IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.17, No.3, July 2023, pp. 325~336
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.86302

Deep Learning Approaches for Nusantara Scripts Optical Character Recognition using Convolutional Neural Network, ConvMixer, and Visual Transformer

Agi Prasetiadi*[1], Julian Saputra[2], Iqsyahiro Kresna[3], Imada Ramadhanti[4]
[1,2,3,4]Faculty of Informatics, Institut Teknologi Telkom Purwokerto, Purwokerto
e-mail: ***[1]agi@ittelkom-pwt.ac.id**, [2]19102008@ittelkom-pwt.ac.id, [3]hirokresna@gmail.com, [4]19102003@ittelkom-pwt.ac.id

Abstrak

Jumlah penutur bahasa daerah yang mampu membaca dan menulis aksara tradisional di Indonesia semakin berkurang. Jika dibiarkan begitu saja, bisa jadi aksara-aksara Nusantara akan punah dan metode membacanya akan terlupakan di masa depan. Untuk mengantisipasi hal ini, penelitian ini bertujuan untuk melestarikan pengetahuan membaca aksara kuno dengan mengembangkan model Deep Learning yang mampu membaca gambar dokumen yang ditulis menggunakan salah satu dari 10 aksara Nusantara yang telah kami kumpulkan, yaitu Bali, Batak, Bugis, Jawa, Kawi, Kerinci, Lampung, Pallawa, Rejang, dan Sunda. Meskipun penelitian sebelumnya telah berusaha membaca aksara-aksara Nusantara menggunakan berbagai algoritma Machine Learning dan Convolutional Neural Network, namun mereka lebih fokus pada aksara tertentu dan kurang dalam pendekatan terintegrasi mulai dari pengenalan jenis aksara hingga pembacaan per aksara. Penelitian ini merupakan yang pertama yang secara komprehensif mencakup seluruh rentang aksara Nusantara, termasuk deteksi jenis aksara dan pembacaan aksara. Dalam penelitian ini, kami menggunakan model Convolutional Neural Network, ConvMixer, dan Visual Transformer, dan kemudian membandingkan kinerja masing-masing model tersebut. Hasil penelitian menunjukkan bahwa model kami mencapai akurasi 96% dalam mengklasifikasikan jenis aksara Nusantara, dan mencapai tingkat akurasi antara 93% hingga sekitar 100% dalam membaca karakter pada kesepuluh aksara Nusantara tersebut.

Kata kunci—*Aksara Nusantara, Optical Character Recognition, Convolutional Neural Network, ConvMixer, Visual Transformer*

Abstract

The number of speakers of regional languages who can read and write traditional scripts in Indonesia is decreasing. If left unaddressed, this will lead to the extinction of Nusantara scripts, and their reading methods may be forgotten in the future. This study aims to preserve the knowledge of reading ancient scripts by developing a Deep Learning model that can read document images written using one of the 10 Nusantara scripts we have collected: Bali, Batak, Bugis, Javanese, Kawi, Kerinci, Lampung, Pallava, Rejang, and Sundanese. While previous studies have tried to read traditional Nusantara scripts using various Machine Learning and Convolutional Neural Network algorithms, they have primarily focused on specific scripts and lacked an integrated approach from script type recognition to character recognition. This study is the first to comprehensively address the entire range of Nusantara scripts, encompassing script type detection and character recognition. Convolutional Neural Network, ConvMixer, and Visual Transformer models were utilized, and their respective performances were compared. The results indicate that our models achieved 96% accuracy in classifying Nusantara script types and achieved accuracy rates ranging from 93% to approximately 100% in reading characters across the ten Nusantara scripts.

Received Juny 29th,2023; Revised July 24th, 2023; Accepted July 30th, 2023

Keywords—*Nusantara Script, Optical Character Recognition, Convolutional Neural Network, ConvMixer, Visual Transformer*

1. INTRODUCTION

The Nusantara region, encompassing the present-day territory of Indonesia, is renowned for its vibrant cultural diversity, encompassing a vast array of languages and scripts [1] [2]. Indonesia is a treasure trove of linguistic heritage, boasting 719 regional languages, among which 707 are still actively utilized today [3]. Each distinct region within the Indonesian archipelago possesses unique linguistic characteristics, including using specific scripts to transcribe and preserve their local languages [4].

