

Augmentation of Additional Arabic Dataset for Jawi Writing and Classification Using Deep Learning

Safrizal Razali^{1*}, Kahlil Muchtar^{1,2}, Muhammad Hafiz Rinaldi¹, Yudha Nurdin¹, and Aulia Rahman¹

¹Department of Electrical and Computer Engineering, Faculty of Engineering, Universitas Syiah Kuala

²Telematics Research Center, Universitas Syiah Kuala

Jln. Tengku Syech Abdur Rauf No. 7 Kopelma Darussalam, Banda Aceh, Aceh, Indonesia 23111

*e-mail: safrizal.razali@usk.ac.id

Abstract—This research aims to create an additional dataset containing Arabic characters for writing Jawi script and to train classification models using deep learning architectures such as InceptionV3 and ResNet34. The initial stage of the study involves digital image processing to obtain the additional Arabic character dataset from several sources, including HMBD, AHAWP, and HUCD, encompassing various connected and disconnected forms of Jawi script. Image processing includes steps such as preprocessing to enhance image quality, segmentation to separate Arabic characters from the background, and augmentation to increase dataset variability. Once the dataset is formed, we train the models using appropriate training data for each InceptionV3 and ResNet34 architecture. The classification evaluation results indicate that the model with ResNet34 architecture achieved the best performance with an accuracy of 96%. This model successfully recognizes Jawi script accurately and consistently, even for classes with similar shapes. The main contribution of this research is the availability of the additional Arabic character dataset that can be utilized for Jawi script recognition and performance assessment of various deep learning models. The study also emphasizes the importance of selecting the appropriate architecture for specific character recognition tasks. The research findings affirm that the model with ResNet34 architecture has excellent capability in recognizing the additional Arabic characters for writing Jawi. The results of this research have the potential to support further developments in Jawi character recognition applications and provide valuable insights for researchers in the field of character recognition sourced from Arabic characters. Dataset augmentation results can be accessed at <https://singkat.usk.ac.id/g/En0skCKGAR>.

Keywords: additional arabic characters, classification model training, deep learning architectures, jawi script

I. INTRODUCTION

In the Nusantara region, particularly in the Aceh region of Indonesia, various ancient relics bear inscriptions in the Jawi script. These relics manifest in diverse forms, including tombstones, currency, books, and Islamic manuscripts. Some of these historical artifacts, inscribed with the Jawi script, date back as far as 600 years [1]. Preserving these artifacts is of utmost importance, and one effective method involves digitizing them into images. The Jawi script, when represented in digital image form, can undergo conversion into machine-readable text through Optical Character Recognition (OCR) technology.

OCR technology requires a comprehensive dataset to accurately recognize all characters. Currently, one available Jawi script dataset is from the research conducted by [2]. This dataset only consists of 10 samples for each separate Jawi character. More extensive datasets are needed to enable machine learning or deep learning to perform better classification. Based on a search through search engines, there is no online repository found for

Jawi script datasets, which could facilitate researchers in conducting Jawi character recognition.

There are several variants of the Arabic script, and one of them is the Jawi script. In addition to the Jawi script, there are the Farsi and Urdu scripts, which are also variants of the Arabic script. Currently, there are several online available handwritten Arabic script datasets that researchers or interested parties can access for free, including the dataset from the study by [3]. This research acquired Arabic script images from 125 contributors, with dimensions of 300×300 pixels, and encompassing 115 classes, including numeral classes. This dataset is named the Arabic Handwritten Characters' Dataset (HMBD). Following the acquisition and pre-processing process, they applied deep learning using two Convolutional Neural Network (CNN) architectures, HMB1 and HMB2. They also employed optimization, regularization, and dropout techniques. In addition to the HMBD dataset, this study also utilized the CMATER and AIA9k datasets. The research using the HMB1 architecture yielded a classification success rate of 97.3%, while HMB2 achieved 96.8%. Furthermore,

dataset augmentation improved the classification rate to 98.4% using the HMB1 architecture. Meanwhile, research on Arabic script conducted by [4] focused on constructing a dataset containing handwritten Arabic script characters, words, and paragraphs, without conducting classification tests. The resulting dataset from this study included 53,199 characters, 8,144 words, and 241 paragraphs, all in image format, named the Handwritten Arabic Alphabets, Words, and Paragraphs Dataset (AHAWP). Another study on Arabic script was performed by [5] and [6], which created a dataset of handwritten Urdu script in the Nasta'liq style, encompassing separate, connected characters, and numerals, named the Handwritten Urdu Character Dataset (HUCD). The HUCD dataset was trained using 74,285 samples and evaluated with 21,223 samples in the test dataset, achieving a recognition rate of 98.82% across 133 classes. Several other studies on Arabic script and its variants were conducted by [7], [8], and [9].

Studies on the Jawi script conducted by [10] and [11] did not yield comprehensive datasets as they only employed isolated handwritten Jawi characters, with only 10 data samples per character. Another research effort to create a Jawi script dataset was undertaken by [12], though this dataset is not available online.

From all the mentioned studies and other undisclosed ones, a comprehensive Jawi script dataset has not yet been identified. Even searches across various databases such as Kaggle Datasets, IFN/ENIT, CVL datasets, IAM, UCI Datasets for Machine Learning, Data.gov, and others have failed to uncover a Jawi script dataset. This critical gap poses significant challenges for researchers and practitioners interested in the preservation and recognition of the Jawi script.

In our efforts to address this issue, we implemented a dataset augmentation approach. Dataset augmentation is a strategy aimed at expanding and enhancing the diversity of training data without the need to create entirely new data [13]. In this context, augmentation focuses on manipulating existing datasets by applying various transformations to images or other data. By doing so, we can increase the quantity and diversity of training data without the manual collection of new data. The use of augmentation to broaden and improve dataset variation can be cost-effective compared to manual techniques of collecting new data [14].

Several traditional techniques commonly used in dataset augmentation involve flipping, cropping, and rotation [14], [15]. In addition to these techniques, there are more advanced and distinct augmentation methods, such as random erasing and mixing images [16]. The random erasing technique, as described by [17], randomly removes specific parts of an image to generate training data with varying levels of occlusion. This research demonstrates that Random Erasing significantly reduces the risk of overfitting and enhances performance in image classification, object detection, and person recognition tasks.

Furthermore, the augmentation technique involving

mixing images, as conducted by Ionue [18], is employed to generate new images by combining two randomly selected images from the training set. Additionally, research [19] proposes a more general data augmentation method involving linear combinations of pairs of example data, which has proven effective in reducing overfitting. Moreover, research [20] introduces the Random Image Cropping and Patching (RICAP) data augmentation technique, involving random cropping and patching of images. The results of this study indicate that RICAP achieves state-of-the-art test results in various image processing tasks.

However, the research approach we employ differs slightly from the research conducted by [17]. We replace the dot components of Arabic script images with corresponding dot images. These dot images are extracted separately from the Arabic script. Our research also varies from the methods used in studies [19] and [20], which combine two images with different objects into a unified image. In contrast, our research combines two images: the original script image and the dot image, as part of the dataset augmentation process. To achieve this, we utilize existing Arabic script datasets such as HMBD, AHAWP, and HUCD, known for their success in recognizing handwritten Arabic scripts. It is important to note that our approach focuses on augmentation rather than creation.

The primary goal of our research is not only to augment the dataset but also to evaluate the performance of additional Arabic characters used to write Jawi by utilizing deep learning techniques. This endeavor will not only fulfill our research needs but also provide a valuable resource for researchers and enthusiasts committed to preserving and recognizing the Jawi script.

The selection of ResNet34 and InceptionV3 architectures for this research is based on their high performance. Research conducted by [21] strategically applied ResNet34 to address the challenge of vanishing gradients and further equipped it with a self-attention mechanism. This augmentation resulted in improved feature extraction, leading to commendable accuracy, particularly in complex classification tasks. The study by [22] underscores the vital role of InceptionV3 in the field of image-based sentiment analysis. Its ability to focus on specific regions within images, such as human faces, has proven to be a significant advantage. Furthermore, the model achieves exceptional accuracy on the CK+ dataset. These compelling findings provide strong reasons to adopt ResNet34 and InceptionV3 as the foundational architectures for this research, aligning with the research's central goal of Arabic character classification for Jawi writing.

