The object of this research is the BISINDO alphabet gestures which are static hand movements used by the Deaf community in Indonesia to communicate, with each letter having a unique hand pattern influenced by culture and regional variations. The problem solved is the accuracy and performance of BISINDO hand gesture classification which is less than optimal due to complex gesture variations and computational limitations on lightweight devices.

This study examines the performance of BISINDO alphabet gesture classification using deep learning models with 4 architectures: VGG-19, ResNet-50, MobileNetV2, and Inception-V3. This object is a collection of images used as a dataset consisting of 10,400 images (7,280 training, 2,080 validation, and 1,040 testing). The results show that the MobileNetV2 and VGG-19 architectures achieve the highest accuracy of 100%, followed by Inception-V3 (99%) and ResNet-50 (98%). The results of this study indicate that the superior performance of MobileNetV2 is due to its efficient depthwise separable convolutional architecture, while the superiority of VGG-19 lies in its very deep architecture and the use of small convolutional filters to capture very detailed hierarchical features of gestures. Meanwhile, Inception-V3 excels thanks to the inception module that captures gesture features at various scales. The results of this performance comparison, MobileNetV2 is the most superior because of its computational efficiency that supports high performance, as well as adaptation to complex variations in BISINDO alphabet gestures. The results of this study can also be applied in the fields of education and communication for the deaf community, especially in embedded applications, thus enabling real-time sign language accessibility

Keywords: BISINDO, alphabet gestures, performance, CNN, VGG-19, MobileNetV2, ResNet-50, Inception-V3

UDC 004.8
DOI: 10.15587/1729-4061.2025.332096

IDENTIFYING THE BEST MODELS FOR BISINDO ALPHABET GESTURE CLASSIFICATION TO SUPPORT THE COMMUNICATION NEEDS OF THE DEAF COMMUNITY

Ilka Zufria
Lecturer
Department of Computer Science
Universitas Islam Negeri Sumatera Utara
Golf Field's str., 120, Deli Serdang, Indonesia, 20353
Tengku Henny Febriana Harumy
Corresponding author
Lecturer*
E-mail: hennyharumy@usu.ac.id
Syahril Efendi
Lecturer*
*Department of Computer Science
Universitas Sumatera Utara
Dr. T. Mansur str., 9, Medan, Indonesia, 20222

Received 24.03.2025
Received in revised form 25.05.2025
Accepted 05.06.2025
Published 30.12.2025

How to Cite: *Zufria, I., Harumy, T. H. F., Efendi, S. (2025). Identifying the best models for BISINDO alphabet gesture classification to support the communication needs of the deaf community. Eastern-European Journal of Enterprise Technologies, 6 (2 (138)), 26–41. https://doi.org/10.15587/1729-4061.2025.332096*

1. Introduction

Sign language is the primary means of communication for the deaf community, enabling them to interact effectively in a variety of social and educational contexts [1]. In Indonesia, BISINDO (Indonesian Sign Language) is a widely used sign language system characterized by unique gestures that reflect cultural and regional nuances [2]. Despite its important role, a communication gap between the deaf community and the general public still exists due to limited understanding of sign language and the lack of easily accessible assistive technology [3].

Recent advances in computer vision and deep learning have shown the potential to bridge this gap through automatic recognition of sign language gestures, facilitating real-time translation and increasing accessibility [4]. However, the application of these technologies to BISINDO remains under-explored, especially in terms of achieving high accuracy and computational efficiency on resource-constrained devices, which are often the primary means of communication in developing regions such as Indonesia.

Further relevant issues currently include the low accuracy of BISINDO gesture classification, computational limitations on lightweight devices, the lack of comprehensive datasets, and the communication gap for the deaf community. The limitations of previous studies include the use of inefficient models, lack of comprehensive model comparison, limited datasets, a lack of focus on BISINDO, and model generalization issues. This research idea is relevant because it directly addresses these issues by comparing deep learning models to find accurate, efficient, and applicable solutions in the context of BISINDO alphabet classification.

The model comparison consists of four leading deep learning (CNN) models: VGG-19, ResNet-50, MobileNetV2, and Inception-V3. These four classification models will later be viewed in terms of accuracy and loss performance from both the training, validation, and testing sets as well as time efficiency in processing them [5]. It should be noted that the four models used in this study do not have the same number of convolutional layers [6]. Therefore, each model gives different results in terms of its performance. Each model follows its own training architecture. To obtain the best accuracy. Initially, the

model is trained with a number of combinations of convolutional layers, density layers, dropouts, and other optimizers that evaluate each model after each iteration; then, the complexity is increased to obtain the best model performance [7].

