JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 10 Number 2 | December 2025: 308-322
DOI: 10.15575/join.v10i2.1606

A Comparison Analysis Between ResNET50 and XCeption for Handwritten Hangeul Character using Transfer Learning

Dede Kurniadi[1], Nabila Putri Nurhaliza[2], Benedicto B. Balilo Jr[3], Hilmi Aulawi[4], Asri Mulyani[5]
[1,2,5]Department of Computer Science, Institut Teknologi Garut, Indonesia
[3]CS/IT Department, College of Science, Bicol University, Philippines
[4]Department of Industrial Engineering, Institut Teknologi Garut, Indonesia

Article Info

Article history:

Received April 14, 2025
Revised June 08, 2025
Accepted July 16, 2025
Published August 17, 2025

Keywords:

Adam Optimizer
Grad-CAM
Hangeul Character
K-Fold
ResNet50
Transfer Learning
Xception

ABSTRACT

The enthusiasm for Korean pop culture in Indonesia has contributed to a growing interest in learning the Korean language, including its writing system, Hangeul, which currently ranks as the 6th most studied language. Hangeul has a unique structure, where each character is arranged in syllabic blocks of consonants and vowel combinations. The main challenge in Korean character classification lies in the similarity between characters and the complex structure, making it more difficult for models to recognize. This study aims to compare two deep convolutional neural networks are ResNet50 and Xception, using transfer learning for handwritten Hangeul character classification. While previous studies have examined CNN-based character recognition, this study highlights the effectiveness of deeper architectures with limited yet augmented data. Unlike earlier works, it incorporates Grad-CAM visualizations, transfer learning with partial fine-tuning, and multiple train-test ratios to analyze model behavior. A total of 1,920 images across 24 classes were evaluated using 5-fold cross-validation, with extensive augmentation and preprocessing to simulate variation. The Machine Learning Life Cycle (MLLC) framework assessed model performance through accuracy, precision, recall, F1-score, and AUC. Both models achieved high performance, with ResNet50 consistently outperforming Xception in most folds, especially in precision and F1-score. ResNet50 achieved perfect scores (100%) across all metrics, while Xception also performed strongly with up to 99.74% accuracy. These results indicate that ResNet50 is more effective in classifying Korean letters on the dataset used in this study. For future research, a robustness evaluation can be applied using data that was not included in previous training or testing.

Corresponding Author:

Dede Kurniadi
Department of Computer Science, Institut Teknologi Garut
Jl. Mayor Syamsu No. 1, Garut, Indonesia. 44151
Email: dede.kurniadi@itg.ac.id

1. INTRODUCTION

The increasing popularity of Korean culture, including K-pop music and Korean dramas, has attracted many people worldwide, including in Indonesia, to learn the Korean language. According to a survey conducted by the Korean Foundation for International Cultural Exchange in 2022, 83.6% of Hallyu consumers in Indonesia expressed an overall positive perception of Korea [1]. With the high enthusiasm for Korean culture, it is not surprising that many Indonesians are eager to learn the Korean language [2]. A report by Duolingo in 2023 revealed that Korean ranked as the sixth most popular language on the platform [3]. The Korean language is considered relatively difficult to learn because it uses two writing systems: Hanja and Hangeul [4]. Hanja consists of characters adapted from the original Chinese Hanzi script Hangeul is the Korean alphabet, with 24 basic characters. It has a unique structure where characters are arranged in syllabic blocks, combining consonants and vowels [5].

Therefore, image classification can be applied as a learning aid to facilitate the understanding and use of the Korean language. Image classification can be performed using various classical deep learning algorithms, such as Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Ensemble Learning, Convolutional Neural Network (CNN), and Multi-Layer Perceptron (MLP) [6]. CNN is an evolutionary algorithm of MLP, specifically designed to recognize patterns from two-dimensional data, such as images and videos [7]. Previous studies have shown that CNN can be used for various character recognition tasks, including Japanese [8] and Arabic characters [9].

Various studies have been conducted on classification, including the classification of Korean characters. Previous research conducted a study on simple Optical Character Recognition (OCR) to recognize Hangeul characters using Artificial Neural Network (ANN), showing the highest accuracy reaching 97% [10], though performance remained limited when faced with handwriting variations. Another study using a backpropagation neural network reached 80.83% accuracy [11], while image processing techniques such as Canny edge detection also improved results, yielding 99.1% accuracy from a small dataset of 120 images [12]. In addition, other researchers who conducted modeling for Korean letter classification with the VGG-16 architecture produced an accuracy of 99.52% [13]. However, these studies often lacked architectural diversity and did not compare deeper or more recent CNN variants. A study comparing various ANN models found that CNN outperformed MLP and RNN for Korean character recognition, with CNN reaching 96.5% accuracy [14]. This shows that image processing techniques can play a role in improving model performance, so it is possible that other techniques, such as data augmentation, can help improve character recognition accuracy by increasing data variation [12].

Although studies like [15] have explored CNN architectures such as Inception V2, VGG-16, and Xception for unrelated tasks like road crack detection with accuracies exceeding 99%. This study addresses that gap by evaluating both architectures, chosen based on their proven performance in diverse image classification tasks [14] [13] architectures such as ResNet50 and Xception were chosen for application in Hangeul character classification, considering that both showed good performance in image classification tasks. While lightweight models like MobileNetV2 and EfficientNet are optimized for deployment on edge devices, they may underperform in complex classification scenarios. Deeper models like ResNet50 and Xception incorporate advanced techniques, such as residual connections [16] and depthwise separable convolutions [17], that enhance feature learning in noisy or highly variable data conditions (e.g., handwriting). For instance, ResNet50V2 and Xception achieved classification accuracies of 98.88% and 99.17%, respectively, significantly outperforming MobileNetV2 at 93.71%[18] Similarly, Xception also demonstrated higher accuracy than MobileNetV2 in a litter classification system [19]. Unlike prior research that relied on small architectures or non-visual explanations, our approach fine-tunes deeper models using a balanced dataset of 1,920 images.

Although the dataset size is relatively modest for deep CNNs, this work aims to explore how modern architectures perform under constrained data conditions, a realistic challenge in low-resource language processing. To support a more reliable performance assessment under such limitations, this study applies k-fold cross-validation instead of relying on a single train-test split. This method enables the model to be trained and validated on different partitions of the dataset, providing more consistent evaluation results. As supported by other papers, k-fold cross-validation is widely used to reduce evaluation bias and improve robustness, especially when working with limited data [20], [21]. Furthermore, visual explanation through Grad-CAM enables understanding of model decisions beyond numerical metrics.