However, over time, these scripts have faced the risk of fading into obscurity as their speakers and users have become increasingly rare [5]. The initial script introduced in Indonesia was the Pallawa script, associated with the Sanskrit language. Over time, this has evolved into the Nusantara script, encompassing a range of distinct writing systems such as the Javanese script, Balinese script, Lontara or Bugis script, Rejang script, Lampung script, Batak script, Pallawa script, Kawi script, Sundanese script, and Kerinci script [6].

Optical Character Recognition (OCR) is an artificial intelligence technique for text recognition [7]. Its application has become widespread in Indonesia, exemplified by the Know Your Customer (KYC) policy implemented in financial institutions, which necessitates customers to submit their identification documents, including KTP or SIM, in digital form [8]. This technology offers benefits such as facilitating digitalization, streamlining operations, and enhancing efficiency. However, it is worth noting that specific scripts in Indonesia lack dedicated OCR models, posing a challenge for accurate recognition and interpretation.

The existing literature primarily focuses on the study of OCR for Javanese and Balinese scripts, with Javanese being the most extensively studied script, followed by Balinese script, as seen in Table 1. In a prior study on digitizing printed Javanese script, the approach involved text segmentation and character classification, which included feature extraction, eigenface calculation, nearest centroid classification, and reconstruction. However, the classification accuracy achieved was only 60.6% due to the inadequate discriminatory power of the selected binary features when dealing with over 600 character classes [9]. Similarly, in previous research on Balinese script, Convolutional Neural Network (CNN) and Support Vector Machine (SVM) were employed to transliterate the script into Latin characters, achieving accuracy of 86% and 82% [10]. Nevertheless, this approach had limitations regarding the number of character classes trained.

Notably, most Nusantara scripts follow the Abugida writing system, wherein various combinations of consonants and vowels result in unique glyphs. As possible combinations can quickly escalate into thousands of plausible glyphs, it becomes necessary to revert to recognizing the most fundamental form of individual characters before constructing the combined characters.

Table 1 Research data on Nusantara Scripts accessed online as of June 2023.

Script	IEEE	GS	Model Status	Script	IEEE	GS	Model Status
Javanese	15	859	v	Batak	2	62	v
Balinese	25	488	v	Pallawa	0	36	x
Lontara	1	95	v	Kawi	0	74	v
Rejang	0	12	x	Sunda	2	151	v
Lampung	4	101	v	Kerinci	0	13	v

In the "Model Status" column, "v" indicates that a model has been built, while "x" indicates that a model has not yet been built for that particular script. Please note that this table includes the number of research papers related to each script in IEEE and Google Scholar (GS).

In order to safeguard and preserve the Nusantara script from the threat of extinction, we developed an OCR model tailored explicitly for Nusantara scripts. The research utilized ten distinct script types, encompassing a comprehensive dataset of 582 script images. The script types used in this study are as follows: Batak, Rejang, Sunda, Javanese, Balinese, Kerinci, Pallava, Kawi, Lampung, and Bugis.

The creation of OCR for the Nusantara script involved three primary aspects: 1) the classification of script types into the ten categories mentioned earlier, 2) the identification of script letters within each script type, and 3) the accurate localization of script letters. This research primarily focused on the first two aspects: the classification of script types and the recognition of script letters. To accomplish this, we employed a CNN as the image recognition technique. Additionally, we explored the performance of ConvMixer and Visual Transformer on both the best and worst-performing script recognition models.

The paper's contribution is developing a Deep Learning model for OCR of Nusantara scripts, a pioneering effort to conduct a comprehensive investigation encompassing training on all Nusantara scripts. The model can classify script types into ten categories and recognize script letters within each script type. The paper explores the performance of CNN, ConvMixer, and Visual Transformer on script recognition models. The development of this model is significant as it can help preserve the knowledge of reading ancient Nusantara scripts and prevent them from becoming extinct in the future. The structure of this paper is organized into three main sections: the research methodology employed, the findings derived from each experimental phase, and, ultimately, the concluding remarks. All models are constructed utilizing the TensorFlow framework.

2. METHODS

2.1 Dataset Reconstruction

Most of the Nusantara scripts are of the Abugida type, an alphasyllabary script in which consonants and vowels are usually combined and considered a basic script [11]. The formation of additional vowels and consonants is usually written before, above, after, or below the basic script, forming a glyph or a new script formation.