II. BACKGROUND

A. Jawi Script

The Jawi script, a variant of Arabic script, consists of 35 characters, comprising 29 pure Arabic characters

Table 1. Original Arabic Script [2]

No	Script name	Jawi	Forms			
			Isolated	Start	Middle	End
1	Alif	ا	ا	-	-	ا
2	Ba	ب	ب	ب	ب	ب
3	Ta	ت	ت	ت	ت	ت
4	Tsa	ث	ث	ث	ث	ث
5	Jim	ج	ج	ج	ج	ج
6	Hha	ح	ح	ح	ح	ح
7	Kha	خ	خ	خ	خ	خ
8	Dal	د	د	-	-	د
9	Dzal	ذ	ذ	-	-	ذ
10	Ra	ر	ر	-	-	ر
11	Zai	ز	ز	-	-	ز
12	Sin	س	س	س	س	س
13	Syin	ش	ش	ش	ش	ش
14	Shad	ص	ص	ص	ص	ص
15	Dhad	ض	ض	ض	ض	ض
16	Tho	ط	ط	ط	ط	ط
17	Zho	ظ	ظ	ظ	ظ	ظ
18	Ain	ع	ع	ع	ع	ع
19	Ghain	غ	غ	غ	غ	غ
20	Fa	ف	ف	ف	ف	ف
21	Qaf	ق	ق	ق	ق	ق
22	Kaf	ك	ك	ك	ك	ك
23	Lam	ل	ل	ل	ل	ل
24	Mim	م	م	م	م	م
25	Nun	ن	ن	ن	ن	ن
26	Wau	و	و	-	-	و
27	Ha	ه	ه	ه	ه	ه
28	Hamzah	ء	ء	ء	ء	ء
29	Ya	ي	ي	ي	ي	ي

and an additional 6 characters to accommodate the Malay language, which cannot be represented by the Arabic script. Similar to Arabic script, Jawi script is written cursively from right to left. Some characters in the Jawi script have different forms as they can be connected at the start, middle, and end, as seen in Table 1 and Table 2.

Research on OCR for the Jawi script is currently limited, whereas there is a considerable amount of research on other scripts. Other variants of the Arabic script are still in official use, such as Urdu and Farsi, making them more attractive to researchers. In contrast, the Jawi script is no longer used in official state activities. Currently, the Jawi script is mainly employed in Islamic religious education.

B. Data Augmentation

Data augmentation is an essential technique in the development of machine learning and deep learning models aimed at expanding and enriching the training dataset [16] and [23]. The fundamental concept behind

Table 2. Additional Arabic Script for Writing Jawi [2]

No	Script name	Jawi	Forms			
			Isolated	Start	Middle	End
1	Nya	نْ	نْ	نْ	نْ	نْ
2	Ca	چ	چ	چ	چ	چ
3	Nga	غْ	غْ	غْ	غْ	غْ
4	Pa	پ	پ	پ	پ	پ
5	Ga	گ	گ	گ	گ	گ
6	Va	وْ	وْ	-	-	وْ

data augmentation is to generate additional variations of existing data without altering the essential information contained in the data. By introducing such variations, trained models have a better chance of understanding real-world diversity and can generalize more effectively when faced with previously unseen situations.

In the context of image data, data augmentation can generally be categorized into two main methods: basic image manipulation and deep learning approaches [16] and [15]. Basic image manipulation methods include techniques such as kernel filters, geometric transformations, random erasing, color space transformations, and mixing images. On the other hand, deep learning approaches involve techniques like adversarial training, neural style transfer, and GAN data augmentation. Understanding the differences between these basic and deep learning methods allows researchers and machine learning practitioners to select augmentation techniques that suit their specific training dataset expansion needs and model performance improvements.

One key aspect of data augmentation is geometric transformations, including flipping, rotation, and cropping, enabling the training dataset to have variations in orientation, size, and object positions. Conversely, photometric transformations like color changes and edge enhancement help introduce variations in image appearance and contrast. Additionally, the use of kernel filters allows more complex image processing, such as sharpening or blurring images, which can be valuable in enhancing the model's robustness against various real-world visual effects.

Beyond these basic techniques, advanced approaches like mixing images and random erasing introduce more complex variations and focus on constructing stronger feature representations in the model. Mixing images involves merging two images by taking the pixel value averages, a technique known as "SamplePairing," which has proven effective in reducing error rates, especially in scenarios with limited data [19] and [20]. This approach allows the model to learn from the variations that arise when different images are combined, ultimately aiding in better generalization.

Meanwhile, random erasing aims to prevent overfitting by randomly removing portions of pixels from an image [17]. Consequently, the model is compelled to pay more attention to the entire image and learn more descriptive features. This helps the model become more resilient to

overfitting and results in stronger feature representations in the training dataset.

Emerging techniques such as adversarial training, GAN-based data augmentation, and neural style transfer open doors for more creative exploration in creating training data variations. By understanding and combining these various techniques, machine learning researchers and practitioners can create diverse and representative training datasets, ultimately helping machine learning models become more reliable and effective in facing real-world challenges. Data augmentation is a crucial tool in the machine learning toolbox, enhancing model performance and ensuring success in real-world applications.

C. Deep Learning

Deep learning is one of the fields of machine learning. In recent years, deep learning has gained prominence among researchers. Gradually, deep learning has become the most widely used computational approach [24]. One of the strengths of deep learning is its ability to process large amounts of data. One of the most widely used deep learning architectures is the Convolutional Neural Network (CNN). CNN is an evolution of the Multi-Layer Perceptron (MLP) designed to process two-dimensional data. CNN is a part of the Neural Network used in image classification and image processing. CNN is categorized as a type of Deep Neural Network due to its high network depth and extensive use in digital image processing. CNN methods generally focus on increasing the scale of one network dimension, such as width, depth, or image resolution. CNN has a superior approach to image classification and object recognition by utilizing matrix multiplication to identify patterns in an image. CNN comprises four main layers: convolutional layer, pooling layer, activation function, and fully connected layer [25]. Several CNN architectures commonly used by researchers include ResNet34 and InceptionV3.

The Residual Neural Network (ResNet) is a type of architecture in deep learning that enables the creation of artificial neural networks resembling human thought patterns. The ResNet architecture is referred to as state-of-the-art due to its impressive capabilities in classification, object detection, and semantic segmentation. ResNet architecture employs pre-trained models, thus saving time by eliminating the need for configuring layers within it. The working principle of ResNet involves forming a network deeper than usual, achieving an optimized number of layers to overcome the vanishing gradient problem (VGP). VGP is a challenge stemming from instances where the gradient learned by the model cannot reach the first layer due to repeated multiplicative factors that prevent the initial layer from receiving any gradient [26]. ResNet offers various architecture variants with layer counts starting from 18, 34, 50, 101, and up to 152 layers. This study utilizes the ResNet-34 architecture, consisting of 34 layers. Each layer employs a 3×3 kernel with feature sizes of 64, 128, 256, and 512.

The InceptionV3 architecture is one of the CNN models used in image analysis and object detection. InceptionV3 is part of the Inception group, which improves label smoothing and additional classification to share label information within the network. Label smoothing adjusts classification during training to estimate the effect of label dropout, aiming to make the predicted classes align with the criteria [27]. In the development of deeper networks, the InceptionV3 architecture is an advancement from the InceptionV2 architecture.

III. METHOD

Our proposed method will be thoroughly discussed in this study's methodology. We chose this method because it aligns closely with our research objectives, which include augmenting the Jawi character dataset and evaluating classification performance using deep learning. We will provide a detailed explanation of the steps, procedures, tools, and materials used in this research.

A. Research Tools and Materials

The research employed a set of laptops equipped with CUDA GPUs as its primary tools. These laptops were crucial for processing digital images of the Jawi script dataset. In addition, they played a pivotal role in performing the recognition of the Jawi script through the implementation of deep learning techniques and OpenCV.

The research materials employed in this study encompass the Arabic script dataset, HMBD, as used in the research by [3], the Arabic script dataset, AHAWP, as employed in the research by [4], along the Urdu script dataset, HUCD, used in the studies by [5] and [6]. These datasets are stored in the form of digital images. Subsequently, these datasets are spatially engineered to produce an additional six Arabic characters for writing Jawi, resulting in 22 forms, as indicated in Table 2.

The augmented dataset is then divided into three parts: training data and validation data for model creation, and testing data for model evaluation. The proportion of division for the augmented dataset is 70% for training data, 20% for validation data, and 10% for testing data [28]. Figure 1 illustrates the complete process conducted in this study.

B. Preprocessing and Dataset Augmentation

The acquisition of datasets includes the Arabic script dataset HMBD [3] containing 6,270 characters, the Arabic script dataset AHAWP [4] with 789 characters, and the Urdu script dataset HUCD [5] and [6] featuring 1,482 characters. These datasets are acquired for processing in the spatial domain to generate Jawi characters as presented in Table 2. The complete sources of the Jawi script datasets can be found in Table 3.