This study will build and compare CNN-based Korean Hangeul character classification models using the ResNet50 and Xception architectures. ResNet50 consists of 50 convolutional layers with residual blocks, enabling deeper network training without losing gradient information [16]. Meanwhile, Xception uses a channel-based convolutional splitting technique to improve feature extraction efficiency without significantly increasing the number of parameters [17]. Both architectures are supported by the Adam optimizer, which is able to adaptively adjust the learning rate based on the first and second moment estimates, thereby accelerating convergence and improving training stability [22]. While previous works have used CNNs such as VGG-16 [13] or simple ANN-based OCR systems [10], few have compared deeper CNN models on handwritten Hangeul. This study contributes to the field by performing a focused evaluation of two advanced CNN architectures are ResNet50 and Xception, leveraging transfer learning and Grad-CAM visualizations for interpretability. With this approach, each architecture is expected to optimize Hangeul Korean characters. The next section of this paper is organized as follows: Section 2 presents the methodology based on the Machine Learning Life Cycle (MLLC) framework. Section 3 discusses experimental results and analysis. Section 4 concludes the study and outlines future directions.

2. METHODOLOGY

The methodology used in this study is the Machine Learning Life Cycle (MLLC), which includes four stages, namely data collection, data pre-processing, CNN model design, and performance evaluation. In character recognition tasks, conventional techniques such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and basic Multi-Layer Perceptron (MLP) often struggle with spatial invariance and require manual feature extraction [23], making them less effective for complex scripts like Hangeul. Classical CNNs such as LeNet or VGG-16 offer improvements, but are either too shallow or parameter-heavy [24] without incorporating depth optimization or computational efficiency. To address these limitations, this study compares two modern CNN architectures are ResNet50 and Xception, that are designed to solve degradation and efficiency issues through residual connections and depthwise separable convolutions, respectively. Both models are implemented with transfer learning using pre-trained ImageNet weights [25]. We explore partial fine-tuning strategies where the earlier convolutional layers are frozen to retain general visual features, while the final layers are retrained to adapt to the handwritten Hangeul domain. This balances computational efficiency and domain-specific learning. MLLC was chosen because, in its use, it provides general and systematic guidance for building efficient machine learning projects [26] so that it can be applied in various model training, including CNN, the framework for this research is shown in Figure 1.

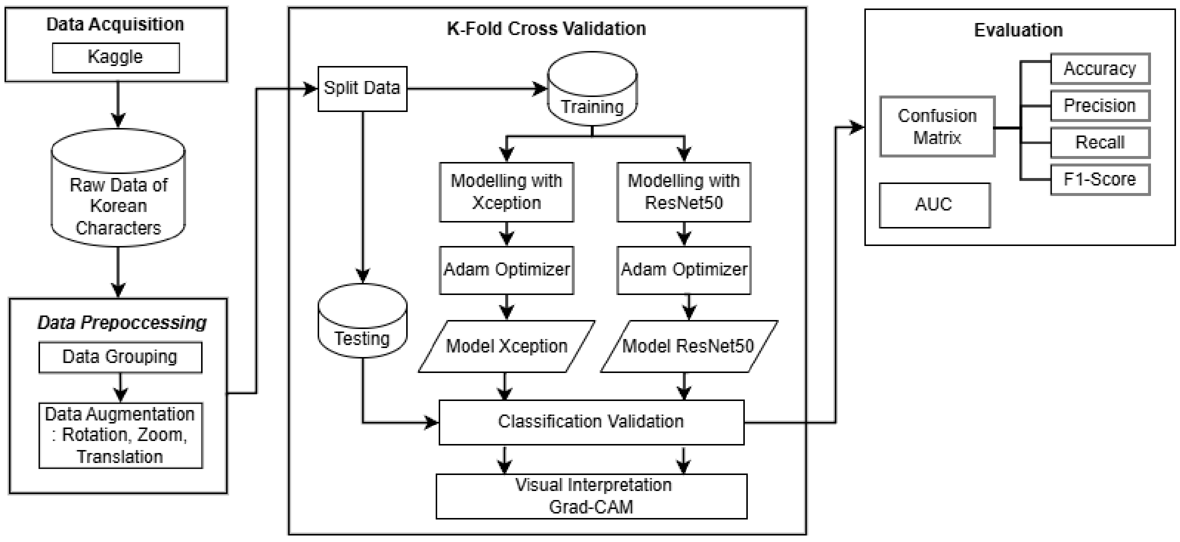


Figure 1. Research Workflow Framework

As illustrated in Figure 1, this research begins with the acquisition of a handwritten Korean character dataset from Kaggle [27]. The quantity and quality of the dataset are crucial factors influencing model performance. Therefore, the collected data is ensured to have sufficient resolution and include handwriting variations from different individuals to improve the model's generalization ability in recognizing diverse writing styles.

After the acquisition, the preprocessing stage includes grouping the data by class to maintain a balanced class distribution. To further enhance generalization, data augmentation techniques such as rotation, scaling (magnification), and translation are applied. These augmentations increase the diversity of training samples and help the model better recognize varied handwriting patterns.

The processed data is then used in a 5-fold cross-validation scheme, where the dataset is divided into five equal parts. In each fold, four parts are used for training and one for testing, ensuring each sample is involved in both training and evaluation across the folds. This technique helps reduce overfitting and yields a more robust estimate of the model's performance.

Two convolutional neural network (CNN) architectures—Xception and ResNet50—are trained independently in each fold using the Adam optimizer. Training is conducted from scratch in every fold to allow each model to learn robust and fold-independent feature representations of Korean characters. This modeling approach enables a fair and consistent performance comparison between architectures.

After training, model evaluation is conducted using several classification metrics, including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC) [28]. These metrics are used to assess not only the predictive performance of the models but also their reliability and applicability in real-world scenarios.

To further improve model transparency and interpretability, Gradient-weighted Class Activation Mapping (Grad-CAM) is applied to visualize the spatial regions of the input images that most influenced the model's decisions. Grad-CAM generates heatmaps that highlight key areas in the input that contributed to classification outputs [29]. This visual explanation aids in understanding the decision-making process of the CNN models, providing insight into the learned features and ensuring the model behaves as expected.

3. RESULT AND DISCUSSION

In this study, a total of 1,920 images of Korean characters were collected for training and testing the model. The MLLC methodology applied in this study focuses on four main stages. Model evaluation is carried out using various metrics, such as accuracy, precision, recall, and F1-score, to ensure that the model not only has high accuracy but is also able to generalize data well in Korean character classification. The following sections explain each stage in detail.

3.1. Result

This section presents the results obtained from each stage of the classification process using the ResNet50 and Xception architectures. Despite the limited dataset size of 1,920 images, both models achieved strong and consistent performance across all cross-validation folds. Data augmentation techniques were applied to introduce greater variation in handwriting styles and enhance generalization. Deep CNN architectures typically require larger datasets to avoid overfitting. The results range from data collection to model performance across various metrics and comparisons between the implemented models. These results are important for evaluating how well the models perform in recognizing Korean characters and identifying which architecture provides the most reliable classification. The explanation begins with the data acquisition process.