This script behavior is known as complex text rendering in the computer world. Take the example of the Javanese letter, 'ha'. One letter will have at least five other variations if combined with swara diacritics [12]. Moreover, these variations will have at least four other variations if combined with Panyigeg diacritics. All of these variations still have four other variations if combined with Wyanjana. In total, 1 'ha' letter can have $1 \times 6 \times 5 \times 5 = 150$ variations. Therefore, if we include all basic letters in Javanese script, we can get thousands or tens of thousands of script variations. This is a large number dataset. Therefore, to handle the large number of variations encountered, we will use an approach per basic letter, not after the letter changes. In this way, we hope to obtain faster, more efficient, and more accurate letter-reading performance.

Some of our Nusantara script datasets are built based on Unicode. Unicode is an industry standard for computers to consistently display and process text and symbols from all writing systems in the world [13]. Unicode consists of characters, character encoding techniques and standards, code point mappings, descriptions of characteristics such as uppercase and lowercase letters, and reference data combinations. Nusantara scripts already included in Unicode include Bugis script, Balinese script, Sundanese script, Rejang script, Javanese script, Batak script, Makassar script, and Kawi script.

This research used ten types of script data with a total dataset of 582 script images, where Batak script had 51 script images, Rejang script had 37 script images, Sunda script had 55 script images, Javanese script had 85 script images, Balinese script had 102 script images, Kerinci script had 38 script images, Pallava script had 65 script images, Kawi script had 87 script images, Lampung script had 32 script images, and Bugis script had 30 script images.

The character image list is obtained from three primary sources: manual handwriting, online resources, and font-to-image extraction from Google Noto fonts. In preparation for our dataset for characters derived from manual handwriting and online resources, we first cropped each script to isolate individual characters, removing any surrounding blank spaces. Subsequently, all characters were meticulously organized and stored in separate folders, using formal names representing each character, such as "batak_letter_da" for the Batak script's "da," or "javanese_digit_one" for the Javanese script representing the number one, and so forth. This systematic organization ensures the dataset's clarity and accessibility.

After that, we do augmentation on our dataset. Data augmentation is modifying or manipulating original images into standard forms, which are then transformed in shape and position [14]. The purpose of augmentation is to improve the accuracy of the trained CNN model. This is because using augmentation methods increases the amount of data, which can improve the accuracy of the model [15]. In this study, the random resize crop method was used to augment the letter images of each character [16].

Each of these characters will be augmented 20 or 21 times, resulting in approximately 12,180 character variations. We will divide the data into 80% for training and 20% for validation. All annotations can be done automatically using a program for these ten types of scripts. However, the finalization of the class will be done using human judgment. A list of samples of scripts can be seen in Figure 1.

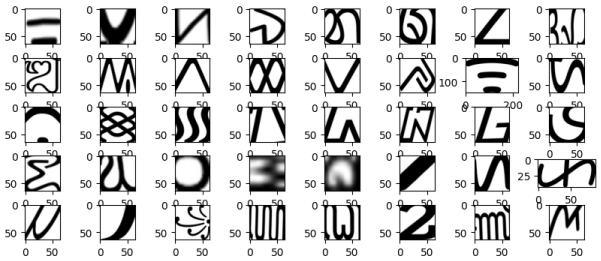

Fig. 1 List of samples of Nusantara Scripts used in this research.

2.2 Classification

Three deep learning approaches, namely CNN, ConvMixer, and Visual Transformer, were employed to construct models for image object classification. In the case of CNN, the architecture can be composed of various combinations of hidden layers, such as Conv2D, MaxPooling, Flatten, GlobalAveragePooling, Dense, and Dropout. There are no fixed rules governing the construction of the architecture, but different configurations of hidden layers can lead to variations in accuracy, performance, and model size [17] [18] [19] [20]. Similarly, the ConvMixer architecture breaks down the convolution process into two simpler processes: depthwise convolution and pointwise convolution based on input mixing [21]. In the case of a Visual Transformer, the input images are divided into patches and embedded with position information before being processed through multiple layers of the general transformer encoder. Consequently, this technique enables the utilization of transformers, typically used in NLP tasks, for image classification purposes [22] [23] [24].

2. 2.1 Classification of Character Types

The first problem in this research is to classify the types of letters into ten types of

scripts. We will use a CNN configuration to create a model that can classify script types like this, as shown in Figure 2. Here, we configure the number of filters in the convolution layer with (64, 16, 32, 64). First, a 64 × 64 input image will enter the convolution layer, followed by max pooling, until it enters the flattening layer and ends with a dense layer of 10 with softmax activation. The output of this last layer will contain the probabilities of each possible script type. The highest probability will be used as the benchmark for the script type. We will use the arg max function to search for the highest probability. For example, out of 1000 characters detected in a document, 980 are predicted as Javanese script, 19 as Sundanese script, and one as Bugis script. Then, we take the most common class as the language classifier in the document. Next, we will call the model to read letter by letter according to its language.

