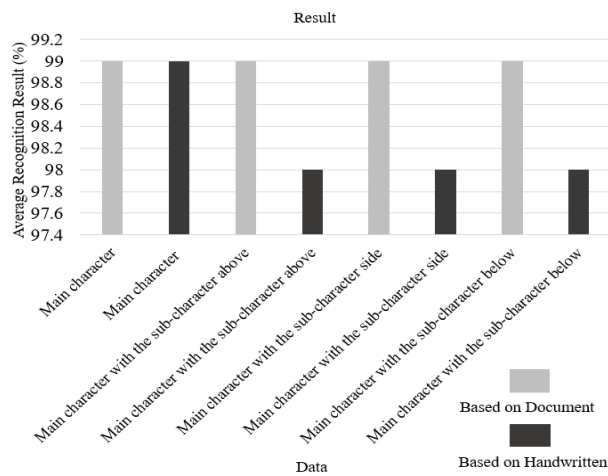


Figure 13. Recognition average result for all data

Table 4. Average accuracy of Lampung character recognition

No	Data	Accuracy (%)
1	Main character	99
2	Main character with the sub-character above	99
3	Main character with the sub-character side	99
4	Main character with the sub-character below	98
5	Handwritten main character	99
6	Handwritten main character with the sub-character above	98
7	Handwritten main character with the sub-character side	99
8	Handwritten main character with the sub-character below	98
Average		98.6

According to the findings depicted in Figure 13 and Table 4, the recognition of Lampung characters achieved an average accuracy of 99% for main characters derived from documents and handwriting, main characters with sub-characters positioned above based on documents, main characters with sub-characters positioned beside based on documents, and main characters with sub-characters positioned below. Consequently, the recognition outcomes for the 8 data samples yielded an average accuracy rate of 98.6%.

3.5. Comparison of result achieved with previous works

We carried out a comparison of several pieces of literature collected from diverse sources, encompassing methods, training data size, testing data size, and accuracy values. The proposed system has shown good results compared to previous works. According to Table 5, the test result with the best accuracy was 99.3%. Based on the comparison conducted on handwriting characters, methods, training size, testing size, and accuracy in several literature, we concluded that Lampung characters based on documents and handwriting, using DCNN with a training size of 6502, testing size of 832, and an accuracy of 99.3%, surpassed the previous research.

Table 5. Comparison of results achieved with previous works

Character	Previous work	Training size	Testing size	Accuracy (%)
Kannada [11]	HOG+DCNN	1954	488	98.8
Gujarat [12]	CNN	8000	2000	97.21
Devanagari [13]	HOG+NN	434	186	97.06
Bangla [14]	CNN	67500	7500	95.25
Tifinagh [15]	CNN	19968	4992	99.27
Jawi [16]	SVM+ED	125	28	90.4
Javanese [29]	RF	4200	1800	97.7
Bangla [30]	CNN	77614	19224	98.03
Lampung	DCNN	6502	832	99.3

4. CONCLUSION




In this paper, a document-based and handwritten Lampung character recognition system was developed using DCNN technology. The system was trained on 440 characters derived from 8 different data types, leading to high levels of accuracy in both training and testing phases with an average of 98.00% and 99.31% respectively. In addition, the system also achieved a fast execution time, with an average of 283 seconds, surpassing previous solutions in terms of both accuracy and speed. These results make the DCNN architecture suitable for use in recognizing Lampung characters and are expected to make it easier for Lampung people to recognize Lampung characters.

REFERENCES




- [1] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in *20th International Conference of Computer and Information Technology, ICCIT 2017*, Dec. 2017, vol. 2018, pp. 1–5, doi: 10.1109/ICCITECHN.2017.8281853.
- [2] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, doi: 10.1561/2200000006.
- [3] C. L. Liu, F. Yin, Q. F. Wang, and D. H. Wang, "ICDAR 2011 Chinese handwriting recognition competition," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, Sep. 2011, pp. 1464–1469, doi: 10.1109/ICDAR.2011.291.
- [4] D. Cireşan and U. Meier, "Multi-Column Deep Neural Networks for offline handwritten Chinese character classification," in *Proceedings of the International Joint Conference on Neural Networks*, Jul. 2015, vol. 2015, pp. 1–6, doi: 10.1109/IJCNN.2015.7280516.