3.1.1 Data Acquisition

Hangeul is a Korean script system created by King Sejong in 1443 and officially introduced through the Hunminjeongeum document in 1446. This script is designed to accurately represent the sounds of the Korean language, with a systematic structure that facilitates the learning process and character recognition [30]. In the context of this research, each image in the dataset represents one Hangeul character in various handwriting styles. This is so that the model is able to recognize letterforms more flexibly. Before being used in the training process, the dataset was analyzed to ensure its quality and distribution. The first step in data acquisition is downloading the dataset from Kaggle. After the dataset was downloaded, the data was separated based on each class according to the characters represented, namely 14 consonants and 10 vowels, to ensure that each class had a clear representation. The dataset used in this study consists of 1,920 images of Korean characters classified into a total of 24 classes. These classes are illustrated in Table 1, which displays samples of the Korean characters used in this study, consisting of 14 consonants and 10 vowels.

Table 1. List of Korean Character Classes Used in The Dataset

Consonants	Name (C)	Vowels	Name(V)
ㄱ	Giyeok	ㅏ	A
ㄴ	Nieun	ㅑ	Ya
ㄷ	Digeut	ㅓ	Eo
ㄹ	Rieul	ㅒ	Yae
ㅁ	Mieum	ㅗ	O
ㅂ	Bieup	ㅛ	Yo
ㅅ	Siot	ㅜ	U
ㅇ	Ieung	ㅠ	Yu
ㅈ	Jieut	ㅡ	Eu
ㅊ	Chieut	ㅣ	I
ㅋ	Kieuk		
ㅌ	Tieut		
ㅍ	Pieup		
ㅎ	Hieut		

Table 1 includes all character classes considered in this research, ensuring a balanced representation of both consonants and vowels. For example, [ㄱ] (Giyeok) represents the k/g sound produced when the back of the tongue touches the palate. The letters [ㄴ] (Nieun) represent the n sound, [ㄷ] (Digeut) represents the d/t sound. These letter names generally indicate the initial sound they represent, such as g in Giyeok, d in Digeut, or b in Bieup, and so on [31]. The image resolution is examined to ensure it remains sufficient for the classification process, preserving the visual details of each character so they can be effectively utilized by the model. The image resolution was checked to remain adequate for the classification process so that the visual information on each character remains clear and can be optimally used by the model. Diversity in the dataset is an important factor in improving the generalization ability of the model. Therefore, the collected images include various writing styles with variations in line thickness and character proportions. Thus, the model is expected to be able to recognize Korean characters more accurately, even when faced with different writing variations.

3.1.2. Data Preprocessing

Preprocessing is an important stage in the machine learning process that aims to prepare the raw data to be used by the model [32]. It includes processes such as data cleaning, data augmentation, and data division into training and testing subsets [33]. In the context of Hangeul character classification, it is also important to ensure that the distribution of data within each class remains balanced so that the model is not skewed towards a particular class. In addition, splitting the dataset into training and testing data is done to evaluate the generalization ability of the model to data that has not been seen before [34]. After the dataset was downloaded, a selection of the classes to be used in this study was made to suit the needs of the experiment. The dataset consisted of 24 classes with a relatively balanced number of samples in each class, which helps prevent classification bias.

In addition to dataset division, data augmentation techniques are applied to increase the diversity of the training dataset without having to manually increase the number of images. The augmentation techniques used include rotation of up to 10 degrees, horizontal and vertical shifts of 10%, and zooming of up to 10%. With this technique, the model can learn to recognize characters in various conditions, positions, and orientations. To ensure that the model can read pixel values more efficiently, each image is also normalized by changing its pixel values into the range of 0 to 1. To clarify how augmentation changes the appearance of the original image, Figure 2 shows some examples of comparisons between images before and after augmentation.

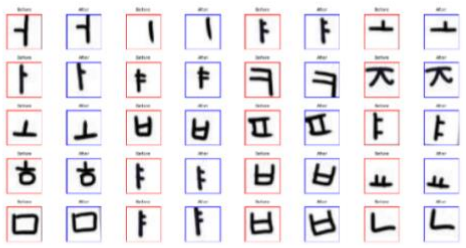


Figure 2. Example of results before and after augmentation

Figure 2 shows the original image marked with a red box (Before), while the augmentation results are given a blue box (After). This shows that augmentation can produce image variations that are different from the original image. With this variation, the model will be better able to recognize characters even in various position conditions or orientations that are different from the training data. This augmentation technique also helps prevent overfitting by ensuring that the model does not just memorize certain patterns but also learns to recognize characters based on more general features. Technically, the augmentation is applied only to the training set using the ImageDataGenerator function. Meanwhile, the validation set only uses pixel rescaling without augmentation to ensure an objective evaluation of the model's generalization ability.

3.1.3 Modelling

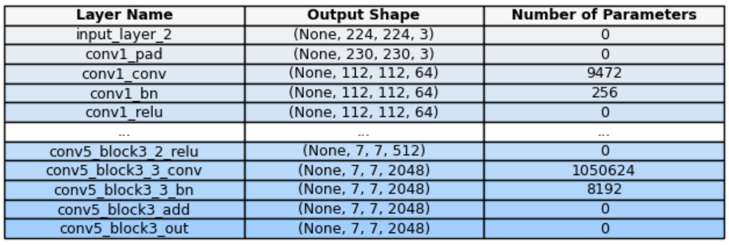
The modeling stage consists of three main processes, namely building a model, determining the optimizer, and training. The models used are ResNet50 and Xception, each of which is adjusted to the dataset. For optimization, Adam is used so that the learning process is more stable and efficient. The model is trained using 5-fold cross-validation, where in each fold, the data is split into training and testing subsets and trained for 20 epochs. During training, evaluation metrics such as accuracy, precision, recall, F1-score, and AUC are monitored to ensure optimal performance while minimizing the risk of overfitting.

During the modeling phase, both ResNet50 and Xception architectures are initialized with pre-trained weights from ImageNet. Rather than retraining all layers, we adopt a partial fine-tuning strategy. Specifically, the base convolutional layers are frozen, while custom classification layers comprising global average pooling, dense layers, batch normalization, and dropout are added and trained on the Hangeul dataset. This approach reduces overfitting risk and leverages pre-learned visual representations. An ablation study on different fine-tuning levels is left as future work.

The first model used is ResNet50, which is one of the CNN architectures. ResNet50 implements residual connections to solve the degradation problem in deep networks by introducing identity shortcuts [35], allowing gradients to propagate more effectively during training. The basic idea, as shown in Equation (1), is represented by:

$$y = F(x, \{Wi\}) + x \tag{1}$$

where x is the input, F is the residual function (typically a series of convolutional, batch norm, and activation layers), and $\{Wi\}$ It is a learnable parameter. The presence of skip connections allows the network to learn the residual mapping more easily. This model is initialized with pre-trained weights from ImageNet, which is a collection of weights from training on a large dataset containing dozens of images from various classes. By utilizing these weights, the model already has an initial understanding of basic visual patterns such as edges, textures, and shapes, so it does not need to learn from scratch. This approach speeds up the training process and increases accuracy because the model only needs to adjust its weights to the characteristics of the dataset used. In this case, the ResNet50 architecture used can be seen in Figure 3.