Preprocessing and data augmentation were executed utilizing OpenCV. These processes concentrated on

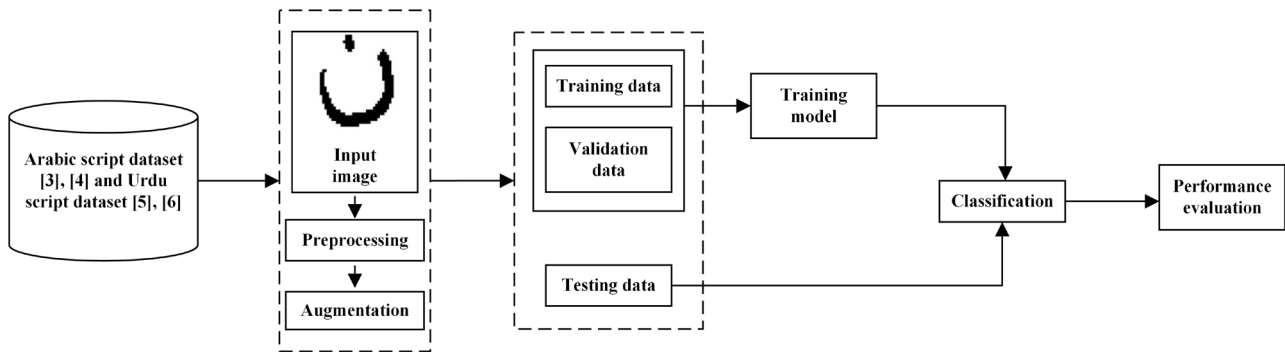


Figure 1. Block diagram of the research methodology

the primary shapes of the additional Arabic characters intended for writing Jawi. For instance, consider the case of the extra Arabic character “ن” used for Jawi writing. It shares the same core shape as the Arabic character “ن” (nun). Here, the main shape “ن” was extracted from the “ن” character image. This main shape extraction encompassed the identification and removal of existing dots, followed by the incorporation of three dots in the character’s middle. This augmentation approach was applied to all supplementary Arabic characters meant for Jawi writing, incorporating 22 forms comprising separate, initial-connected, middle-connected, and end-connected variations. All the augmented Jawi characters were stored in a database sized 300×300, chosen to align with the dimensions of the frequently used HMBD images in this study.

Augmenting additional Arabic characters for Jawi writing, as presented in Table 2, necessitates images of extra dots that will be superimposed onto the original Arabic script images from HMBD, AHAWP, and the original Urdu script of HUCD. These images of extra dots required for dataset augmentation consist of single-dot and triple-dot configurations. They were generating these single-dot and triple-dot datasets initiated with the Otsu method [29] to threshold the images, followed by utilizing Connected Component Labeling (CCL) with eight connectivity to segment the HMBD images. CCL was employed to exclude objects larger than 1,500 pixels,

as they were deemed primary components of characters, leaving only single-dot or triple-dot object images. The augmentation of the dot dataset persisted by cropping solely the sections containing dot objects. For example, the “Ca” script (چ) augmentation involves flipping images with three dots, resulting in a dot dataset with two dots above and one dot below.

Creating additional Arabic script characters for Jawi writing, derived from the HMBD Arabic script dataset, involved reading the image dataset and undergoing preprocessing steps, such as converting images to grayscale via the Otsu method. This grayscale conversion aimed to streamline the dataset augmentation process and mitigate image processing complexity. Another preprocessing phase encompassed segmenting the images using CCL, intended to identify and label each dot object within the script images. For supplementary Jawi characters with one dot, the dot object was positioned on the upper section of the script. Conversely, for additional Jawi characters with three dots, the dot objects were identified as separate objects, sized between 10 to 700 pixels in each processed image. Objects larger than 700 pixels were deemed main components of the script, not dots. The dataset augmentation was realized by overlaying the original dot positions of the HMBD Arabic script images with the previously extracted dots.

The AHAWP Arabic script dataset did not offer a complete array of Arabic characters. Within the AHAWP dataset, only one main part form of several Arabic characters sharing the same main part form was available. For instance, while a dataset for the character “Ain” (ع) existed, no corresponding dataset was available for the character “Ghain” (غ). In contrast to the preprocessing procedure for the HMBD Arabic script dataset, the augmentation process of supplementary Arabic script characters for Jawi writing, originating from AHAWP, commenced by creating a 300×300 white template, aligning with the size of the dataset derived from the HMBD Arabic script dataset. This step became necessary due to the varying sizes of the AHAWP dataset images, ranging from a maximum of 720×128 to a minimum of 48×128. The process continued with the Otsu method-based thresholding of AHAWP images, followed by identifying contours forming lines encompassing the binary image objects. Upon contour identification, sections

Table 3. Source of additional Arabic characters data for writing Jawi

No	Script name	Forms			
		Isolated	Start	Middle	End
1	Nya (ن)	ن [3]=466	ن [3]=468	ن [3]=460	ن [3]=468
2	Ca (چ)	چ [3]=472	چ [3]=472	چ [4]=820	چ [3]=477
3	Nga (غ)	غ [3]=920	غ [4]=820	غ [3]=458	غ [3]=459
4	Pa (ف)	ف [3]=464	ف [3]=464	ف [3]=451	ف [3]=463
5	Ga (ك)	ك [5]=750	ك [3]=65 [5]=753	ك [3]=77 [5] =691	ك [5]=776
6	Va (و)	و [3]=398	-	-	و [3]=397

of the image housing objects were cropped, eliminating the background without objects. Following cropping, the resulting object was preserved, with the subsequent step entailing the placement of the image at the center of the previously created white template. The dataset augmentation progressed by adding other dots from the previously extracted dot dataset onto the AHAWP script image to generate supplementary Arabic script characters for writing Jawi.

The augmentation of additional Arabic script characters for Jawi writing, drawn from the Urdu script source HUCD, involved resizing the size from 64×64 to 300×300. The dataset augmentation process included placing dots from the previously extracted dot dataset onto the HUCD dataset to create the desired supplementary Arabic script characters for Jawi writing.

C. Training Model

Training and validation of the model were conducted using augmented Jawi script images. The training and validation processes were performed on a computer equipped with a CUDA GPU to maximize the utilization of deep learning techniques. This encompassed model initialization, data processing, hyperparameter tuning, and training until optimal convergence was achieved. The deep learning architectures employed in this Jawi script OCR system were ResNet-34 and InceptionV3.

Before conducting the model training, the dataset of additional Arabic script images for writing Jawi was organized and stored in a structured folder with image dimensions of 300×300. This process ensured the consistency of image sizes used by the model throughout training. During the training of this additional Arabic script dataset for writing Jawi, pre-trained model weights were not employed; rather, a new set of model weights was trained. The output layer was adjusted to accommodate the 22 classes of additional Arabic script for writing Jawi, as indicated in Table 2.

Throughout the model training process, the selection of hyperparameters became a crucial factor. Hyperparameters, distinct from model parameters, are predefined settings influencing the learning process, and they are not learned from the data during training [30]. These hyperparameters played a pivotal role in model performance. Table 4 displays the hyperparameters employed in this research, providing explicit values for parameters such as learning rate, batch size, and optimizer. The training process was

Table 4. Hyperparameters and loss function for training the model used in this study

No	Hyperparameter	Value
1	Optimizer	Adam
2	Learning rate	0,01
3	Batch size	32
4	Epoch	50
5	Loss function	Cross entropy

executed for 50 epochs, allowing the model to adapt its internal parameters during this time. During training on the training data, model weights were updated based on the gradients of the loss function using backpropagation. Meanwhile, when evaluating the model on validation data, model weights remained unchanged and were solely used for computing loss and accuracy. Throughout the training process, the best-performing model would be saved if its loss and accuracy surpassed the previously trained model. This emphasis on hyperparameters underscores their significance in shaping the model's learning process and overall performance.

D. Testing and Performance

After the completion of model training, testing of the trained model is performed. The testing process begins by loading the model that achieved the best accuracy, and this loaded model is then used to evaluate the testing data.

The next step involves assessing the performance of the testing data on the constructed model. Performance testing is conducted using metrics such as accuracy, precision, recall, and F1-score. To calculate these performance metrics, a confusion matrix is employed. A confusion matrix is an evaluation method used to measure the performance of a multi-class classification model. The confusion matrix tallies the number of correct and incorrect predictions for each class by comparing the predicted results against the actual conditions of the data used. For the multi-class problem of additional Arabic characters for the Jawi script, involving 22 classes, the confusion matrix will take the form of 22×22, where each row represents the true class, and each column represents the predicted class. This confusion matrix includes various combinations like True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) for each existing class. Table 5 provides an example of the confusion matrix for the additional Jawi script class “Ca Isolated” (؄).

Table 5. Confusion matrix for multi-class additional arabic characters for Jawi script, class "Ca Isolated " (؄) [33], [34]

Class	Prediction class				
	Ca Isolated (؄)	Nya Isolated (؄)	...	Ga Isolated (؄)	Va Isolated (؄)
Actual class	Ca Isolated (؄)	TP (؄)	FN (؄)	...	FN (؄)
	Nya Isolated (؄)	FP (؄)	TN (؄)	...	TN (؄)
	...	...	...	...	...
	Ga Isolated (؄)	FP (؄)	TN (؄)	...	TN (؄)
	Va Isolated (؄)	FP (؄)	TN (؄)	...	TN (؄)

Accuracy is one of the most popular metrics in multi-class classification and is directly computed from the confusion matrix. The accuracy formula takes the sum of TP and TN as the numerator and the total sum of all entries in the confusion matrix as the denominator. TP and TN are the elements classified correctly by the model and are on the main diagonal of the confusion matrix, while the denominator also includes all elements off the main diagonal that were misclassified by the model. In the context of classifying additional Arabic characters for the Jawi script with 22 classes, accuracy depicts how well the model can correctly classify all these classes. For instance, considering the class “Ca Isolated” (چ) as shown in Table 5, accuracy indicates the extent to which the model predicts testing data from the actual class to the predicted class, TP (چ). Equation (1) is used to calculate accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision is the fraction of relevant instances among the retrieved instances. On the other hand, recall is the fraction of relevant instances that have been retrieved. Both precision and recall are based on relevance. Referring to Table 5 in the context of classifying additional Arabic characters for the Jawi script with 22 classes, precision depicts the extent to which the fraction of “Ca Isolated” (چ) characters predicted correctly from the total predicted instances of that class. Meanwhile, recall illustrates how much of the fraction of “Ca Isolated” (چ) characters have been successfully found from all the actual instances of “Ca Isolated” (چ) characters present in the dataset. Equations (2) and (3) are used to calculate precision and recall.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