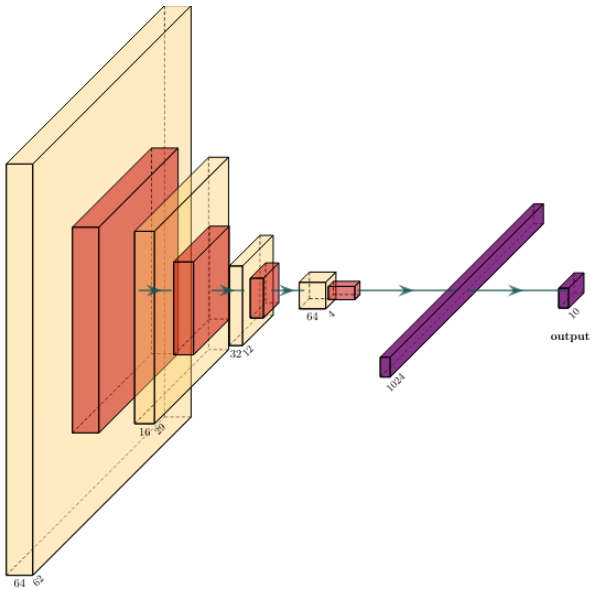

Fig. 2 The Architecture of CNN for Character Type Detection.

2. 2.2 Classification of Character Reading

The second challenge addressed in this study involves the development of ten CNN models designed for letter-by-letter text recognition. Specifically, three sequential architectures will be employed: CNN, ConvMixer, and Visual Transformer. CNN is chosen for the initial classification tasks due to its simplicity, widespread use, and well-known success in image classification. Its straightforward architecture facilitates easier implementation and faster model development, which is especially valuable when dealing with diverse Nusantara scripts. CNN's proven track record in recognizing objects within images and its extensive use in OCR applications provide a robust foundation for this research stage. By utilizing CNN as the starting point, we establish a reliable benchmark for evaluating the subsequent effectiveness of ConvMixer and Visual Transformer approaches.

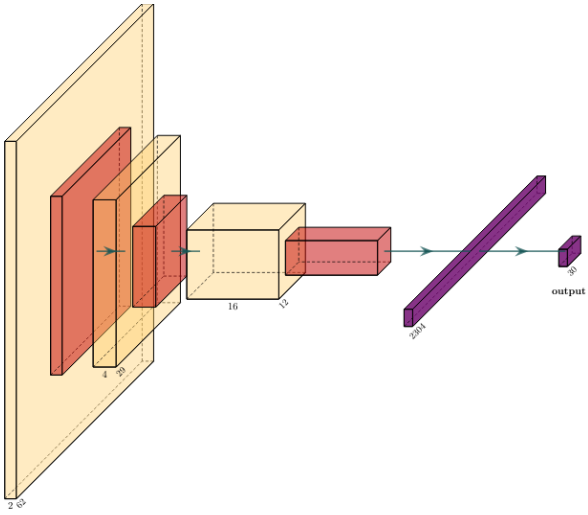

Fig. 3 The CNN Architecture in one of the Nusantara Characters Reader.

Note the architecture example in Figure 3 as a reference. If we use this architecture to read Bugis script, we need to do a slightly different configuration than the previous script type classifier architecture. Here, we use the number of filter configurations in the convolutional layer as follows (2, 4, 16). In addition, the output is not a dense layer of 10, but this time, it is 30 because the number of basic Bugis scripts we trained is 30. Since there are 10 Nusantara scripts that we want to read, there will be ten models to read all of these scripts.

Subsequently, one model exhibiting superior performance and another demonstrating inferior performance will be selected for further examination utilizing ConvMixer and Visual Transformer architectures. Regarding the ConvMixer-based model, the initial layer of the model accepts image inputs with dimensions of 32x32 pixels and three color channels. Subsequently, the images undergo processing through a convolutional stem layer, which performs convolution operations with a designated number of filters and a patch size of 2. The resulting outputs from this layer are subjected to an activation function, specifically the GELU activation, and subsequently normalized using batch normalization techniques to ensure stable and effective training.