Layer Name	Output Shape	Number of Parameters
input_layer_2	(None, 224, 224, 3)	0
conv1_pad	(None, 230, 230, 3)	0
conv1_conv	(None, 112, 112, 64)	9472
conv1_bn	(None, 112, 112, 64)	256
conv1_relu	(None, 112, 112, 64)	0
...	...	...
conv5_block3_2_relu	(None, 7, 7, 512)	0
conv5_block3_3_conv	(None, 7, 7, 2048)	1050624
conv5_block3_3_bn	(None, 7, 7, 2048)	8192
conv5_block3_add	(None, 7, 7, 2048)	0
conv5_block3_out	(None, 7, 7, 2048)	0

Figure 3. ResNet50 Architecture

As shown in Figure 3, the ResNet50 architecture used in this study consists of several main components. This model was developed through programming using TensorFlow and Keras, which allows flexible implementation and modification of the architecture. The model receives input in the form of a 224×224 pixel image with 3 color channels (RGB). The initial layer of the model consists of several convolution and normalization operations, such as conv1_pad, conv1_conv, conv1_bn, and conv1_relu, which aim to extract basic features from the input image. In order for the model to adapt to the dataset used, the last layer of the model is replaced with several additional layers, namely the Global Average Pooling layer, the Dense layer with 256 neurons and ReLU activation, and the normalization and dropout layer of 30% to reduce the risk of overfitting. The last output layer consists of the number of neurons corresponding to the number of character classes available, with a softmax activation function to perform classification.

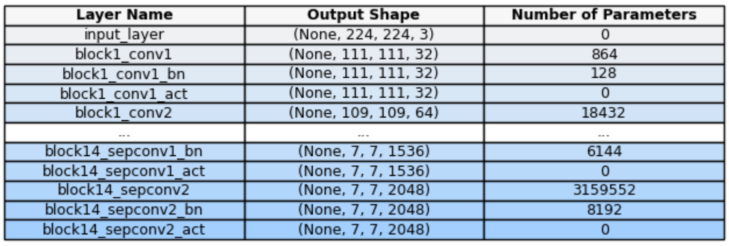
For the 2nd architecture used in the study, namely Xception, this model is known as an extension of the Inception architecture that replaces the standard Inception modules with *depthwise separable convolutions* [36]. As shown in Figure 4, the architecture of this model consists of several main components that enable efficient feature extraction. This form of convolution reduces the number of parameters and computational cost by decomposing a standard convolution into two operations. First, the operation is defined by a depthwise convolution, as shown in Equation (2), which applies a single filter to each input channel:

$$(D * W)_{ij} = \sum_{c=1}^{C} D_{ij}^{c} \cdot W_{ij}^{c} \tag{2}$$

where D_{ij}^{c} denotes the value at position (i, j) in the c-th channel of the input feature map, and W_{ij}^{c} represents the kernel at the same position and channel. The result is then passed to a pointwise convolution, as expressed in Equation (3), using a 1×1 kernel to combine all channel outputs:

$$(P * V)_{ij} = \sum_{k=1}^{K} P_{ij}^{k} \cdot V_{ij}^{k} \tag{3}$$

where P is the result from the depthwise convolution, and V is the pointwise kernel. The combination of both operations forms what is known as a *separable convolution*, which Xception stacks to form a deep architecture.



Layer Name	Output Shape	Number of Parameters
input_layer	(None, 224, 224, 3)	0
block1_conv1	(None, 111, 111, 32)	864
block1_conv1_bn	(None, 111, 111, 32)	128
block1_conv1_act	(None, 111, 111, 32)	0
block1_conv2	(None, 109, 109, 64)	18432
...	...	...
block14_sepconv1_bn	(None, 7, 7, 1536)	6144
block14_sepconv1_act	(None, 7, 7, 1536)	0
block14_sepconv2	(None, 7, 7, 2048)	3159552
block14_sepconv2_bn	(None, 7, 7, 2048)	8192
block14_sepconv2_act	(None, 7, 7, 2048)	0

Figure 4. Xception Architecture

The initial layers of the model in Figure 4 consist of a standard convolution operation on block1_conv1, followed by batch normalization (block1_conv1_bn) and non-linear activation (block1_conv1_act). This process aims to extract basic features from the input image before passing through subsequent layers.

Next, the model uses a separable convolution layer, which is a characteristic of the Xception architecture. Separable convolution divides the convolution operation into two stages: depthwise convolution and pointwise convolution. This can improve computational efficiency without sacrificing model performance. Examples of this layer are seen in block14_sepconv1_bn, block14_sepconv1_act, and block14_sepconv2. In the final stage, the model uses the last convolution layer (block14_sepconv2) with 2,048 filters to generate high-level features before classification. Softmax activation is used in the output layer to map the results to the appropriate class. During the training process, a transfer learning strategy can be applied, where only the last few layers are retrained, while the pre-trained weights in the early layers are retained to capture the general features of the dataset used.

To ensure optimal learning, the Adam optimizer is used with an initial learning rate of 0.0001. Adam updates the model parameters using both the first and second moment estimates of the gradients [37], as shown in Equation (4), which combines adaptive learning rates with momentum:

$$\theta_t = \theta_{\{t-1\}} - \alpha \cdot \frac{mt}{\sqrt{v_t} + \in} \tag{4}$$

where mt and vt are the bias-corrected estimates of the first and second moments of the gradients, α is the learning rate, and $\in$ is a small constant to maintain numerical stability.
During training, several techniques are applied to avoid overfitting, such as:

- Early Stopping, which will stop training when the model's performance on the validation data begins to decline for several consecutive epochs.
- ReduceLROnPlateau, which will automatically reduce the learning rate when the val_loss does not improve after several epochs so that the model can continue to learn more effectively.

The number of training epochs (20) was determined empirically after monitoring training and validation losses. However, no formal hyperparameter optimization was conducted, and further improvements could be obtained by experimenting with learning rate schedules, regularization strategies (e.g., weight decay), or using automated tuning tools. These aspects remain areas for future exploration to ensure optimal convergence for each architecture under different training conditions. If training is insufficient, the model may not adequately learn the data patterns, whereas excessive training may lead to overfitting. Before analyzing the results of the model evaluation on the confusion matrix, it is necessary to first observe how the model training process takes place. Figure 5 shows a graph of the results of model training using the ResNet50 architecture, which illustrates the trend of changes in accuracy and loss during the training process.