The F1-Score measures the performance of a classification model based on the confusion matrix, combining the Precision and Recall metrics through the concept of harmonic mean. The F1-Score formula can be interpreted as a weighted average between Precision and Recall, where the F1-Score value reaches its best at 1 and worst at 0. The relative contribution of Precision and Recall is equal in F1-Score, and this harmonic mean is useful for finding the optimal balance between these two metrics. Table 5 shows the confusion matrix for the “Ca Isolated” (چ) class. From this confusion matrix, we can calculate the values of Precision and Recall for this class. Using these Precision and Recall values, we can compute the F1-Score, which provides an overview of how well the model can accurately classify “Ca Isolated” (چ), considering the balance between Precision and Recall. The F1-Score provides important information about the model’s ability to recognize the “Ca Isolated” (چ) class

effectively, especially if the data quantity of this class is relatively smaller compared to other classes in the dataset. Equation (4) is used to calculate the F1-Score.

$$F1-Score = \frac{2}{Precision^{-1} + Recall^{-1}} \quad (4)$$

IV. RESULT AND DISCUSSION

The first step carried out in this study is the augmentation of an additional Arabic character dataset for writing Jawi. This process involves processing the Arabic script dataset HMBD, the Arabic script dataset AHAWP, and the Urdu script dataset HUCD to augment a comprehensive Jawi character dataset. During this process, the transformation of Arabic and Urdu character images into the desired additional Arabic characters for writing Jawi is undertaken. The outcomes of dataset augmentation encompass the number of successfully augmented additional Arabic characters for writing Jawi and the appearance of each of these additional Arabic characters for writing Jawi, both connected and isolated, as presented in Table 2. As an illustration, several examples of original Arabic characters and the additional Arabic characters for writing Jawi resulting from the augmentation process are also provided.

A. Augmentation of Dots Dataset

Before the creation of the augmentation of the dots dataset for writing the Jawi script, the generation of the dots dataset was conducted. The dots dataset comprises single dots and triple dots. The triple dots consist of two forms: one with a single upper dot and two lower dots, and the other with two upper dots and a single lower dot, used to create the script “Ca” (چ).

The single-dot dataset is augmented from the HMBD dataset of the script classes “Ba” (ب) and “Dhad” (ض), this choice is solely due to these characters having a single dot. The augmentation of the three-dot dataset also originates from the HMBD dataset of the script classes “Syin” (ش) and “Tsa” (ث); the selection of these classes is also purely based on the fact that these characters have three dots.

The process of creating the dot dataset starts with reading the original HMBD images, which are then converted to binary images using the Otsu method. The subsequent step involves performing eight-connectivity Connected Component Labeling (CCL) to group interconnected pixels and form distinct objects. The augmentation process continues by applying a filter to remove objects larger than 1,500 pixels, as they are considered to be the main parts of the characters, leaving only dot-shaped objects.

For the specific augmentation of the script “Ca” (چ), a set of three dots is required, with two dots on top and one dot at the bottom. These dots are augmented from images with the arrangement of one dot on top and two dots at the bottom by vertically flipping the image. Examples of the resulting dot augmentation can be seen in Figure 2.

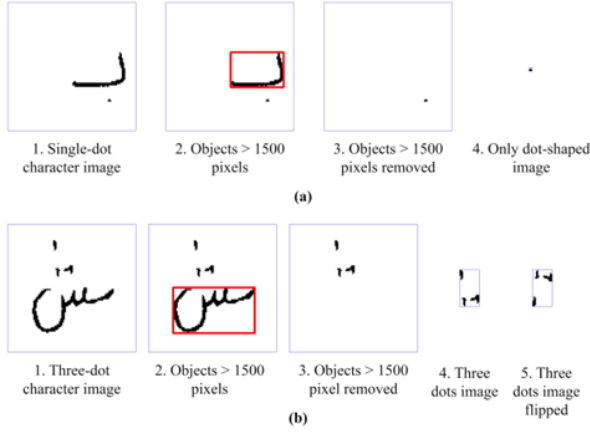


Figure 2. (a) The process of augmenting the single-dot dataset, (b) The process of augmenting the three-dot dataset

B. Augmentation of Additional Arabic Script Dataset for Jawi Writing

In the augmentation of an additional Arabic script dataset for Jawi writing, the Arabic script dataset HMBD is predominantly utilized and serves as the benchmark for the dataset size from other sources, namely AHAWP and HUCD. This dataset is selected due to its consistent image size, and it boasts a comparatively superior average resolution when compared to other dataset sizes.

The augmentation of an additional Arabic script dataset for Jawi writing from the HMBD dataset, which comprises single dots, including the script “Ga” (گ) originating from the script “Kaf” (ك) and the script “Va” (و) originating from “Wau” (و), is approached slightly differently than the additional Arabic script for Jawi writing with three dots. The distinction lies in how the dots are positioned to form the Jawi script. The augmentation of the Jawi script dataset from the HMBD dataset with single dots begins by reading the color image and converting it to grayscale. Once the image is in grayscale, the coordinates of the global minimum point are determined based on the lowest grayscale level vertically and horizontally within the processed image array. These global minimum coordinates serve as a reference for placing the additional dots on the processed image. The dots are placed 15 pixels below the global minimum coordinates. The process continues with checking the dimensions of the image being processed, ensuring that the dot is larger than the original image dimensions. If adding the dot results in larger dimensions, the dot image from the dot dataset is not added, and the process continues to the next image. From Table 6, it can be observed that only the character “Kaf” (ك) in its connected and middle-connected writing forms from the HMBD dataset is chosen. This is due to the fact that the writing style of the character “Ga” (گ) in its separated and end-connected forms is distinct from the original Arabic script writing style in the HMBD dataset. The separated form of the character “Kaf” (ك) in the HMBD dataset is written as “Kaf” (ك), while the end-connected form of the character “Kaf” (ك) in the

Table 6. The number of additional Arabic script data for Jawi writing used in classification using deep learning

No	Jawi script	Forms			
		Isolated	Start	Middle	End
1	Nya (ن)	ن [3]=361	ن [3]=392	ن [3]=393	ن [3]=384
2	Ca (چ)	چ [3]=397	چ [3]=391	چ [4]=398	چ [3]=384
3	Nga (غ)	غ [3]=396	غ [4]=391	غ [3]=376	غ [3]=374
4	Pa (ف)	ف [3]=385	ف [3]=390	ف [3]=391	ف [3]=378
5	Ga (ك)	ك [5]=395	ك [3]=38 [5]=356	ك [3]=46 [5]=344	ك [5]=387
6	Va (و)	و [3]=397	-	-	و [3]=396

HMBD dataset is written as “Kaf” (ك). The augmentation of the additional Arabic script dataset for Jawi writing from the HMBD dataset with single dots encountered several failures because, after adding the dot, the resulting image dimensions exceeded those of the original HMBD image.

In the process of augmenting an additional Arabic script dataset for Jawi writing with three dots, encompassing script “Nya” (ن), “Ca” (چ), “Nga” (غ), and “Pa” (ف) from the HMBD dataset, color images are read from the dataset. Three-dot images are randomly selected from the dot dataset. The process continues by ensuring that the HMBD dataset images are not larger than the dot dataset images. Subsequently, the images from the HMBD dataset are segmented using the CCL method to separate interconnected objects. After the objects are separated, object filtering is carried out with criteria ranging from 10 pixels to 700 pixels in size. Objects falling outside this range are disregarded. Upon selecting objects that meet the criteria, bounding boxes are created to determine the middle point coordinates of the bounding box. These middle point coordinates serve as a reference for overlaying the corresponding dot image onto the image from the HMBD dataset. If images fail to meet the size criteria of 10 pixels to 700 pixels, the images from the HMBD dataset are disregarded. By implementing this method, certain images remain unprocessed into additional Arabic script data for Jawi writing due to instances of errors, such as dot sizes not meeting the size criteria (10 pixels to 700 pixels), lacking dots, and middle point coordinates of the bounding box being too close to the edges. Consequently, overlaying the dot would cause the size of the additional Arabic script dataset for Jawi writing to exceed 300×300. The provided illustration in Figure 3 showcases the augmentation process applied to the HMBD dataset. Panel (a) demonstrates the augmentation process for generating single-dot Jawi characters from the HMD dataset source. Similarly, Panel (b) illustrates the augmentation process for creating three-dot Jawi characters from the HMBD

dataset source.