Therefore, the scientific relevance of this topic lies in its contribution to the field of computer vision, the development of inclusive technologies, addressing local challenges such as BISINDO, and supporting the global agenda on accessibility, making it an important issue and worthy of continued research. The findings of these studies are expected to provide valuable insights into the most optimal model for BISINDO gesture classification and can be applied to the creation of a BISINDO gesture translation system to text or speech directly, which can im`prove social inclusion by facilitating communication between the Deaf community and the general public, for example in everyday interactions, public services, or education.

2. Literature review and problem statement

In paper [4] stated that the CNN model can detect video datasets and can even detect in real time, which is done on a smartphone device with the Android operating system. Regarding performance in recognition, real-time testing managed to achieve its best accuracy of 88.46%. Not only that, the model can also recognize images quickly, with an average detection duration of 24 milliseconds in real-time testing. The weaknesses of this study are still found where the model performance still has not achieved an accuracy above 90%.

In the sign language prediction study [2], it was used the CNN+LSTM method and focused on filtering, layers, and BISINDO objects. Based on the training, validation, and testing models, CNN+LSTM is the optimal model for this item. With this model, the accuracy reaches 90%, loss 19%, and testing 80%. In this study, the weakness is that the testing is not as optimal as for training, which only gets 80% accuracy.

This study [3] introduces a new model that utilizes extracted key points to recognize gesture words in BISINDO. The use of MediaPipe to extract body posture and hand gesture data from videos is a significant step forward. Experimental results demonstrate the effectiveness of the proposed LSTM model with an impressive testing accuracy of 92.857% on validation data and only one error in gesture classification. This study only uses one model, where the performance of other models may provide better performance results.

In this study [8], the SLR conducted aims to provide information on how to develop an image classification model in BISINDO recognition. CNN, LSTM, and CNN-LSTM are the three most widely used algorithms in image recognition. Second, it was also found that the use of a combined CNN-LSTM algorithm and a better dataset in terms of lighting conditions, varying angles, and diverse gestures can help improve the performance of the predictive model. This is because it supports generalization and reduces bias in the model that can cause inaccurate results. This study doesn't compare the performance of each model, but only combines CNN with LSTM and emphasizes the prediction results based on differences in conditions or variations of the image.

This study shows [9] that CNN has a higher and more stable accuracy rate compared to the RNN model, where the CNN model produces an accuracy rate of 89%. While the RNN model shows an accuracy of 88.46%. Both results indicate that the CNN model is superior in classifying BISINDO dataset images. This study only compares the accuracy of the two CNN and RNN algorithms. Not comparing the architecture of one of the algorithms and the accuracy results also only reach 89% without measuring other performance.

This study shows [10] that integrating quality evaluation in a fundus image classification system achieves better performance than without quality evaluation for almost all CNN architectures used. DenseNet-201, without quality evaluation, achieves an overall accuracy of 78.68%, sensitivity of 88.71%, and specificity of 95.92% for DR, sensitivity of 32.38% and specificity of 88.24 for GLC, sensitivity of 34.78% and specificity of 98.30% for AMD, and sensitivity of 86.18% and specificity of 81.48% for NR. For the same model, with quality assessment, the overall accuracy was 85.83%, sensitivity 89.83%, and specificity 96.69% for DR, sensitivity 36.46%, and specificity 89.58% for GLC, sensitivity 72.73%, and specificity 100% for AMD, and sensitivity 87.93%, and specificity 89.52% for NR. This study also compared the performance of several different CNN models, but not for Citra Bisindo but fundus images.

In this paper [11], a lightweight autoencoder classification network based on CBAM mechanism is proposed for corn disease identification. This article compares CBAM-Autoencoder, Autoencoder, MobileNet, VGG16, Inception V3, ResNet-50, and DenseNet201. Experimental analysis proves that the CBAM-Autoencoder network has good performance in corn leaf disease classification, and the average accuracy rate reaches 99.44%. This study has compared the performance of more than 5 CNN models and obtained the maximum average accuracy results. But not about the Bisindo image but corn leaf disease.

Previous studies show that an unexplored problem is providing a comparative analysis of performance on Bisindo alphabet gesture classification using several CNN architectures. From previous research, this comparison of gesture classification performance uses different deep learning algorithms, not one algorithm with other architectures. In this regard, this underlines the need to research to answer three main questions:

1. Provide a comparative analysis of the performance of BISINDO alphabet classification using several CNN architectures, including VGG-19, MobileNetV2, ResNet-50 and Inception-V3, especially in providing learning to improve and find the best training, validation and prediction testing performance that has never been studied before.