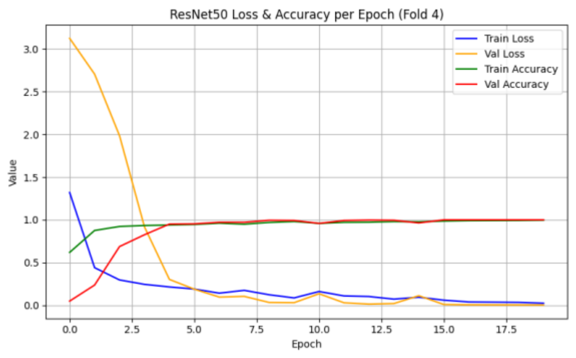


Figure 5. Best ResNet50 training results

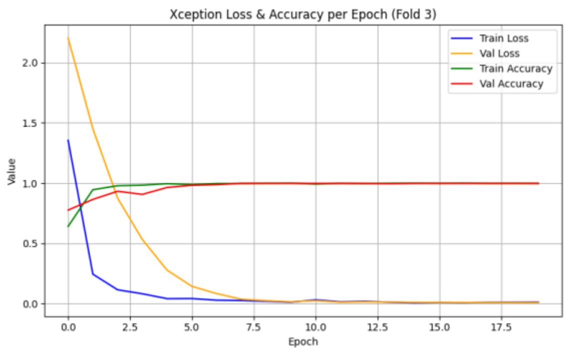


Figure 6. Best Xception training results

During training, loss and accuracy metrics were carefully monitored to evaluate the model's learning performance. As illustrated in Figure 5, the model employing the ResNet50 architecture, trained using 5-fold cross-validation, demonstrates excellent and rapid convergence in Fold 4. The training accuracy increases sharply within the first few epochs, exceeding 95% by epoch 5 and eventually reaching 100% by the final epoch. Concurrently, the training loss decreases significantly during the early stages and stabilizes at a very low value in later epochs. The validation performance closely follows the training trend, with validation accuracy, precision, and recall all reaching 100%, and validation loss approaching zero, indicating excellent generalization. The training curve shown represents the best result achieved by ResNet50 in this cross-validation setup. Meanwhile, a picture of the training results from Xception is shown in Figure 6.

Based on Figure 6, illustrates the training dynamics of the Xception model during Fold 3. The model exhibits a rapid increase in both training and validation accuracy within the first few epochs, reaching close to 100% accuracy around epoch 3–4. After this point, accuracy remains stable, indicating convergence and consistent performance. Correspondingly, both training and validation loss show a sharp decline at the beginning of training. Training loss quickly drops to near zero, while validation loss also steadily decreases and stabilizes at a low value, indicating that the model is not overfitting and maintains good generalization to validation data. The graphic patterns shown in Figures 5 and 6 show that the model can quickly adjust its weights to the training data, with a relatively stable learning trend after the initial phase. The curve that does not show significant fluctuations also indicates that the model has reached convergence without any indication of overfitting.

After the model has been trained, the best weight obtained during training is used for the classification validation process on the test data. This stage involves evaluating the model’s ability to classify Korean characters using metrics such as accuracy, precision, recall, F1-score, and AUC. These results serve as the basis for assessing the overall performance of the model, which will be further discussed in the evaluation section.

3.1.4. Evaluation

The trained model is then evaluated with the available data to measure its ability to recognize learned patterns and ensure its performance remains optimal in classifying. This evaluation is done by calculating key metrics such as accuracy, precision, recall, AUC, and F1-score, which are obtained directly from the programming results using the Keras and Scikit-learn libraries. The best results from the model are obtained based on the weights that have been stored during the training process, which can be seen in Tables 2 and 3.

Table 2. Performance Result of ResNet50 Architecture

Fold (K=5)	Accuracy (%)	Precision (%)	Recall (%)	AUC (%)	F1-Score (%)
1	99.48	100	99.48	100	99.65
2	99.48	99.74	99.22	100	99.46
3	100	100	100	100	100
4	100	100	100	100	100
5	99.48	99.74	99.48	100	99.50

Table 2 shows a summary of the performance evaluation results of the model with the ResNet50 architecture with 5-fold cross-validation. Based on the results shown in the table, the model demonstrates consistently high performance across all folds, with accuracy ranging from 99.48% to 100%. The AUC values remain at 100% in every fold, indicating excellent capability in distinguishing between classes. Precision, Recall, and F1-score metrics also show minimal variation and remain at near-perfect values. Among all folds, the best performance was observed in Fold 4, which achieved perfect scores across all metrics (Accuracy, Precision, Recall, AUC, and F1-Score all at 100%). Although Fold 3 also shows identical metric values, Fold 4 had a lower loss value, indicating better model generalization and stability. Therefore, Fold 4 is considered the best-performing fold for the ResNet50 model in this evaluation.

Table 3. Performance Result of Xception Architecture

Fold (K=5)	Accuracy (%)	Precision (%)	Recall (%)	AUC (%)	F1-Score (%)
1	99.22	99.48	99.22	99.48	99.31
2	99.48	99.22	99.48	99.72	99.31

Fold (K=5)	Accuracy (%)	Precision (%)	Recall (%)	AUC (%)	F1-Score (%)
3	99.74	99.74	99.74	100	99.74
4	99.74	99.48	99.48	100	99.48
5	99.74	99.74	99.74	100	99.74

Table 3 presents the evaluation results of the Xception model using 5-fold cross-validation. The results demonstrate that the model consistently performs well across all folds, with high values in all evaluation metrics. Accuracy, precision, recall, AUC, and F1-score all exceed 99%, with the highest scores reaching 99.74% across multiple metrics in folds 3, 4, and 5.

These results indicate that the Xception model is highly reliable in recognizing Korean character patterns. The confusion matrix further supports this, showing accurate class-wise prediction distributions. Figure 7 illustrates the confusion matrix for the best-performing fold of the Xception model, confirming its strong classification capability across all character categories.

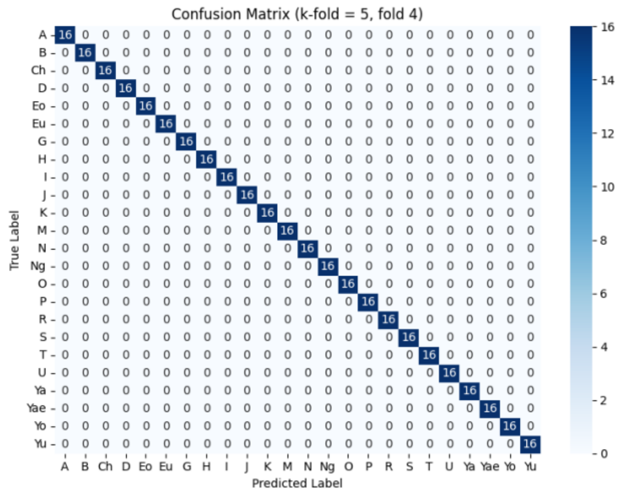


Figure 7. ResNet50 Confusion Matrix (fold 4)

Figure 7 shows the confusion matrix of the modeling results using the ResNet50 architecture. The matrix clearly demonstrates perfect classification performance, as all predictions lie exactly on the main diagonal with no misclassifications. Each class has 16 correct predictions, indicating that the model successfully recognized all samples in each class. The uniformly dark blue diagonal also reflects consistent and accurate predictions across all 24 classes. This result supports the conclusion that Fold 4 achieved the best overall performance, with both perfect metric values and zero classification errors. Meanwhile, the confusion matrix results of the modeling using the Xception architecture for Korean character classification are shown in Figure 8.