The process of augmenting an additional Arabic script dataset for Jawi writing from the AHAWP Arabic script dataset, specifically targeting the “Ca Middle” (ا) and “Nga Start” (ن) scripts, is conducted due to the absence of the primary components of these characters, namely “ا” and “ن”, in the HMBD dataset. The augmentation of datasets for these characters initiates by generating a completely white-colored template with a grayscale level of 255, sized at 300×300. The purpose of this white template is to address the non-uniform image dimensions present in the AHAWP Arabic script dataset. The subsequent steps involve reading images from the AHAWP dataset and applying Otsu’s thresholding method to create binary images. The thresholding process aims to segregate objects from the background by determining an appropriate threshold through the Otsu algorithm. Next, contours are detected on the binary images from AHAWP using contour tracing along the object edges. Based on these contours, image cropping operations are employed to retain only the segments containing the objects. The cropped images are then overlaid onto the previously generated white template. These 300×300-sized images on the white template are further combined with one of the three dot points from the previously augmented dot dataset. The dot placement is determined based on the search for global minimum coordinates both vertically and horizontally within the image. The dots are positioned 25 pixels below the vertical global minimum coordinates. The process continues by assessing the dimensions of the processed image to ensure that the dots are larger than the original image dimensions. If adding the dots results in the image dimensions exceeding the original size, the dot image from the dot dataset is excluded, and the process proceeds to the next image. The provided illustration in Figure 4 showcases the augmenting process applied to the AHAWP dataset.

The augmentation of an additional Arabic script dataset for writing Jawi characters from the Urdu HUCD dataset begins by transforming the image resolution from 64×64 to 300×300. Once the images are of size 300×300, the process of determining the placement location of dots is carried out by searching for the global minimum coordinates both vertically and horizontally, following the conversion of images into grayscale format. The dot addition location is set 25 pixels below the global minimum coordinates vertically. The selection of dots to be added is done randomly. Dot placement is executed if, after adding the dot, the image size remains 300×300; if the size becomes larger, the dot addition process is disregarded. Figure 5 illustrates the augmenting process of creating Jawi characters from the HUCD dataset source.

The evaluation method used in dataset creation involves a meticulous manual visual inspection to ensure the quality and accuracy of the generated images. Images that have successfully passed the overlaying dot phase without enlarging their size beyond 300×300 will be included in the dataset collection that will be utilized in the processes

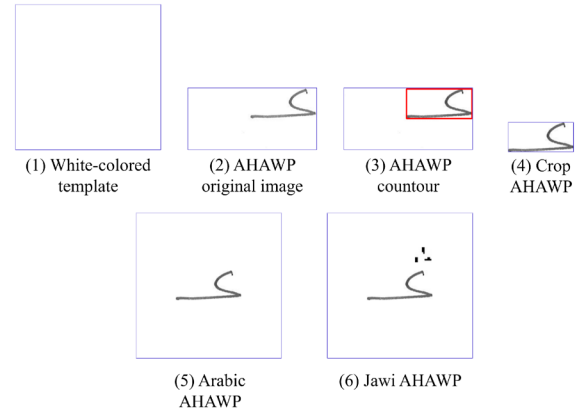


Figure 4. The process of augmenting Jawi characters from the AHAWP dataset

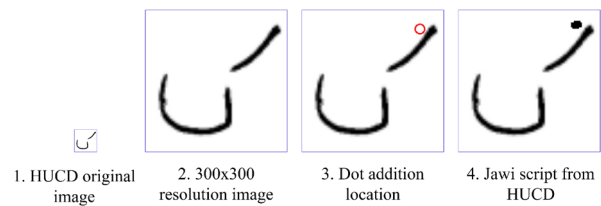


Figure 5. The process of augmenting Jawi characters from the HUCD dataset

of training, evaluation, and testing in deep learning. With this visual inspection method, it can be ensured that every image in the augmented dataset meets the required quality standards and characteristics as an additional Arabic script dataset for writing Jawi.

The number of successfully created datasets was then manually selected to produce a per-class dataset ranging from 350 to 450 data points to align with the average dataset count of HMBD. The dataset quantity utilized for deep learning classification comprised the Arabic script dataset HMBD [3] with 6,270 characters, the Arabic script dataset AHAWP [4] with 789 characters, as well as the Urdu script dataset HUCD [5] and [6] with 1,482 characters. The distribution of these datasets per class can be observed in Table 6. Subsequently, these datasets were divided into three portions for each class with a data training proportion of 70%, data validation of 20%, and data testing of 10%. Dataset augmentation results can be accessed at <https://singkat.usk.ac.id/g/En0skCKGAR>.

C. Training Model

In this research, the training of the Arabic script classification model for Jawi writing was conducted using only the ResNet34 and InceptionV3 architectures. The selection of these architectures was based on their popularity among researchers, as they have demonstrated high accuracy and are widely adopted as starting points for developing new models [31] and [32]. The training results and the best epoch for each of these architectures can be observed in Table 7.

The use of 50 epochs in this research constitutes

Table 7. Training results and best epoch

Architecture	Epoch	Train Loss	Train Acc	Val Loss	Val Acc
InceptionV3	35	0.0894	0.9700	0.1844	0.9493
ResNet34	46	0.0024	0.9995	0.2115	0.9563

an initial testing phase that was conducted to assess the model's performance. As a fundamental research in its experimental stage, this number of epochs was chosen to comprehend how the model responds to the data and specific settings. These preliminary results will serve as the basis for determining whether further epochs are necessary for subsequent training. The importance of learning curve convergence is acknowledged; however, within the context of initial testing, we aimed to establish an initial understanding of our model and dataset before deciding whether more epochs are required. The utilization of 50 epochs is intended to provide a foundation for subsequent decisions regarding the optimal number of epochs for our model.

The training of the InceptionV3 architecture for the classification of additional Arabic script characters for Jawi writing was conducted for 50 epochs with hyperparameters as presented in Table 4. Figure 6 illustrates the training and validation loss curves of the InceptionV3 architecture. From the figure, it can be observed that at the beginning of training (epoch 1), both training and validation loss values are relatively high. This indicates that the model is still struggling to generalize well on unseen data. However, as epochs progress, both training and validation loss tend to decrease significantly. This suggests that the model gradually learns to represent the data better, improving its generalization capability. At certain points (around epochs 6 to 8), the validation loss seems to slightly increase before decreasing again. This phenomenon might be attributed to temporary overfitting, where the model adapts too closely to the training data, leading to reduced performance on validation data. Nevertheless, as the performance on the training data continues to improve, the model eventually overcomes overfitting and enhances its performance on the validation data.

Figure 7 depicts the training and validation accuracy curves of the InceptionV3 architecture over 50 epochs. From the figure, it can be observed that the accuracy of the training data consistently increases significantly as epochs progress. This improvement indicates that the model effectively learns to recognize patterns in the training data and becomes more precise in classification. However, at the beginning of training, the accuracy of the validation data appears to be unstable and tends to fluctuate. This is related to the fluctuations in validation loss observed earlier. This instability suggests that the model is not yet consistently generalizing to unseen data. Nevertheless, after a few epochs, the accuracy of the validation data also experiences enhancement and becomes more stable. This phenomenon demonstrates that the model is starting

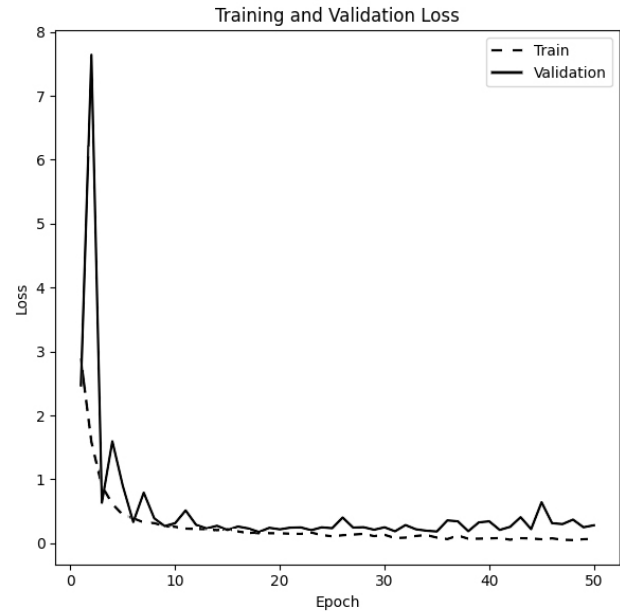


Figure 6. Training and validation loss of the inceptionV3 architecture

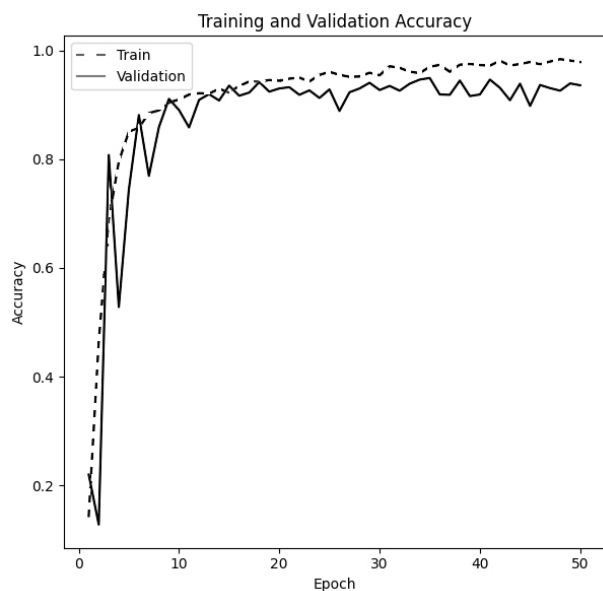


Figure 7. Training and validation accuracy of the inceptionV3 architecture

to grasp common features within the data, enabling it to accurately recognize new, unseen data.