The ConvMixer blocks serve as the fundamental component of the model. These blocks contain two key elements: a depthwise convolution layer and a pointwise convolution layer. The depthwise convolution layer performs spatial convolutions, capturing and analyzing local patterns and relationships within the input feature maps. Through an element-wise addition operation, the output of this layer is combined with the original input, establishing a residual connection that facilitates the flow of information. It is worth noting that while the ConvMixer block may resemble the Depthwise Separable Convolution layer, the pointwise convolution layer in the ConvMixer block acts as a residual hub between the input and the depthwise-convoluted input, as illustrated in Figure 4.

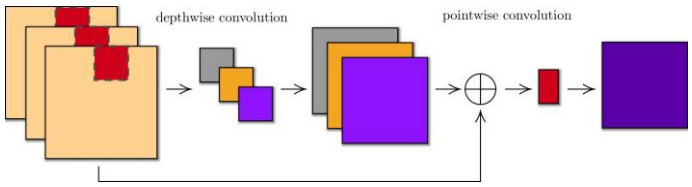

Fig. 4 ConvMixer's Basic Architecture Block.

The number of ConvMixer blocks incorporated into the model is determined by the depth parameter, which has been configured to have a value of 8 in this particular setup. However, this value can be modified to adjust the complexity of the model and meet specific

requirements of the task at hand. Following the ConvMixer blocks, a global average pooling layer is employed to aggregate spatial information and reduce the dimensionality of the feature maps. Subsequently, a dense layer with softmax activation is appended to perform the classification task, producing predicted probabilities for each class. During the training process, the model is optimized using a specified learning rate, weight decay, batch size, and number of epochs. The learning rate determines the magnitude of the parameter updates during optimization, while weight decay regulates the amount of regularization applied to the model weights.

Finally, we will test the selected scripts further using Visual Transformer, as shown in Figure 5. Visual Transformer operates on the principles of the Transformer architecture, where the input image transforms patches. These patches are subsequently subjected to linear projection, incorporating position embedding. The processed patches are then fed into a sequence of eight stacked transformer encoding layers, followed by Multi-Layer Perceptron (MLP) layers for classification purposes.

The Visual Transformer model is configured with specific parameters to optimize its performance in script recognition tasks. The model incorporates a learning rate of 0.001 and a weight decay 0.0001 to regulate the training process and prevent overfitting, with a batch size of 2 and 100 epochs. The input images are resized to an image size of 32 pixels. The images are then divided into patches of size 4, allowing the model to extract detailed information from different sections of the input images. The number of patches is determined based on the division of the image size by the patch size.

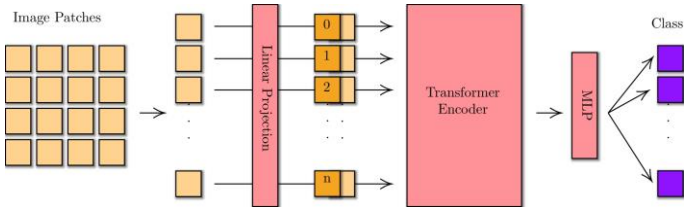

Fig. 5 Visual Transformer Architecture.

The model utilizes a projection dimension of 32 and employs four attention heads to facilitate the attention mechanism, enabling the model to focus on different aspects and features within the script images simultaneously. These transformer layers play a crucial role in capturing and encoding the spatial relationships and dependencies within the input images. Eight transformer layers are employed to further enhance the model's capacity for representation, allowing for a deeper understanding and analysis of the script images. These layers build upon each other, progressively refining the learned representations. The final classification is performed through multi-layer perceptron (MLP) heads with dense layers, providing the capacity to map the learned features to the corresponding script categories.

The CNN model is trained using the Adam optimizer. On the other hand, the ConvMixer and Transformer models are trained using the Adam with Weight Decay optimizer. The chosen loss function for all models is sparse categorical crossentropy, defined as $loss = -\Sigma\, y_i \log y'_i$, where y represents the ground truth and y' represents the predicted output. The smaller the loss, the better the model learns.

3. RESULTS AND DISCUSSION

3.1 Script Type Recognition Model

This section presents the performance evaluation results of our Convolutional Neural Network (CNN)-based script type recognition model. The model was developed using a dataset of 10 different scripts, and each script's characters were augmented to create approximately 20

variations. The training dataset comprised 12,222 character images, representing 582 unique characters. The model underwent training for a total of 20 epochs, with each epoch divided into two batches. To assess the model's performance and generalization ability, we conducted an evaluation using a 20% validation split.