2. Provide a comparison of the model's time speed in validating and testing data to impact classification efficiency.

3. Utilize models that are supported by fine-tuning to get the best model with the help of hyperparameter tuning from each architecture. The parameters chosen are the number of nodes in the final layer, the number of dropouts, and the learning rate applied to each architecture [12], which can provide an idea of how the best performance in classification is produced.

3. The aim and objectives of the study

The aim of the study is to identify the best models for BISINDO alphabet gesture classification. This will make it possible to provides a practical and scalable technological foundation to support the communication needs of the Deaf community, while encouraging further innovation in assistive technology.

To achieve this aim, the following objectives are accomplished:

– to classify and predict BISINDO alphabet gesture images using four models (VGG-19, ResNet-50, MobileNetV2, and Inception-V3) and apply optimization techniques such as Adam optimizer, regularization and hyperparameter tuning to prevent overfitting;

– to measure the accuracy performance values of the four models (VGG-19, ResNet-50, MobileNetV2, and Inception-V3) from the testing process and compare the results of the test accuracy with the confusion matrix to determine the best model;

– to analyze between accuracy and loss performance through comparative experiments in the training and validation processes, documenting the advantages and disadvantages of each model in terms of the stability of the accuracy and loss values;

– to recommend the best model based on the overall comparative analysis of the performance of the BISINDO alphabet classification using several architectures including VGG-19, MobileNetV2, ResNet-50 and Inception-V3 in terms of model performance during training, validation and testing, comparison of time efficiency, accuracy and loss values and accuracy and doubt of alphabet prediction on the testing dataset.

4. Materials and methods

The object of study is the BISINDO alphabet gesture (Indonesian Sign Language), which is used to represent the letters of the alphabet (A-Z). BISINDO is a natural sign language used by the Deaf community in Indonesia, which has unique gesture characteristics influenced by culture and regionality. This research specifically focuses on static gestures that represent the letters of the alphabet, which are then classified using four deep learning models: VGG-19, MobileNetV2, ResNet-50, and Inception-V3. In addition, this study also discusses model performance in terms of accuracy (calculated through metrics such as accuracy, precision, recall, and F1 Score using a confusion matrix), computational efficiency (time spent on validation and testing and resource requirements) [13], and the balance between the two for practical applications on devices with limited resources. These subjects include Deaf individuals whose gestures are represented in the dataset used for model training, validation, and testing. The Deaf community in Indonesia, estimated to number around 2.5 million people according to data from the Indonesian Ministry of Health in 2022, is the main focus because they are the end users who will benefit from the BISINDO gesture recognition system. Hand gestures produced by individuals in the community, which vary based on factors such as personal style, age, cultural background, or region of origin.

The initial hypothesis in this study focused on the performance of deep learning models in BISINDO gesture classification. It was hypothesized that models with modern architectures such as Inception-V3 and ResNet-50 would provide the highest accuracy due to advanced mechanisms such as inception modules and residual connections that can capture complex gesture features. It was also expected that MobileNetV2 would be the most computationally efficient with only 3.5 million parameters, suitable for mobile devices, thanks to depthwise separable convolution. Furthermore, it was hypothesized that MobileNetV2 would be the most optimal model in balancing accuracy and efficiency for practical applications. In contrast, VGG-19 was predicted to be less efficient and prone to overfitting on the limited BISINDO dataset.

It is assumed that the BISINDO alphabet gesture dataset used in this study is representative enough to train and test deep learning models, although it may have limitations in the amount and variety of data. It was assumed that the BISINDO alphabet gestures in the dataset (in the form of images or videos) have consistent visual features (e. g., hand shape, finger angle, hand position) that can be extracted by models such as VGG-19, ResNet-50, MobileNetV2, and Inception-V3. In addition, the researchers also assume that the data augmentation techniques (such as rotation or lighting changes) or data preprocessing can compensate for the limited variations in the dataset, so that the model can learn the gesture features well.

It is assumed that the four selected deep learning models – VGG-19, ResNet-50, MobileNetV2, and InceptionV3 can effectively capture the BISINDO gesture features because of their architectures that have been proven in the task of alphabet gesture image recognition. It was assumed that modern architectures such as inception modules (Inception-V3) and residual connections (ResNet-50) will be superior in handling gesture complexity compared to simpler models such as VGG-19, while MobileNetV2 will be efficient due to depthwise separable convolution. It was also assumed that optimization techniques such as Adam optimizer, dropout (range 0.3 to 0.7), and L2 regularization (coefficient 0.001) can prevent overfitting [14], even on the limited BISINDO dataset.