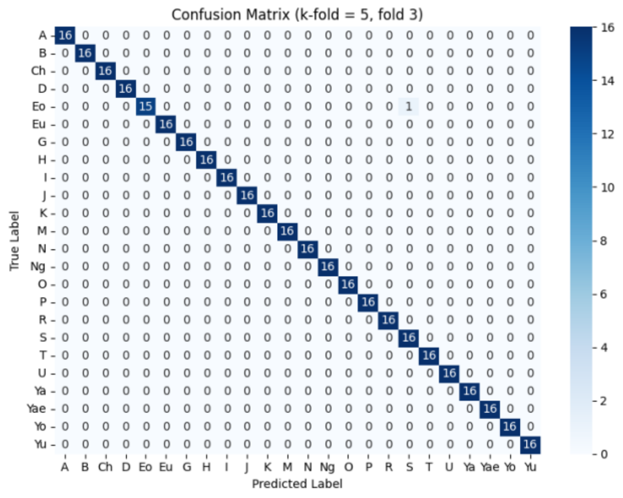


Figure 8. Xception Confusion Matrix (fold 3)

Figure 8, shows the confusion matrix of the Xception model for fold 3. The model demonstrates excellent performance, as reflected by the strong concentration of values along the main diagonal, indicating that most characters were correctly classified. All character classes were perfectly predicted except for a single misclassification in the character "Eo", which was incorrectly predicted as "S". This minor error may stem from visual similarities between the two characters. Despite this, the model achieves near-perfect performance with very few classification mistakes, showcasing its strong ability to distinguish Korean characters. To further investigate this misclassification, a Grad-CAM visualization was applied and is shown in as shown in Figure 9 (for "Ch") and Figure 10 (for "Eo").

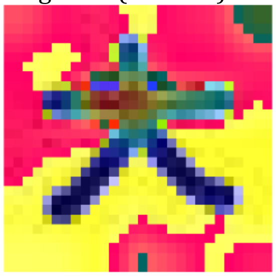


Figure 9. Grad-CAM visualization for the 'ch' character.

The visualization shown in Fig 9 highlights the regions in the image that contributed most to the model's decision. The heatmap shows that the model focuses on the diagonal strokes forming an *X-like pattern* and the central intersection, which are distinctive features of the Korean character "Ch" (ㅊ). This indicates that the model has successfully identified the key visual structure of the character. However, the similarity in shape to other characters may have influenced the misclassification.

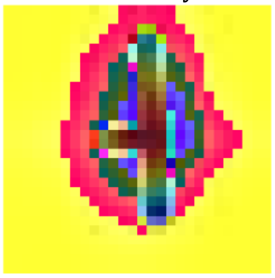


Figure 10. Grad-CAM visualization for the 'eo' character

The Grad-CAM visualization for the character "Eo" (ㅓ), as shown in Figure 10, shows that the model heatmap focuses on the vertical line in the middle as well as some of the short horizontal lines, which are the main features of this character. Despite this focus, the confusion matrix in Figure 8 shows that there was one case of misclassification for the character 'Eo'. This indicates that although the model recognizes the main visual features well, there is still a possibility of confusion of similar shapes with other characters, leading to misprediction.

The Grad-CAM heatmap supports this interpretation by showing that even though the model's attention is on the correct area, it is not fully able to distinguish the distinctive features of the letter 'Ch' and 'Eo', especially if there are variations in writing style or visual shapes that resemble other letters. Grad-CAM is used here to visualize these key features by analyzing the gradients of the final convolutional layer of the model. In this case, the model likely uses the final convolutional layer (often called conv5_block3_out in architectures like ResNet50) to learn important features. Using Grad-CAM in frameworks like TensorFlow or Keras helps in understanding what the model is focusing on but also how the model makes decisions, offering insight into both successful and erroneous classifications.

Grad-CAM visualization provides interpretive insights into the model's decision-making process. However, since misclassifications only occur in two classes, the interpretability analysis performed is still limited in scope. Nevertheless, this approach still shows potential in identifying model weaknesses. For further studies, applying similar analysis to datasets with more diverse error distributions can provide a more comprehensive understanding of inter-class confusion patterns, especially between visually similar characters such as consonants and vowels.

While Grad-CAM offers qualitative insight, a statistical comparison is also necessary to quantitatively assess model superiority. To further validate the performance differences between ResNet50 and Xception, a sign test was conducted based on results from k-fold cross-validation. The sign test, a non--parametric method appropriate for small sample comparisons across paired observations [38] supports the conclusion that both models perform comparably well, and that the choice between them may depend on secondary considerations such as model size or inference speed rather than classification accuracy alone. Table 4 presents the mean values of each metric along with the number of folds where one model outperformed the other.

Table 4. Sign Test Evaluation of ResNet50 vs Xception Across K-Folds

Metric	ResNet50 Mean (%)	Xception Mean (%)	Fold Compared	ResNet50 > Xception	ResNet50 < Xception	Ties	Sign Test p-value
Accuracy	99.96	99.58	4	3	1	1	0.3125
Precision	99.90	99.53	4	4	0	1	0.0625
Recall	99.64	99.53	5	3	2	0	0.5000
AUC	100	99.84	2	2	0	3	0.2500
F1-Score	99.72	99.52	5	4	1	0	0.1875

As shown in Table 4, ResNet50 outperformed Xception in most folds across all metrics, particularly in precision and F1-score, where ResNet50 achieved higher values in 4 out of 5 folds. However, the resulting p-values from the sign test (e.g., p = 0.0625 for precision and p = 0.1875 for F1-score) suggest that these differences are not statistically significant at the conventional alpha level of 0.05. The AUC metric showed perfect performance in two folds (with three ties), and similarly did not reach significance (p = 0.25). These findings imply that while ResNet50 shows a slight consistent edge over Xception in mean performance, the difference is not strong enough to be deemed statistically significant with the current dataset.

3.2. *Discussion*

Compared to previous studies using VGG-16, the models in this study showed better performance across all metrics evaluated. As seen in Table 5, the previous VGG-16 achieved significant results with accuracy above 99%, with precision and recall still below 96%. In contrast, ResNet50 and Xception in this study consistently reached higher precision, recall, and F1-score values, with ResNet50 achieving a perfect score of 100%. This highlights the advantage of using more advanced architectures for Korean character classification.