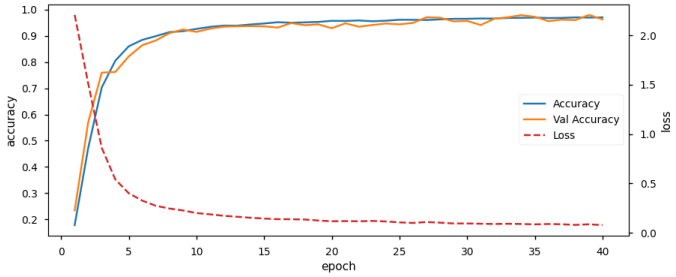

Fig. 6 Performance of Script Type Recognition Model.

As depicted in Figure 6, the results indicate a positive trend in both accuracy and validation accuracy. We observed that both metrics exhibited consistent growth as the model underwent training, indicating effective learning and adaptation. The model's best performance, considering the proximity of accuracy and validation accuracy, was achieved when accuracy reached approximately 96.58% and validation accuracy reached around 96.48%. This indicates a high level of consistency between the model's performance on the training data and its generalization ability on unseen validation data.

Furthermore, the loss function demonstrated a consistent and steady decrease, gradually approaching a value close to zero. This indicates that the model successfully learned the underlying patterns and features of the script types during training, resulting in improved accuracy. Overall, the model showcased considerable accuracy in script type recognition. To further enhance its performance in sentence recognition, a simple statistical analysis can be employed to identify the most frequently occurring script type.

3.2 Comparative Analysis of the 10 Individual Models

In this experimental setup, individualized models were trained for each script using a standardized architecture consisting of three convolutional neural network (CNN) layers. As depicted in Figure 7, the results exhibit promising character classification outcomes across all models. Most models achieved a reading accuracy of over 95%, except for the model designed explicitly for the Lampung script.

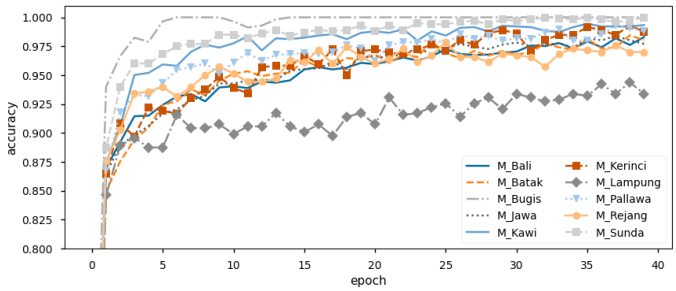

Fig. 7 Accuracy of Script Recognition Models: Script-Specific Performance.

Figure 8 illustrates the logarithmic-scale loss models for the reading performance of each script type. The loss values exhibit a decreasing trend over time, with the Bugis script model experiencing the most significant loss, followed by the Sundanese script model. The models for Bugis and Sundanese scripts demonstrate the highest accuracy, while those for Lampung and Rejang scripts achieve relatively lower accuracies.

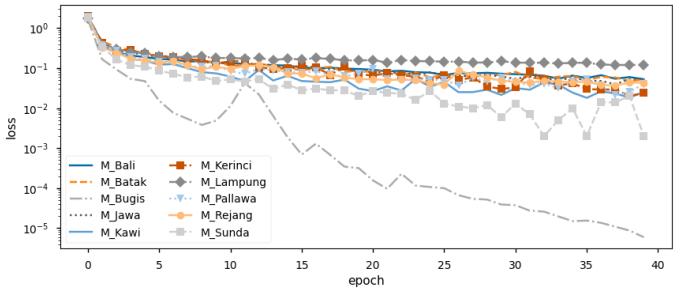

Fig. 8 Loss of Script Recognition Models: Script-Specific Performance (Log Scale).

The loss bias performance of all ten models is presented in Figure 9. Loss bias refers to the absolute difference between the loss and bias loss values. It serves as an indicator for assessing overfitting or underfitting in the models. A bias loss approaching zero suggests a well-balanced learning and testing proportion in the models. The results reveal that most models exhibit good performance without significant signs of overfitting. Notably, the model for the Kerinci script initially experienced overfitting around the 15th epoch but subsequently recovered and returned to a normal state.

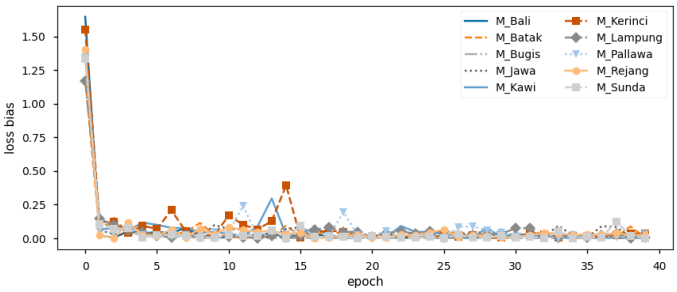

Fig. 9 Loss Bias of Script Recognition Models: Script-Specific Performance.