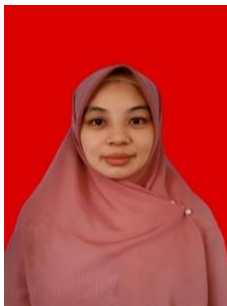
- [5] M. Shams, A. A. Elsonbaty, and W. Z. El Sawy, "Arabic handwritten character recognition based on convolution neural networks and support vector machine," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, pp. 144–149, 2020, doi: 10.14569/IJACSA.2020.0110819.
- [6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems*, vol. 4, pp. 3320–3328, 2014.
- [7] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3642–3649, doi: 10.1109/CVPR.2012.6248110.
- [8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2016, vol. 2016-December, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.
- [9] J. Suykens and V. Joss, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, pp. 293–300, 1999.
- [10] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems*, vol. 13, no. 1, pp. 409–415, 2001.
- [11] S. T. Narasimhaiah and L. Rangarajan, "Recognition of compound characters in Kannada language," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 6, pp. 6103–6113, Dec. 2022, doi: 10.11591/ijece.v12i6.pp6103-6113.
- [12] J. Pareek, D. Singhania, R. R. Kumari, and S. Purohit, "Gujarati Handwritten Character Recognition from Text Images," *Procedia Computer Science*, vol. 171, pp. 514–523, 2020, doi: 10.1016/j.procs.2020.04.055.
- [13] N. Singh, "An Efficient Approach for Handwritten Devanagari Character Recognition based on Artificial Neural Network," in *2018 5th International Conference on Signal Processing and Integrated Networks, SPIN 2018*, Feb. 2018, pp. 894–897, doi: 10.1109/SPIN.2018.8474282.
- [14] P. Chakraborty, A. Islam, M. A. Yousuf, R. Agarwal, and T. Choudhury, "Bangla Handwritten Character Recognition Using Convolutional Neural Network," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 132, pp. 721–731, 2022, doi: 10.1007/978-981-19-2347-0_56.
- [15] L. Niharmine, B. Outtaj, and A. Azouaoui, "Tifinagh handwritten character recognition using optimized convolutional neural network," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 4, pp. 4164–4171, Aug. 2022, doi: 10.11591/ijece.v12i4.pp4164-4171.
- [16] F. Arnia, K. Saddami, and K. Munadi, "Moment invariant-based features for Jawi character recognition," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 3, pp. 1711–1719, Jun. 2019, doi: 10.11591/ijece.v9i3.pp1711-1719.
- [17] A. K. Jean, M. Diarra, B. A. Bakary, G. Pierre, A. K. Jérôme, and U. B. Franche-comté, "Application based on Hybrid CNN-SVM and PCA- SVM Approaches for Classification of Cocoa Beans," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, pp. 231–238, 2022, doi: 10.14569/IJACSA.2022.0130927.
- [18] M. Usman, S. Ahmed, J. Ferzund, A. Mehmood, and A. Rehman, "Using PCA and Factor Analysis for Dimensionality Reduction of Bio-informatics Data," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 5, 2017, doi: 10.14569/ijacsa.2017.080551.
- [19] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro, "CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection," *IEEE Access*, vol. 8, pp. 91916–91923, 2020, doi: 10.1109/ACCESS.2020.2994762.
- [20] A. M. Ismael and A. Şengür, "Deep learning approaches for COVID-19 detection based on chest X-ray images," *Expert Systems with Applications*, vol. 164, p. 114054, Feb. 2021, doi: 10.1016/j.eswa.2020.114054.
- [21] F. Demir, A. Sengur, and V. Bajaj, "Convolutional neural networks based efficient approach for classification of lung diseases," *Health Information Science and Systems*, vol. 8, no. 1, p. 4, Dec. 2020, doi: 10.1007/s13755-019-0091-3.
- [22] F. A. Khan *et al.*, "Chest x-ray analysis with deep learning-based software as a triage test for pulmonary tuberculosis: a prospective study of diagnostic accuracy for culture-confirmed disease," *The Lancet Digital Health*, vol. 2, no. 11, pp. e573–e581, Nov. 2020, doi: 10.1016/S2589-7500(20)30221-1.
- [23] T. Rahman *et al.*, "Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray," *Applied Sciences (Switzerland)*, vol. 10, no. 9, p. 3233, May 2020, doi: 10.3390/app10093233.
- [24] M. J. Horry *et al.*, "COVID-19 Detection through Transfer Learning Using Multimodal Imaging Data," *IEEE Access*, vol. 8, pp. 149808–149824, 2020, doi: 10.1109/ACCESS.2020.3016780.
- [25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Jul. 2017, vol. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
- [26] M. Turkoglu, "COVIDDetectioNet: COVID-19 diagnosis system based on X-ray images using features selected from pre-learned deep features ensemble," *Applied Intelligence*, vol. 51, no. 3, pp. 1213–1226, Mar. 2021, doi: 10.1007/s10489-020-01888-w.
- [27] M. M. Abubakar, B. Z. Adamu, and M. Z. Abubakar, "Pneumonia Classification Using Hybrid CNN Architecture," in *2021 International Conference on Data Analytics for Business and Industry, ICDABI 2021*, Oct. 2021, pp. 520–522, doi: 10.1109/ICDABI53623.2021.9655918.
- [28] Ž. Vujović, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021, doi: 10.14569/IJACSA.2021.0120670.
- [29] M. A. Rasyidi, T. Baryyah, Y. I. Riskajaya, and A. D. Septyani, "Classification of handwritten javanese script using random forest algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1308–1315, Jun. 2021, doi: 10.11591/eei.v10i3.3036.
- [30] T. Ghosh *et al.*, "Bangla handwritten character recognition using mobilenet v1 architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2547–2554, Dec. 2020, doi: 10.11591/eei.v9i6.2234.