It is assumed that the model trained on the BISINDO dataset can generalize well to unseen data (validation and testing data), especially after applying regularization techniques and hyperparameter tuning (e. g., learning rate 0.00001 to 0.001). It was also assumed that evaluation metrics such as accuracy, precision, recall, and F1 Score calculated through the confusion matrix will provide an accurate picture of the model's performance, even though testing may be done under controlled conditions (e. g., uniform lighting or simple background).

It is assumed that BISINDO alphabet gesture recognition using deep learning models will have a significant social impact on the Deaf community in Indonesia, which is estimated to number 2.5 million people (Ministry of Health of the Republic of Indonesia, 2022) [15]. It was assumed that this system could be applied in practical applications such as real-time translation or learning aids, which would support social inclusion and more effective communication for the Deaf community, although this research only focused on static alphabet gestures.

It is assumed that the images used in this study were acquired using the same technique and equipment, which may not account for variations in imaging protocols.

In this study, several stages will be carried out, as seen in Fig. 1. The topic of this study Performance Comparison of VGG-19, ResNet-50, MobileNetv2, And Inception-V3 in the context of Bisindo alphabet gesture classification next, there are several steps taken as follows:

A. Data augmentation.

In the augmentation, it is possible to utilize 10,400 images of Bisindo alphabet gestures. These were split into 7280 for training, 2080 for experimental validation, and 1040 for testing (70:20:10). During this stage, let's configure normalization, image channels, image target sizes, batch sizes, validation sizes, as well as regularization and early stopping techniques [16]. The data was then applied for training, validation, and testing, as illustrated in Fig. 2.

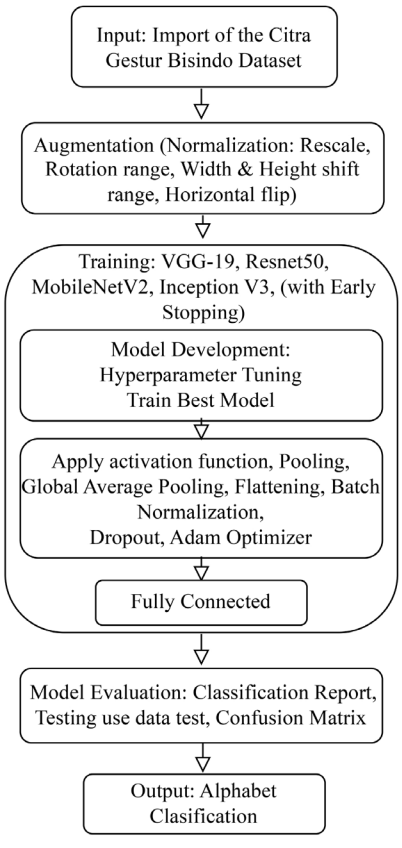

Fig. 1. Proposed methodology

```
train_datagen = ImageDataGenerator(
rescale = 1./255, #scalling
rotation_range = 15, # image rotation 15 degrees
width_shift_range = 0.2, # width shift 20% of image width
height_shift_range = 0.2, # height shift 20% of image height
horizontal_flip = True # rotate image horizontally
)
valid_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
#Data Training
train_generator = train_datagen.flow_from_directory(
train_directory,
target_size=(224,224), # Image input standards for All Architectures
batch_size = 32,
class_mode = "categorical"
)
valid_generator = valid_datagen.flow_from_directory(
valid_directory,
target_size=(224,224), # Image input standards for All Architectures
batch_size = 32,
class_mode = "categorical"
)
test_generator = test_datagen.flow_from_directory(
test_directory,
target_size=(224,224), # Image input standards for All Architectures
batch_size = 32,
class_mode = "categorical",
shuffle=False
)
```

Execution Result:

Found **7280 images** belonging to 26 classes(Training).
Found **2080 images** belonging to 26 classes(Validation).
Found **1040 images** belonging to 26 classes(Testing).

Fig. 2. Train, validation, and test code

B. Training model.

The images are labeled into 26 classes based on the alphabet A to Z. The dataset is then trained on 4 different architectures (VGG-19, MobileNetV2, ResNet-50 and Inception-V3). Then each architecture is fine-tuned and adjusts the final layer based on the dataset (output layer 26). To get the best model, hyperparameter tuning is performed on each architecture. The parameters selected for tuning are the number of nodes in the final layer, the number of dropouts, and the learning rate applied to each architecture [17].