In the upcoming experiment, we aim to expand our investigation by extending two models: one associated with high performance in recognizing Sundanese script and the other associated with low performance in recognizing Lampung script. The objective is to train these models further using ConvMixer and Visual Transformer architectures, thereby obtaining additional insights into the accuracy achieved when employing different models for script recognition.

3.3 Evaluation of alternative models

In this stage of our study, we conducted a performance comparison between the Lampung and Sundanese script recognition models utilizing the extended algorithms. Specifically, we evaluated the effectiveness of two advanced architectures, ConvMixer with 8 depths and Visual Transformer with 8 transformer layers. As shown in Figure 10, the results reveal that a simpler model consisting of 3 CNN layers significantly outperforms the other models in accuracy. Surprisingly, the Visual Transformer model exhibits slower convergence and achieves the lowest accuracy among the evaluated models. The best ConvMixer-based model achieves comparable performance to the lowest-performing 3-layer CNN model.

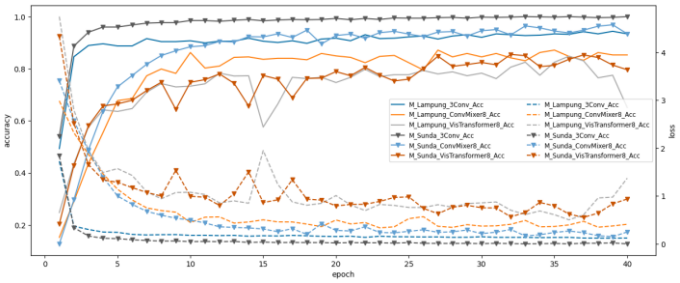

Fig. 10 Performance Comparison of Lampung and Sunda Script Recognition Models with Extended Algorithm, 3-layer CNN, ConvMixer (8 Depths), and Visual Transformer (8 Transformer Layers).

4. CONCLUSION

This study involves the development of multiple deep learning models to address two tasks: 1) classifying 10 Nusantara scripts and 2) recognizing characters within each Nusantara script. The model utilizing a 4-layer CNN architecture achieved an accuracy of approximately 96% in script classification. Regarding character recognition, the model based on a 3-layer CNN architecture displayed remarkable performance, achieving near 100% accuracy for Bugis and Sunda scripts, making them the most accurate models for character reading. Conversely, the lowest-performing reader models were observed for the Lampung and the Rejang scripts. Intriguingly, models constructed using ConvMixer and Visual Transformer architectures demonstrated lower reading performance than conventional CNN architecture, with ConvMixer exhibiting superior performance over Visual Transformer. Given that these models were primarily trained on computer-generated characters, further investigations are required to train them on handwritten characters to enhance their generalization capabilities.

REFERENCES

[1] S. Dardjowidjojo, ``Strategies for a successful national language policy: The Indonesian case," *Int. J. Sociol. Lang.*, vol. 130, no. 130, pp. 35-47, 1998, doi: 10.1515/ijsl.1998.130.35.

[2] D. A. Maharani, M. F. Pratama, and D. Setiawati, ``Cuneiform (Huruf Paku) Sebagai Pelopor Lahirnya Aksara Di Nusantara," *Jurnal Sejarah, Budaya dan Pengajarannya*, vol. 1, no. 2, pp. 1-11, 2022.

[3] M. I. Romadhan, ``Festival Sebagai Media Komunikasi Dalam Membangun Citra Destinasi Wisata Budaya Di Sumenep," *Destinesia: Jurnal Hospitaliti dan Pariwisata*, vol. 1, no. 1, pp. 1-10, 2019. doi: 10.31334/jd.v1i1.549.

[4] A. Mawadda Warohma, P. Kurniasari, S. Dwijayanti, Irmawan and B. Yudho Suprapto, ``Identification of Regional Dialects Using Mel Frequency Cepstral Coefficients (MFCCs) and Neural Network," in *2018 International Seminar on Application for Technology of Information and Communication, Semarang, Indonesia*, 2018, pp. 522-527, doi: 10.1109/ISEMANTIC.2018.8549731.