BIOGRAPHIES OF AUTHORS

Panji Bintoro    is a Lecturer at the Department of Software Engineering, Aisyah University "Aisyah University" Pringsewu, Lampung, Indonesia and has served as a lecturer since 2022. He received a Bachelor of Computer Degree from the Department of Information Engineering, Ahmad Dahlan University (UAD), Yogyakarta-Indonesia in 2019. Then, he received a Master's degree in Computer Science from the Department of Computer Science and Electronics, Gadjah Mada University (UGM), Yogyakarta-Indonesia in 2022. His research interests are mainly in the fields of artificial intelligence, computer vision, and digital image processing. He has been the author or/co-author of more than 10 research publications. He can be contacted at email: panjibintoro09@aisyahuniversity.ac.id.






Zulkifli Zulkifli    is a lecturer at the Department of Informatics Engineering, Faculty of Technology and Informatics, Aisyah University, Indonesia. Currently conducting research on software testing and software engineering. He can be contacted at email: zulkifli@aisyahuniversity.ac.id.



Fitriana Fitriana    is a lecturer in the Department of Medical Records and Health Information, Faculty of Health, Aisyah University, Indonesia, who is currently studying for a Doctoral program at UNS in the biomedical study program. Currently, research is being carried out on the benefits of young coconut water against preeclampsia by testing the biomarkers HIF1A, mTOR, A1M, and HbF. She can be contacted at email: fitriana@aisyahuniversity.ac.id.



Sukarni Sukarni    is a lecturer in the Department of Medical Records and Health Information, Faculty of Health, Aisyah University, Indonesia, who is currently studying the Doctoral program at the UNS in biomedical study program. Currently doing research entitled the Effect of Glucomannan on Macrosomia in a Pregnant Rat Model (*Rattus Norvegicus*) with Diabetes. She can be contacted at email: sukarni@aisyahuniversity.ac.id.