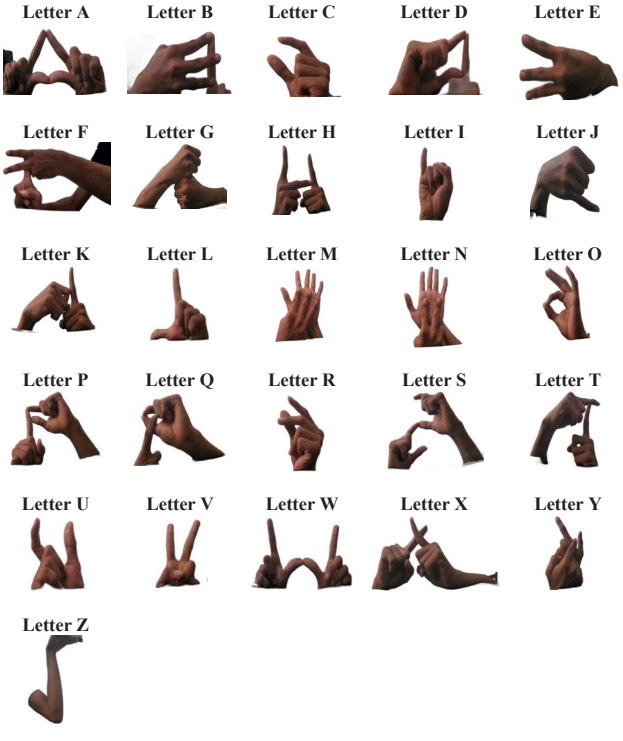

Fig. 3. Alphabet gesture images for 26 classes, A to Z

C. CNN.

CNN is a type of multi-layer artificial neural network. As shown in Fig. 4, CNN consists of four convolutional layers and, among other reduction layers, these layers are set alternately, and finally, a total connection layer is added, similar to a multi-layer perceptron network [18]. In the following, each layer is discussed.

D. Convolution layers.

The convolutional layer is the core of CNN; the size of the area mapped to the input image on the feature map of each level of the convolutional neural network is called the receptive field [19]. A larger receptive field can see a larger range of images, and more global and higher semantic level feature information can be extracted. On the contrary, a small receptive field can extract local features and details [20]. The features of the convolutional layer have hierarchical characteristics. Different convolutional layers have different semantic levels. Shallow convolutional layers respond to corner edges, and high-level convolutional layers respond more strongly to semantically similar regions [11].

E. Hyperparameter tuning.

Hyperparameter tuning: Hyperparameters-such as batch size, learning rate, number of epochs, dropout, and learning rate decay-are set. These hyperparameters play a critical role in the training process and must be carefully selected through experimentation and validation [7].

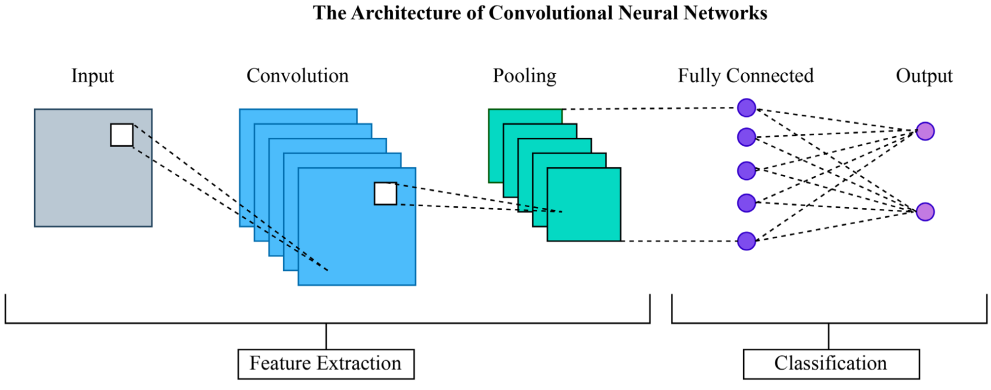

Fig. 4. The architecture of convolution neural networks

F. Model evaluation.

Model evaluation; Each model uses metrics such as accuracy, precision, recall, F1 score, receiver operating characteristic curve, and confusion matrix analysis. These metrics provide a comprehensive understanding of the model's performance, such as correctly classifying instances, handling imbalanced classes, and distinguishing between classes [7, 21].