[5] J. Lo Bianco, ``The importance of language policies and multilingualism for cultural diversity," *Int. Soc. Sci. J.*, vol. 61, no. 199, pp. 37-67, 2010, doi: 10.1111/j.1468-2451.2010.01747.x.

[6] E. Alfian, ``Penggunaan Unsur Aksara Nusantara Pada Huruf Modern," *Jurnal Komunikasi Visual*, vol. 7, no. 1, pp. 42-48, 2014.

[7] P. K. Charles, V. Harish, M. Swathi, and C. H. Deepthi, ``A review on the various techniques used for optical character recognition," *International Journal of Engineering Research and Applications*, vol. 2, no. 1, pp. 659-662, 2012.

[8] A. Ghazi, ``The urgency of electronic Know Your Customer (e-KYC): How electronic customer identification works to prevent money laundering in the fintech industry," *Diponegoro Law Review*, vol. 7, no. 1, pp. 34-52, 2018.

[9] A. W. Mahastama and L. D. Krisnawati, ``Optical character recognition for printed javanese script using projection profile segmentation and nearest centroid classifier," in *2020 Asia Conference on Computers and Communications, ACCC 2020, Institute of Electrical and Electronics Engineers Inc.*, 2020, pp. 52-56. doi: 10.1109/ACCC51160.2020.9347895.

[10] A. R. Widiarti, ``Penelitian Pendahuluan Transliterasi Citra Aksara Bali Menggunakan Ciri Momen Invarian dan Algoritma Klasifikasi SVM atau CNN," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 10, no. 1, pp. 580-589, 2023.

[11] R. Maulana, ``Aksara-aksara di Nusantara: Seri Ensiklopedia," Samudra Biru, 2020.

[12] G. K. Javaholic, ``Gaul Aksara Jawa," LKIS Pelangi Aksara, 2015.

[13] S. R. Saragih and P. Utomo, ``Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks," in *Proceedings of the 2020 4th International Conference on Computer Science and Computational Intelligence (ICCCI)*, 2020, pp. 1-4, doi: 10.30865/komik.v4i1.2691.

[14] N. Ibrahim, G. A. Lestary, F. S. Hanafi, K. Saleh, N. K. C. Pratiwi, M. S. Haq, and A. I. Mastur, ``Klasifikasi Tingkat Kematangan Pucuk Daun Teh menggunakan Metode Convolutional Neural Network," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 10, no. 1, p. 162, 2022.

[15] K. H. Mahmud and S. Al Faraby, ``Klasifikasi Citra Multi-Kelas Menggunakan Convolutional Neural Network," in *Proceedings of the International Conference on Engineering and Optimization (ICEO)*, 2019, pp. 2127-2136. doi: 10.34818/eoe.v6i1.8501.

[16] T. M. Hafiez, D. Iskandar, A. W. SK, and R. F. Boangmanalu, ``Optimasi Klasifikasi Gambar Varietas Jenis Tomat dengan Data Augmentation dan Convolutional Neural Network," *Smart Comp: Jurnalnya Orang Pintar Komputer*, vol. 11, no. 2, pp. 175-186, 2022.

[17] G. F. Fitriana, A. B. Arifa, A. Prasetiadi, F. D. Adhinata, and N. G. Ramadhan, "Improving Accuracy of Cloud Images Using DenseNet-VGG19," *International Journal on Advanced Science, Engineering & Information Technology*, vol. 13, no. 2, 2023.

[18] F. Siddique, S. Sakib, and M. A. B. Siddique, ``Recognition of handwritten digit using convolutional neural network in Python with TensorFlow and comparison of performance for various hidden layers," in *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, pp. 541-546, September 2019.

[19] C. Garbin, X. Zhu, and O. Marques, ``Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, pp. 12777-12815, 2020.

[20] W. A. Haque, S. Arefin, A. S. M. Shihavuddin, and M. A. Hasan, ``DeepThin: A novel lightweight CNN architecture for traffic sign recognition without GPU requirements," *Expert Systems with Applications*, vol. 168, p. 114481, 2021.

[21] A. Trockman and J. Z. Kolter, ``Patches are all you need?," in *arXiv preprint*, arXiv:2201.09792, 2022.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Schmidhuber, and N. Houlsby, ``An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint*, arXiv:2010.11929, 2020.

[23] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, ``A Survey on Vision Transformer," arXiv:2012.12556, 2020.

[24] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan, and Z. He, ``A Survey of Visual Transformers," arXiv:2111.06091, 2021.