Table 5. Comparison with Previous Research

Study	Model	Accuracy (%)	Precision (%)	Recall (%)	AUC (%)	F1-Score (%)
This Study	ResNet50	100	100	100	100	100
	Xception	99.74	99.74	99.74	100	99.74
Hartati [13]	VGG-16 (without binarization)	99.52	95.56	94.11	-	-
	VGG-16 (with binarization)	99.42	95.94	93.11	-	-
Radikto [12]	Backpropagation	99.10	-	-	-	-
Oktaviani [10]	ANN	97	-	-	-	-
Aris [11]	Backpropagation	80.83	-	-	-	-
Snowberger [14]	RNN	90.5	-	-	-	-

This comparison, as shown in Table 5, underlines the importance of model architecture in pattern recognition tasks. While VGG-16 and other classical models, such as ANN and backpropagation networks, had strong performance in previous studies, the results in this study show that ResNet50 and Xception produced superior outcomes across multiple evaluation metrics. Both models achieved notably higher accuracy, precision, recall, F1-score, and AUC values, ResNet50 even achieving perfect scores on all metrics, indicating superior generalization and character recognition capabilities. In contrast, Snowberger's RNN model achieved 90.5% accuracy, which is notably lower than both classical and deep CNN-based models.

Although the proposed model shows excellent classification performance on the Hangeul dataset from Kaggle, limitations arise from the lack of external validation. Without testing on other datasets such as HCL2000 or real-world handwriting samples, the generalizability of the model cannot be ascertained. For future research, validation on completely new data or simulation of domain shift scenarios can provide a more accurate picture of the model's robustness in the context of real applications.

4. CONCLUSION

Based on the evaluation results, it can be seen that the model architecture greatly influences the classification performance. ResNet50 consistently achieved the highest average accuracy, precision, recall, F1-score, and AUC, even reaching 100% in multiple metrics. While the Xception model also performed strongly and achieved a high level of accuracy (99.74%) and AUC (100%), ResNet50 outperformed Xception in most folds across all evaluation metrics. A sign test was conducted to assess whether these differences were statistically significant. The results showed that although ResNet50 had higher mean scores and outperformed Xception in most folds, the differences were not statistically significant at the conventional $\alpha = 0.05$ level. However, some limitations need to be noted, especially the relatively small dataset size (1,920 images) for deep learning models, which increases the risk of overfitting even with augmentation. In addition, hyperparameter tuning is still limited, and the Grad-CAM analysis only covers two cases of misclassification. In addition, testing on independent datasets and applying robustness evaluation can improve the practical reliability of the model. A more comprehensive analysis of Grad-CAM on misclassification will also be conducted to better understand the failure patterns. However, selecting the best model still depends on the specific needs and characteristics of the dataset used, so further analysis of the model's reliability in real-world conditions is still needed. This model could also be extended for practical applications such as real-time OCR systems or multilingual handwritten character recognition, particularly in a multilingual environment.

ACKNOWLEDGEMENTS

We thank our Institut Teknologi Garut and Bicol University colleagues, who have been collaborating and provided insight and expertise that greatly assisted the research. However, they may not agree with all of the interpretations/conclusions of this paper.

REFERENCES

[1] S. Jung-Eun and S. Yoonhee, "Hallyu Story with Statistic Indonesia," *Korean Foundation for International Cultural Exchange (KOFICE)*, Korea, pp. 1–23, Feb. 2022.
[2] T. P. Pramadya and J. Oktaviani, "Korean Wave (Hallyu) dan Persepsi Kaum Muda di Indonesia: Peran Media dan Diplomasi Publik Korea Selatan," *Insignia: Journal of International Relations*, vol. 8, no. 1, p. 87, Apr. 2021, doi: 10.20884/1.ins.2021.8.1.3857.
[3] Hestianingsih, "Pengaruh K-Wave, Korea Masuk Daftar 10 Bahasa Terpopuler 2023," Wolipop.
[4] A. A. Rosyadi, "Karakteristik Surel Bisnis Berbahasa Korea," *JLA (Jurnal Lingua Applicata)*, vol. 4, no. 1, p. 13, Oct. 2020, doi: 10.22146/jla.57448.
[5] C. Young-mee, S. Ho-Min, S. Sung-Ock, and C. Schulz, *Integrated Korean*, Third. University of Hawaii Press, 2020.
[6] Y. Rizki, R. Medikawati Taufiq, H. Mukhtar, and D. Putri, "Klasifikasi Pola Kain Tenun Melayu Menggunakan Faster R-CNN," *IT Journal Research and Development*, vol. 5, no. 2, pp. 215–225, Jan. 2021, doi: 10.25299/itjrd.2021.vol5(2).5831.
[7] W. S. Eka Putra, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," *Jurnal Teknik ITS*, vol. 5, no. 1, Mar. 2016, doi: 10.12962/j23373539.v5i1.15696.
[8] Y. Septiana, A. Mulyani, D. Kurniadi, and H. Hasanudin, "Handwritten Recognition of Hiragana and Katakana Characters Based on Template Matching Algorithm," *IOP Conf Ser Mater Sci Eng*, vol. 1098, no. 3, p. 032093, Mar. 2021, doi: 10.1088/1757-899X/1098/3/032093.

[9] R. E. Wiratna, A. Sahputro, B. D. Hadi, and E. Y. Puspaningrum, "Pengenalan Karakter Hijaiyah Menggunakan Metode Convolutional Neural Network (CNN)," *Jurnal Teknologi Informasi dan Komunikasi*, vol. 17, no. 1, Jul. 2022, doi: 10.33005/scan.v17i1.2937.

[10] S. Oktaviani, C. A. Sari, E. Hari Rachmawanto, and D. R. Ignatius Moses Setiadi, "Optical Character Recognition for Hangul Character using Artificial Neural Network," Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 34–39. doi: 10.1109/iSemantic50169.2020.9234215.

[11] A. Aris Widodo, M. Y. Izza Mahendra, and M. Z. Sarwani, "Recognition of Korean Alphabet (Hangul) Handwriting into Latin Characters Using Backpropagation Method," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 3, no. 2, pp. 50–57, Nov. 2021, doi: 10.25139/ijair.v3i2.4210.

[12] Radikto and Rasiban, "Pengenalan Pola Huruf Hangeul Korea Menggunakan Jaringan Syaraf Tiruan Metode Backpropagation dan Deteksi Tepi Canny," 2022. doi: https://doi.org/10.31004/jpdk.v4i5.6722.