As seen in Table 7, at epoch 35, the validation accuracy reaches its highest value of 0.9493. This epoch demonstrates optimal performance on the validation data. Additionally, at epoch 35, the validation loss is also relatively low, at 0.1844, indicating that the model not only achieves high accuracy but also avoids overfitting at this point. Epoch 35 can be considered the best epoch for the InceptionV3 model with the given data and hyperparameters. At this juncture, the model achieves a fine balance between high accuracy on the validation data and low loss on that data.

Training using the ResNet34 deep learning architecture with the additional Arabic script dataset for Jawi writing was also conducted over 50 epochs. Figure 8 illustrates

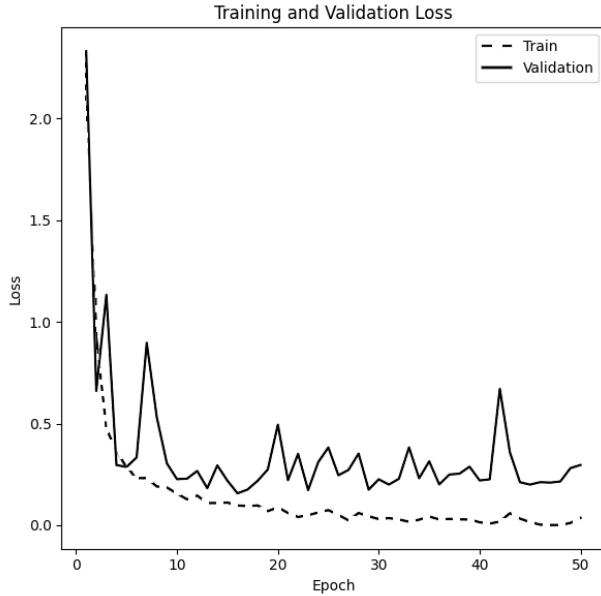


Figure 8. Training and validation loss of the resNet34 architecture

the training and validation loss curves of the ResNet34 architecture over 50 epochs. From the figure, it is evident that the ResNet34 architecture excels in overcoming overfitting issues. Throughout the training process, both the training loss and validation loss consistently decrease from epoch to epoch. Validation loss even reaches its lowest point at epoch 16, with a value of 0.1565, demonstrating the model's capacity to generalize well to unseen data.

Figure 9 illustrates the training and validation accuracy curves of the ResNet34 architecture over 50 epochs. From the figure, it is evident that the accuracy of the training data consistently increases with each epoch, reaching a high accuracy of nearly 0.9980 by the end of training. Furthermore, the accuracy of the validation data experiences steady improvement and reaches its peak value at epoch 46, with a value of 0.9563.

From Table 7, it is observed that at epoch 46, the ResNet34 architecture achieves its best performance with a validation loss of 0.2115 and a validation accuracy of 0.9563. Epoch 46 becomes the optimal point for the model in classifying unseen data.

Through the analysis of the performance of three different deep learning architectures, InceptionV3 and ResNet34, for the classification of additional Arabic script data for Jawi writing, it is found that the InceptionV3 architecture achieves the highest validation accuracy at epoch 35 with a value of 0.9493 and the lowest validation loss of 0.1844. Despite initial fluctuations and temporary overfitting, the model successfully addresses these issues and attains a fine balance between high accuracy on the validation data and low loss. Meanwhile, the ResNet34 architecture demonstrates excellent performance in mitigating overfitting and achieves the lowest validation loss at epoch 16 with a value of 0.1565, along with the highest validation accuracy at epoch 46 with a value of 0.9563. Thus, ResNet34 emerges as a superior choice for the classification task on the additional Arabic script

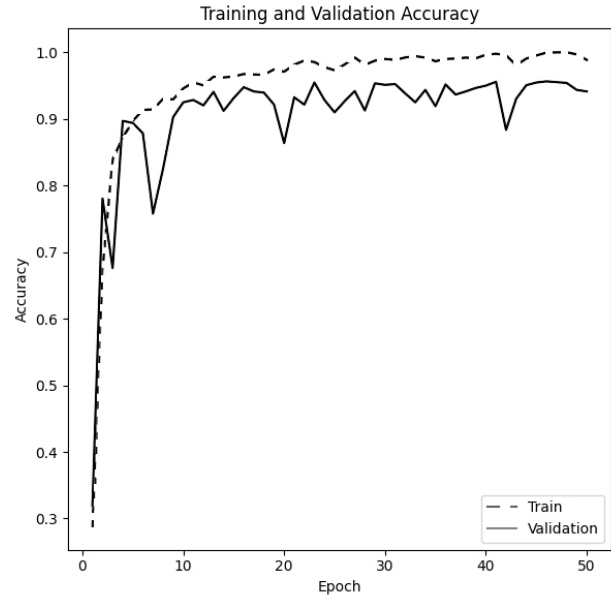


Figure 9. Training and validation accuracy of the ResNet34 architecture

dataset for Jawi writing compared to other architectures.

D. Performance of Additional Arabic Script Classification for Writing Jawi

In this research, the performance of the InceptionV3, and ResNet34 model architectures in recognizing additional Arabic script characters for writing Jawi in both separated and connected forms at the beginning, middle, and end is thoroughly investigated. The classification results based on the confusion matrix table and the evaluation of the classification models are compared and analyzed carefully.

Figure 10 depicts the confusion matrix of the classification results using the InceptionV3 model. From the analysis of the confusion matrix, it can be observed that characters in classes such as “Nya Start” (نـ), “Nya End” (ـن), “Ca Isolated” (چ), “Ca Start” (چـ), “Ca Middle” (ـچ), “Nga Start” (غـ), “Ga Isolated” (گـ), “Ga Middle” (ـگ), “Ga End” (ـگ), “Va Isolated” (ز), and “Va End” (ـز) are classified correctly, indicating that the model performs very well for these classes. However, some classes such as “Nya Middle” (نـ), “Nga Middle” (غـ), “Pa Middle” (پـ), “Pa End” (ـپ), and several other letters exhibit lower classification results, with some character images not being classified correctly.

Analysis of the evaluation results of the classification model using InceptionV3, as shown in Table 8, reveals that the classes with the highest F1-scores are “Nya Start” (نـ), “Nya End” (ـن), “Ca Middle” (چـ), “Nga Start” (غـ), “Ga Isolated” (گـ), and “Ga Middle” (ـگ) with a value of 1.00. Conversely, the “Pa Middle” (پـ) class exhibits the lowest F1-score with a value of 0.69. The potential reason for the variation in classification performance among classes might be due to the similarity in the shapes of characters from some other classes, especially in the connected form, as indicated in Figure 11.

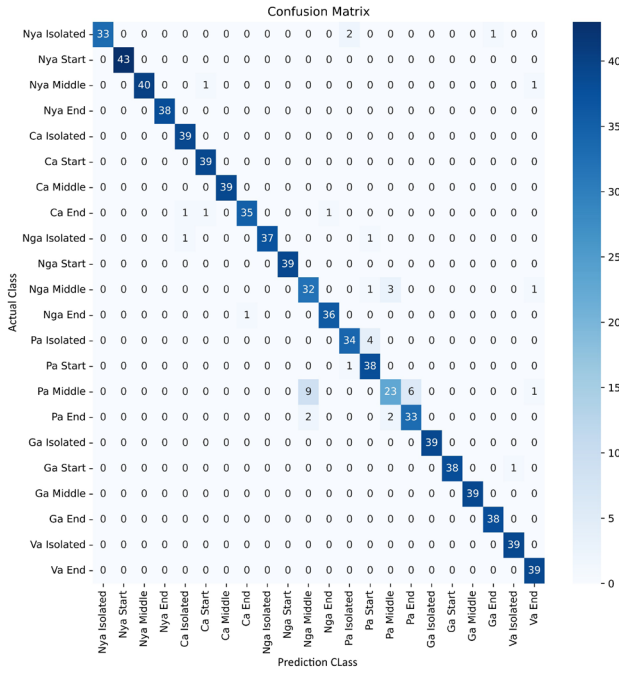


Figure 10. Confusion matrix of the classification model using inceptionV3

Based on the evaluation results, the InceptionV3 model exhibits outstanding performance with an accuracy value of 95%. This signifies that the model adeptly classifies approximately 95% of the data from the entire dataset employed in this evaluation. Moreover, the accuracy, macro average, and weighted average values are equal, underlining that the classification model has consistently and uniformly classified the additional Arabic script data for writing Jawi across all classes. It's important to note that in the context of macro-average, each class is treated equally in computing the average. This means that the evaluation metric, whether it's precision, recall, or F1-score, is calculated independently for each class, and then averaged without considering the proportion of samples in each class. This gives each class an equal say in the final metric, irrespective of the number of data points in each class. It provides a balanced perspective of the model's performance across all classes. On the other hand, the weighted average takes into account the proportion of samples in each class during the computation of the average. Classes with more data points have a larger influence on the final metric. Therefore, the evaluation metric is calculated for each class, and the contributions of each class are weighted by the proportion of samples they represent in the entire dataset. This is particularly important when dealing with imbalanced datasets, where some classes might have significantly more data than others.