G. Activation function.

An activation function (AF) is a function that returns the output generated by a neuron given an input. Each layer that makes up a neural network has an AF that allows prediction. AFs are divided into two types: linear and nonlinear. Linear functions allow the input data to be the same as the output data, and are also applied when linear regression is needed as the output. Meanwhile, nonlinear functions are applied when it is necessary to classify or when there is a categorical output [19]. Activation functions (e. g., ReLU) are applied to feature maps after convolution to add nonlinearity, helping the model capture complex patterns such as hand shapes in BISINDO gestures [22].

H. Pooling layer.

This layer is applied between two convolutional layers, as input, this layer receives the feature maps formed at the output of the convolutional layer; its main function is to reduce the image size while preserving the most resolving features. Finally, in the output of each Pooling layer, the same number of features are obtained as in the output, but significantly compressed. The most common pooling techniques used with different models are maximum pooling and average pooling. In addition, maximum pooling is used to create feature maps with reduced sampling. Average pooling is used to calculate the average value of the filter size. The application of these two pooling techniques provides the ability to learn invariant features and also acts as a regularizer to reduce the problem of overfitting. In addition, these techniques significantly reduce the computational cost and training time of the network, which are important criteria to consider [9].

I. Optimizer.

An optimizer is an algorithm or method used to update model parameters (such as weights and biases in a neural network) during training to minimize a loss function. The loss function measures how far the model's predictions are from the actual target, and the optimizer's job is to find the parameter values that produce the smallest loss. Adam (*adaptive moment estimation*) Optimizer functions to combine momentum and RMSProp by tracking the exponential average of the gradient (first momentum) and the square of the gradient (second momentum) [8].

J. Dropout.

Dropout is a regularization technique that randomly disables neurons during training to prevent overfitting and improve model generalization. In BISINDO's gesture recognition, dropout helps models like ResNet-50 or InceptionV3 learn robust gesture features, ensuring high accuracy on test data even though the dataset is limited. Dropout works with other components (convolution, fully connected layer, Adam optimizer) to produce a reliable model, as evaluated through the classification report and confusion matrix [23].

K. Fully connected layer.

The fully connected layer is a layer that combines features from previous layers to produce final predictions through linear and non-linear transformations. In BISINDO's gesture recognition, this layer converts visual features into alphabetical probabilities, playing a central role in model accuracy [9]. The design influences the trade-off between accuracy and efficiency, as evaluated in the comparison of VGG-19, ResNet-50, MobileNetV2, and InceptionV3. The fully connected layer relies on feature quality from convolution, pooling, and optimization by Adam for optimal performance [24].

L. Model architecture.

1.VGG19: the VGG19 model is similar to VGG16; both follow the same logic, with the difference that VGG19 has more layers, namely 19 layers, but the goal of both models is the same, namely to filter images by only storing discriminative information. The VGG19 model has five blocks, as shown in Fig. 5. The first two blocks contain two convolutional layers with filter sizes of 64 and 128; the middle block contains four convolutional layers with a filter size of 256, and the last two blocks contain two convolutional layers with a filter size of 512 each [7, 19].

2. MobileNetV2: MobileNetV2 is a mobile-embedded vision convolutional neural network. MobileNetV2 reduces computational cost and maintains accuracy using depth-separable convolution after 3×3 convolution. Depthwise separable convolutions use one filter per input channel and 1x1 pointwise convolutions to construct new features from the output [25]. This network-wide technique efficiently processes high-dimensional inputs. Point convolution expands the number of channels, depth convolution, and point convolution projects features back to lower dimensions in each block of inverted residuals with linear constraints [26]. A global average pooling layer reduces the spatial dimension to 1×1, a fully connected layer, and a softmax classification output completes the network. The economical architecture with minimal latency and high precision is suitable for resource-constrained systems [21]. The mobilnetv2 architecture model can be seen in Fig. 6.

3. ResNet50: Fig. 7 shows ResNet-50, a deep convolutional neural network that trains a deep network via residual learning. To reduce a 224 × 224 × 3 input image to 112 × 112, 7 × 7 convolutional layers (Conv1) with 64 filters are used [27]. The architecture consists of four main phases (Layer1–Layer4) with several remaining blocks [28]. These blocks simplify gradient backpropagation by bypassing convolutional layers with shortcut connections. Addressing the vanishing gradient problem allows deeper network training. Each layer deepens and shrinks the network. Layer1 has 64 filters, Layer2 128, Layer3 256, and Layer4 512, reducing the spatial dimensions to 7 × 7 [23]. The feature map is flattened into a 512-dimensional vector and classified using fully connected layers (Dense) and softmax after the last convolutional layer. With its processing depth and efficiency, ResNet-50 is a popular image categorization system [21].