[13] E. Hartati, D. Alamsyah, and Nataliatara, "Korean Letter Handwriting Recognition Using Convolutional Neural Network Method Vgg-16 Arsitektur Architecture," *International Journal of Artificial Intelligence and Robotic Technology*, vol. 1, no. 3, Mar. 2021, Accessed: May 27, 2024. [Online]. Available: https://journal.sracademicpublishing.org/index.php/IJAIRTec/article/view/33/pdf

[14] A. D. Snowberger and C. H. Lee, "Handwritten Hangul Graphemes Classification Using Three Artificial Neural Networks," *Journal of Information and Communication Convergence Engineering*, vol. 21, no. 2, pp. 167–173, Jun. 2023, doi: 10.56977/jicce.2023.21.2.167.

[15] A. Hussain and A. Aslam, "Ensemble-based Approach Using Inception V2, VGG-16, and Xception Convolutional Neural Networks for Surface Cracks Detection," *Journal of Applied Research and Technology*, vol. 22, no. 4, pp. 586–598, Aug. 2024, doi: 10.22201/icat.24486736e.2024.22.4.2431.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385

[17] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.02357

[18] N. Duklan, S. Kumar, H. Maheshwari, R. Singh, S. D. Sharma, and S. Swami, "CNN Architectures for Image Classification: A Comparative Study Using ResNet50V2, ResNet152V2, InceptionV3, Xception, and MobileNetV2," *SSRG International Journal of Electronics and Communication Engineering*, vol. 11, no. 9, pp. 11–21, Sep. 2024, doi: 10.14445/23488549/IJECE-V11I9P102.

[19] Y. Mulyani, R. Kurniawan, P. B. Wintoro, M. Komarudin, and W. M. Al-Rahmi, "International Journal of Aviation Science and Engineering AVIA Systematic Comparison of Machine Learning Model Accuracy Value Between MobileNetV2 and XCeption Architecture in Waste Classification System," vol. 4, no. 2, pp. 75–82, 2022, doi: 10.47355/AVIA.V4I2.70.

[20] T. J. Bradshaw, Z. Huemann, J. Hu, and A. Rahmim, "A Guide to Cross-Validation for Artificial Intelligence in Medical Imaging," Jul. 01, 2023, *Radiological Society of North America Inc.* doi: 10.1148/ryai.220232.

[21] I. K. Nti, O. Nyarko-Boateng, and J. Aning, "Performance of Machine Learning Algorithms with Different K Values in K-fold CrossValidation," *International Journal of Information Technology and Computer Science*, vol. 13, no. 6, pp. 61–71, Dec. 2021, doi: 10.5815/ijitcs.2021.06.05.

[22] M. G. M. Abdolrasol *et al.*, "Artificial Neural Networks Based Optimization Techniques: A review," Nov. 01, 2021, *MDPI*. doi: 10.3390/electronics10212689.

[23] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex and Intelligent Systems*, vol. 8, no. 3, pp. 2663–2693, Jun. 2022, doi: 10.1007/s40747-021-00637-x.

[24] S. Agha, S. Nazir, M. Kaleem, F. Najeeb, and R. Talat, "Performance evaluation of reduced complexity deep neural networks," *PLoS One*, vol. 20, no. 3 MARCH, Mar. 2025, doi: 10.1371/journal.pone.0319859.

[25] Z. Zhou, X. Yang, H. Ji, and Z. Zhu, "Improving the classification accuracy of fishes and invertebrates using residual convolutional neural networks," *ICES Journal of Marine Science*, vol. 80, no. 5, pp. 1256–1266, Jul. 2023, doi: 10.1093/icesjms/fsad041.

[26] M. Abdullah Al Alamin and G. Uddin, "Challenges and Barriers of Using Low Code Software for Machine Learning," *arXiv e-prints*, p. arXiv:2211.04661, Nov. 2022, doi: 10.48550/arXiv.2211.04661.

[27] Jamescasia, "Handwritten Hangul Characters," Kaggle. Accessed: Jan. 22, 2025. [Online]. Available: https://www.kaggle.com/datasets/wayperwayp/hangulkorean-characters/data

[28] B. O. Soufiene and C. Chakraborty, *Machine Learning and Deep Learning Techniques for Medical Image Recognition*. in Advances in Smart Healthcare Technologies. CRC Press, 2023. [Online]. Available: https://books.google.co.id/books?id=BSfdEAAAQBAJ

[29] Y. Zhang, D. Hong, D. McClement, O. Oladosu, G. Pridham, and G. Slaney, "Grad-CAM helps interpret the deep learning models trained to classify multiple sclerosis types using clinical brain magnetic resonance imaging," *J Neurosci Methods*, vol. 353, p. 109098, 2021, doi: https://doi.org/10.1016/j.jneumeth.2021.109098.

[30] S. Purnamawati, D. Rachmawati, G. Lumanauw, R. F. Rahmat, and R. Taqyuddin, "Korean Letter Handwritten Recognition Using Deep Convolutional Neural Network on Android Platform," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2018. doi: 10.1088/1742-6596/978/1/012112.

[31] Y. Son and J. W. Park, "Detecting Forged Audio Files Using 'Mixed Paste' Command: A Deep Learning Approach Based on Korean Phonemic Features," *Sensors*, vol. 24, no. 6, Mar. 2024, doi: 10.3390/s24061872.

[32] K. Maharana, S. Mondal, and B. Nemade, "A review: Data Pre-processing and Data Augmentation Techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022, doi: https://doi.org/10.1016/j.gltp.2022.04.020.

[33] R. Sarki, K. Ahmed, H. Wang, Y. Zhang, J. Ma, and K. Wang, "Image Preprocessing in Classification and Identification of Diabetic Eye Diseases," *Data Sci Eng*, vol. 6, no. 4, pp. 455–471, 2021.

[34] I. Muraina, "Ideal Dataset Splitting Ratios in Machine Learning Algorithms: General Concerns for Data Scientists and Data Analysts," in *7th international Scientific Research Conference*, 2022, pp. 496–504.

[35] M. Shafiq and Z. Gu, "Deep Residual Learning for Image Recognition: A Survey," *Applied Sciences*, vol. 12, no. 18, p. 8972, Sep. 2022, doi: 10.3390/app12188972.

[36] D. A. Alkurdi, M. Cevik, and A. Akgundogdu, "Advancing Deepfake Detection Using Xception Architecture: A Robust Approach for Safeguarding against Fabricated News on Social Media," *Computers, Materials & Continua*, vol. 81, no. 3, pp. 4285–4305, 2024, doi: 10.32604/cmc.2024.057029.

[37] I. H. Kartowisastro and J. Latupapua, "A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using Convolutional Neural Networks," *Revue d'Intelligence Artificielle*, vol. 37, no. 4, pp. 1123–1130, Aug. 2023, doi: 10.18280/ria.370424.

[38] A. D. Hutson and H. Yu, "The Sign Test, Paired Data, and Asymmetric Dependence: A Cautionary Tale," *American Statistician*, vol. 77, no. 1, pp. 35–40, 2023, doi: 10.1080/00031305.2022.2110938.