In addition to using the InceptionV3 architecture, the evaluation of the performance of classifying additional Arabic characters for writing Jawi was also carried out using the ResNet34 architecture. Figure 12 represents the confusion matrix of the classification results using the ResNet34 model. From the analysis of this confusion matrix, it can be observed that characters in most classes

Table 8. Evaluation results of model classification using inceptionV3

No	Class	Precision	Recall	F1-Score	Support
1	Nya Isolated	1,00	0,92	0,96	36
2	Nya Start	1,00	1,00	1,00	43
3	Nya Middle	1,00	0,95	0,98	42
4	Nya End	1,00	1,00	1,00	38
5	Ca Isolated	0,95	1,00	0,97	39
6	Ca Start	0,95	1,00	0,97	39
7	Ca Middle	1,00	1,00	1,00	39
8	Ca End	0,97	0,92	0,95	38
9	Nga Isolated	1,00	0,95	0,97	39
10	Nga Start	1,00	1,00	1,00	39
11	Nga Middle	0,74	0,86	0,80	37
12	Nga End	0,97	0,97	0,97	37
13	Pa Isolated	0,92	0,89	0,91	38
14	Pa Start	0,86	0,97	0,92	39
15	Pa Middle	0,82	0,59	0,69	39
16	Pa End	0,85	0,89	0,87	37
17	Ga Isolated	1,00	1,00	1,00	39
18	Ga Start	1,00	0,97	0,99	39
19	Ga Middle	1,00	1,00	1,00	39
20	Ga End	0,97	1,00	0,99	38
21	Va Isolated	0,97	1,00	0,99	39
22	Va End	0,93	1,00	0,96	39
Accuracy			0,95	852	
Macro Avg		0,95	0,95	0,95	852
Weighted Avg		0,95	0,95	0,95	852

were successfully classified correctly, with high precision, recall, and F1-score values for most classes. Some classes, such as “Nga Middle” (ﻋ), exhibited slightly lower results in classification, with a few examples being misclassified.

In the analysis of the evaluation results of the classification model using ResNet34, as shown in Table 9, it was found that the classes with the highest F1-scores are “Nya Start” (ﻧ), “Nya Middle” (ﻧ), “Nya End” (ﻧ), “Ca Start” (ﻉ), “Ca Middle” (ﻉ), “Nga Isolated” (ﻏ), “Nga End” (ﻏ), and “Ga Isolated” (ﻏ), with a value of 1.00. Conversely, the classes “Nga Middle” (ﻏ) and “Pa Isolated” (ﻑ) showed lower F1-scores with values of 0.85 and 0.84, respectively. The potential reason for the difference in classification performance among classes might be due to the similarity of character shapes within certain classes, especially in the “Nga Middle” (ﻏ) class. Visually similar character shapes can confuse the model

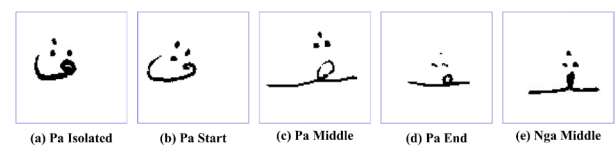


Figure 11. Examples of similarity in the forms of characters from several classes

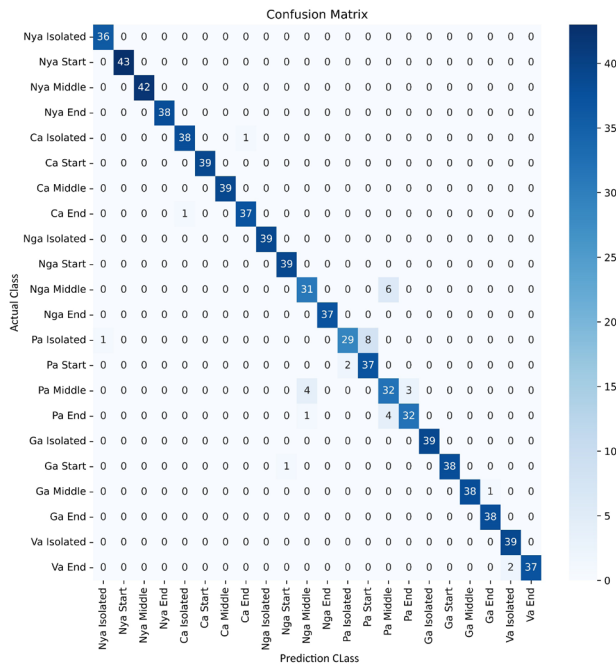


Figure 12. Confusion matrix of the classification model using resNet34

and result in misclassification.

Based on the evaluation results, the ResNet34 model

Table 9. Evaluation results of model classification using resNet34

No	Class	Precision	Recall	F1-Score	Support
1	Nya Isolated	0,97	1,00	0,99	36
2	Nya Start	1,00	1,00	1,00	43
3	Nya Middle	1,00	1,00	1,00	42
4	Nya End	1,00	1,00	1,00	38
5	Ca Isolated	0,97	0,97	0,97	39
6	Ca Start	1,00	1,00	1,00	39
7	Ca Middle	1,00	1,00	1,00	39
8	Ca End	0,97	0,97	0,97	38
9	Nga Isolated	1,00	1,00	1,00	39
10	Nga Start	0,97	1,00	0,99	39
11	Nga Middle	0,86	0,84	0,85	37
12	Nga End	1,00	1,00	1,00	37
13	Pa Isolated	0,94	0,76	0,84	38
14	Pa Start	0,82	0,95	0,88	39
15	Pa Middle	0,76	0,82	0,79	39
16	Pa End	0,91	0,86	0,89	37
17	Ga Isolated	1,00	1,00	1,00	39
18	Ga Start	1,00	0,97	0,99	39
19	Ga Middle	1,00	0,97	0,99	39
20	Ga End	0,97	1,00	0,99	38
21	Va Isolated	0,95	1,00	0,99	39
22	Va End	0,95	1,00	0,97	39
Accuracy			0,96	852	
Macro Avg			96	96	852
Weighted Avg			96	96	852

demonstrated excellent performance with an accuracy of 96%. This indicates that the model can correctly classify around 96% of the data from the entire dataset used in this testing. It shows that the ResNet34 model has a high accuracy rate in recognizing additional Arabic characters for writing Jawi overall. Additionally, the values of precision, recall, and F1-score also demonstrate high consistency for each additional Arabic character class, with most classes having values close to 1.00. This indicates that the classification model has consistently and uniformly classified the additional Arabic character data for each class, regardless of the number of data points in each class.

E. Potential Utilization of Research Findings

This research holds significant potential in utilizing additional Arabic character datasets for writing Jawi and addressing the issue of the limited availability of complete Jawi character datasets online. The findings of this study open up valuable opportunities for advancing knowledge, especially in the fields of natural language processing and character recognition.

Firstly, the additional Arabic character dataset produced in this research can serve as a significant step toward the development of an Optical Character Recognition (OCR) system capable of recognizing Jawi characters. OCR technology holds immense importance in the modern digital world, enabling the conversion of text from image or print formats into editable and searchable digital text. However, recognizing Jawi characters through OCR poses challenges due to the scarcity of datasets encompassing sufficient variations and representations of Jawi characters. With the availability of this additional Arabic character dataset, machine learning models can be better trained to yield more accurate results in Jawi character recognition.

Furthermore, this supplementary Arabic character dataset can address the shortage of complete Jawi character datasets currently available online. The issue of inadequate datasets that cover variations and adequate quantities has hindered the development of machine-learning models for the Jawi language. With the introduction of this additional dataset, this research significantly contributes to enriching the resources available for the Jawi language. These resources can be employed by researchers and developers to create diverse applications and language-based technologies related to the Jawi language.

This study enables researchers and practitioners in the field of Jawi character recognition to harness this additional Arabic character dataset for training and testing new models and improving the performance of existing ones. By utilizing this dataset, research in natural language processing, handwriting recognition, and character recognition can be accelerated, paving the way for further innovations in various technology applications associated with the Jawi language.

Additionally, it's important to note that this research addresses the challenges of collecting Jawi character

datasets, which may be difficult to obtain widely. In this study, the additional Arabic character dataset is employed to complement the Jawi character dataset, offering an effective and practical alternative to overcome limitations in available online data.

Furthermore, this research offers a valuable contribution to language and cultural preservation. By developing reliable Jawi character recognition technology, Jawi-language manuscripts, including valuable and rare ancient texts, can be more easily digitized and preserved. This has positive implications for promoting the Jawi language and the cultural heritage of the communities that use it.

This study demonstrates the importance and scholarly relevance of addressing practical challenges in the fields of natural language processing and character recognition. Through the contributions offered by this research, the use of additional Arabic character datasets for writing Jawi holds great potential for advancing knowledge and technology related to the Jawi language. The findings of this study can serve as a starting point for further research and various applications that will create new opportunities for understanding and utilizing the rich language and cultural heritage of Jawi characters. As an innovative study that contributes to bridging scientific gaps, the outcomes of this research hold value in advancing knowledge about language and language technologies, while providing a positive impact on the communities that use the Jawi script.

V. CONCLUSION

This research aims to augment an additional Arabic character dataset used for writing Jawi. The results of dataset augmentation include the number of successfully generated additional Arabic characters for Jawi writing and the appearance of these additional Arabic characters in both connected and isolated forms. Furthermore, this study involves the creation of a dots dataset, which is a key element in some Jawi characters. This approach involves generating single-dot and three-dot datasets derived from existing Arabic characters. This additional dataset is a crucial step in the development of Jawi character recognition models.

In training the models for recognizing additional Arabic characters for Jawi writing, two well-known deep learning architectures, ResNet34 and InceptionV3, were used. Both of them were successfully trained to classify additional Arabic characters in Jawi writing. The model performance evaluation indicates high accuracy, with the InceptionV3 model achieving 95% accuracy and the ResNet34 model reaching 96%. These results demonstrate the effectiveness of both models in characterizing Jawi characters.

This research also provides a significant contribution, particularly in addressing the limitations of existing online Jawi character datasets. The findings hold great potential for the development of Optical Character Recognition (OCR) technology for Jawi characters. OCR technology is crucial for converting text from image or print formats

into editable and searchable digital text. Additionally, this additional dataset enables the development of more robust and diverse Jawi character recognition technology, opening up new possibilities for various technology applications related to the Jawi language. Beyond its technical benefits, this research has the potential to contribute to language and cultural preservation by facilitating the digitization and preservation of Jawi-language manuscripts, including valuable ancient texts. This has a positive impact on promoting the Jawi language and the cultural heritage of the communities that use it. Furthermore, this study can serve as a foundation for further research and various applications that will create new opportunities to understand and leverage the linguistic and cultural richness of Jawi characters. Overall, this research not only helps solve technical challenges but also has a profound positive impact on the preservation of the Jawi language and culture, as well as the communities that utilize it.

ACKNOWLEDGMENTS

All researchers involved in this study express their gratitude to the Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) of Universitas Syiah Kuala (USK) for funding this research activity through contract number: 412/UN11.2.1/PT.01.03/PNBP/2023.

REFERENCES

- [1] M. F. Nasrudin, K. Omar, M. S. Zakaria, and L. C. Yeun, "Handwritten cursive Jawi character recognition: a survey," in *Proc. International Conference on Computer Graphics, Imaging and Visualisation*, 2008, pp. 247–256.
- [2] S. Safrizal, F. Arnia, and R. Muharar, "Pengenalan aksara Jawi tulisan tangan menggunakan freeman chain code (FCC), support vector machine (SVM) dan aturan pengambilan keputusan," *Jurnal Nasional Teknik Elektro*, vol. 5, no. 1, pp. 45, 2016.
- [3] H. M. Balaha, H. A. Ali, M. Saraya, and M. Badawy, "A new arabic handwritten character recognition deep learning system (AHCR-DLS)," *Neural Comput Appl*, vol. 33, no. 11, pp. 6325–6367, 2021.
- [4] M. A. Khan, "Arabic handwritten alphabets, words and paragraphs per user (AHAWP) dataset," *Data Brief*, vol. 41, pp. 107947, 2022.
- [5] F. Mushtaq, M. M. Misgar, M. Kumar, and S. S. Khurana, "UrduDeepNet: offline handwritten Urdu character recognition using deep neural network," *Neural Comput Appl*, pp. 15229–15252, 2021.
- [6] M. M. Misgar, F. Mushtaq, S. S. Khurana, and M. Kumar, "Recognition of offline handwritten Urdu characters using RNN and LSTM models," *Multimed Tools Appl*, vol. 82, no. 2, pp. 2053–2076, 2023.
- [7] A. El-Sawy, M. Loey, and H. El-Bakry, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Transactions on Computer Research*, vol. 5, pp. 11–19, 2017.
- [8] N. Pratiwi, "Dataset of handwritten arabic characters with Harakat (Fathah, Kasrah, and Dhammah)," *Mendeley Data*, *V1*, Mendeley, 2022.
- [9] M. AbdElNafea and S. Heshmat, "Novel databases for arabic

- online handwriting recognition System,” in Proc International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), 2020, pp. 263–267.
- [10] S. Safrizal, “Pengenalan Karakter Jawi Tulisan Tangan Menggunakan Fitur Sudut,” *VOCATECH: Vocational Education and Technology Journal*, vol. 1, no. 1, pp. 1–4, 2019.
- [11] S. Razali, Fitri Arnia, Rusdha Muharrar, Kahlil Muchtar, and Akhyar Bintang, “Improved classification of handwritten jawi script based on main part of script body,” *Jurnal Rekayasa Sistem dan Teknologi Informasi*, vol. 7, no. 1, pp. 94–104, 2023.
- [12] K. Saddami, K. Munadi, Y. Away, and F. Arnia, “DHJ: A database of handwritten Jawi for recognition research,” in Proc. International Conference on Electrical Engineering and Informatics (ICELTICs), 2017, pp. 292–296.
- [13] S. Joseph and J. George, “Data augmentation for handwritten character recognition of MODI Script using deep learning method,” in *Information and Communication Technology for Intelligent Systems*, 2021, pp. 515–522.
- [14] N. Elaraby, S. Barakat, and A. Rezk, “A conditional GAN-based approach for enhancing transfer learning performance in few-shot HCR tasks,” *Sci Rep*, vol. 12, no. 1, pp. 16271, 2022.
- [15] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *J Big Data*, vol. 6, no. 1, 2019.
- [16] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, “Image data augmentation for deep learning: A survey,” *Journal of Big Data*, vol. 6, no. 60 pp. 1–8, 2022.
- [17] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” in Proc. Conference on Artificial Intelligence, 2017, pp. 13001–13008.
- [18] H. Inoue, “Data augmentation by pairing samples for images classification,” *Computer Vision and Pattern Recognition*, 2018.
- [19] C. Summers and M. J. Dinneen, “Improved mixed-example data augmentation,” in Proc. IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 1–9.
- [20] R. Takahashi, T. Matsubara, and K. Uehara, “Data augmentation using random image cropping and patching for deep CNNs,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2917–2931, 2020.
- [21] A. Stephen, A. Punitha, and A. Chandrasekar, “Designing self-attention-based ResNet architecture for rice leaf disease classification,” *Neural Comput Appl*, vol. 35, no. 9, pp. 6737–6751, 2023.
- [22] G. Meena, K. K. Mohbey, S. Kumar, R. K. Chawda, and S. V. Gaikwad, “Image-based sentiment analysis using inceptionV3 transfer learning approach,” *SN Comput Sci*, vol. 4, no. 3, pp. 242, 2023.
- [23] Institute of Electrical and Electronics Engineers, *2018 International Interdisciplinary PhD Workshop (IIPhDW) : 9-12 May 2018*.
- [24] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J Big Data*, vol. 8, no. 1, pp. 53, Dec. 2021.
- [25] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, “Fundamental Concepts of Convolutional Neural Network,” vol. 172, V. E. Balas, R. Kumar, and R. Srivastava, Eds., in *Intelligent Systems Reference Library*, vol. 172, Cham: Springer International Publishing, pp. 519–567, 2020.
- [26] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. International Conference on Learning Representations, 2015, pp. 1–14.
- [27] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of Tricks for Image Classification with Convolutional Neural Networks,” in Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 558–567.
- [28] L. R. Baltazar, M. G. Manzanillo, J. Gaudillo, E. D. Viray, M. Domingo, B. Tiangco, and J. Albia, “Artificial intelligence on COVID-19 pneumonia detection using chest xray images,” *PLoS One*, vol. 16, no. 10, pp. e0257884, 2021.
- [29] N. Otsu, “A Threshold selection method from gray-level histograms,” *IEEE Trans Syst Man Cybern*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
- [30] R. Andonie, “Hyperparameter optimization in learning systems,” *Journal of Membrane Computing*, vol. 1, no. 4, pp. 279–291, 2019.
- [31] S. R. Shah, S. Qadri, H. Bibi, S. M. W. Shah, M. I. Sharif, and F. Marinello, “Comparing inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A case study on early detection of a rice disease,” *Agronomy*, vol. 13, no. 6, pp. 1633, 2023.
- [32] R. Zhang, L. Du, Q. Xiao, and J. Liu, “Comparison of backbones for semantic segmentation network,” *J Phys Conf Ser*, vol. 1544, no. 1, pp. 012196, 2020.
- [33] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an Overview,” 2020.
- [34] M. Heydarian, T. E. Doyle, and R. Samavi, “MLCM: Multi-Label Confusion Matrix,” *IEEE Access*, vol. 10, pp. 19083–19095, 2022.