4. InceptionV3: Fig. 8 illustrates Inception-v3, a deep image classification convolutional neural network. The input layer processes an image of size 299 × 299 × 3. The Inception modules (A, B, and C) in each stage provide parallel convolutional pipelines with 1 × 1, 3 × 3, 5 × 5 filter sizes and pooling operations for multiscale feature extraction.

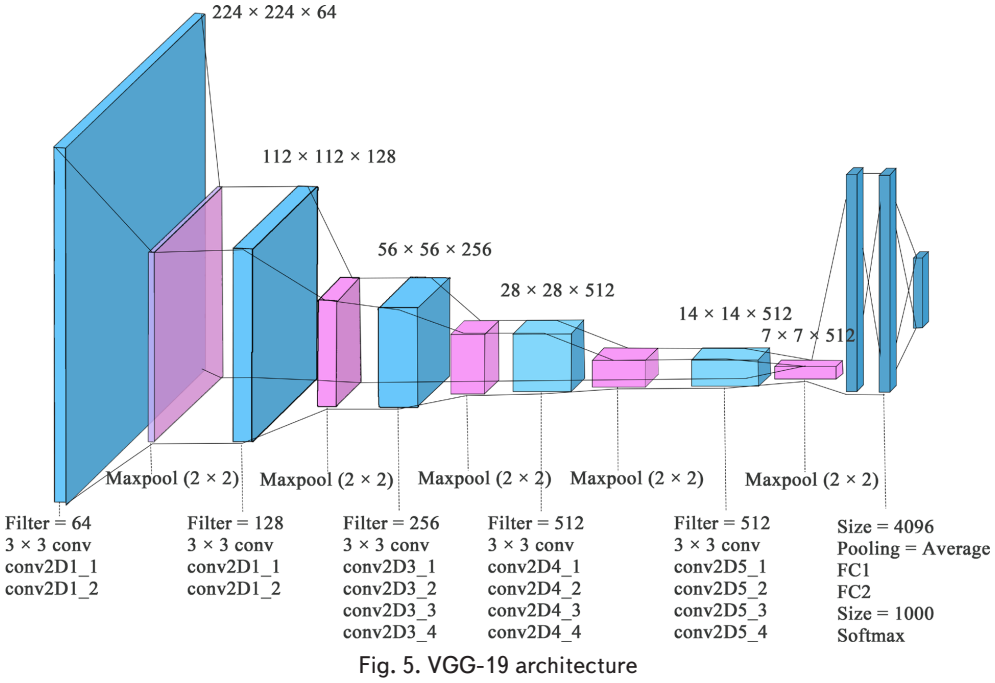

Fig. 5. VGG-19 architecture

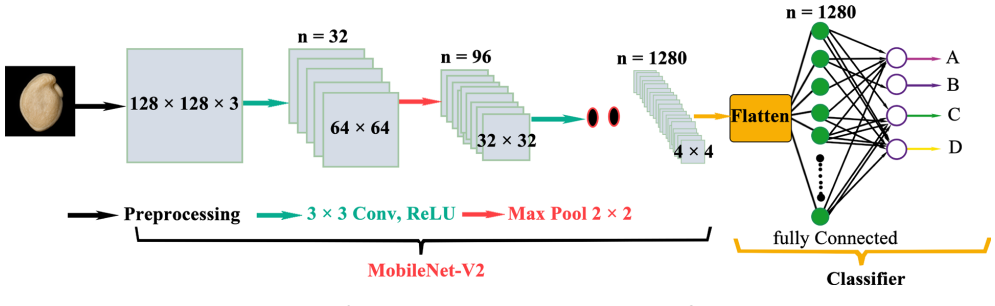

Fig. 6. The architecture of MobileNetV2

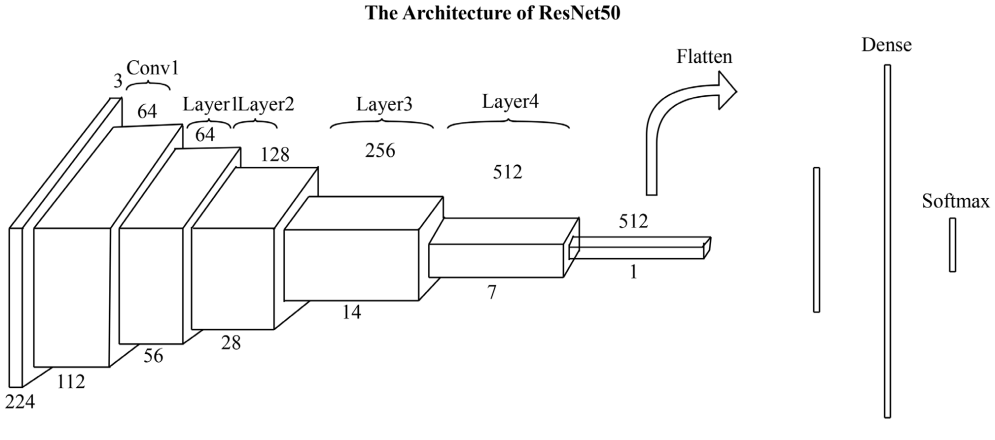

Fig. 7. The architecture of ResNet50

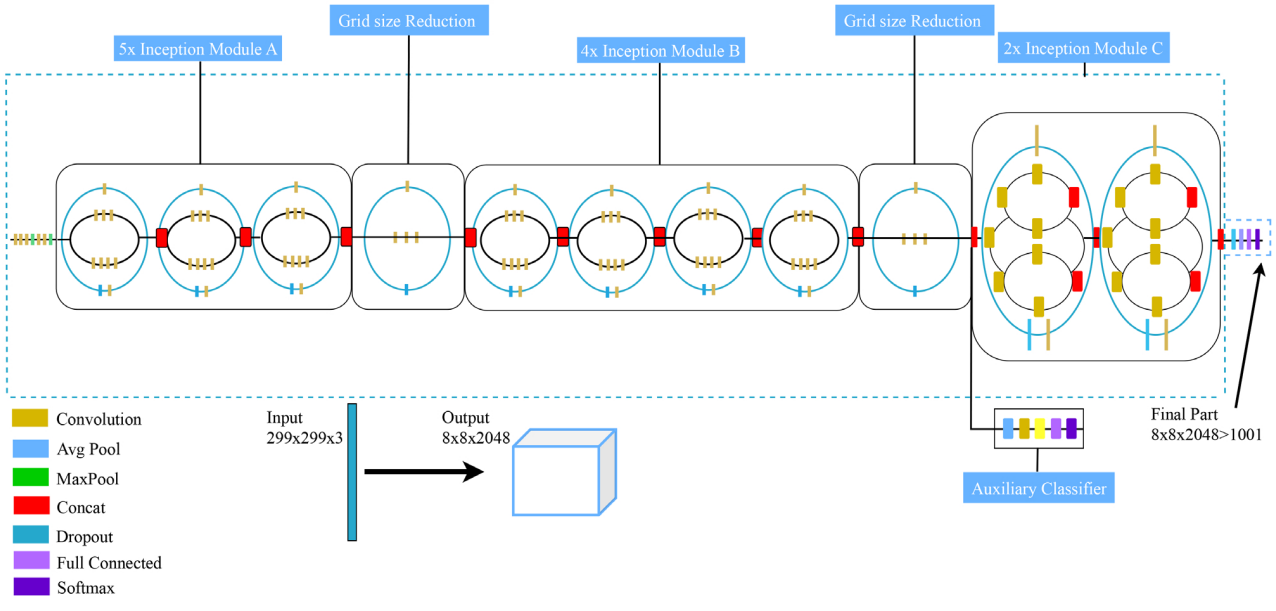

Fig. 8. InceptionV3 architecture

The Inception A module was repeated five times to gather finer details [29]. Reducing the grid size maintains a robust feature representation when down-sampling the feature map. Four applications of Inception Module B improve the learning of abstract features. After another grid size reduction, the Inception C Module refines the high-level features twice [30]. The architecture contains additional classifiers to train convergence. Finally, global average pooling, fully connected, and softmax classification layers are included. This setup allows efficient and deep networks to achieve high accuracy while managing computational complexity [31, 32].

Therefore, these four architectures will be used as models that will be evaluated and their performance compared from the training, validation to testing stages with the BISINDO alphabet image and the early stopping rule will also be used in the process according to the proposed methodology.

5. Results of the performance comparison of 4 CNN models for BISINDO alphabet gesture classification

5. 1. The workflow of each model (VGG-19, MobileNetV2, ResNet50 and InceptionV3) in predicted and classification of BISINDO alphabet gestures

The stages in the process of predicted and classification BISINDO alphabet gestures are divided into two, namely the data pre-processing stage which applies equally to all four models and the data processing stage for each model. To show the initial steps to the final steps for these four models, an example is taken using the image of the hand gesture of the letter "A".

The following starts from the data pre-processing stage which applies equally to all four models with the following steps:

– image input: an image of a hand gesture for the letter "A" in BISINDO, with an original resolution of 300 × 300 pixels, 3 RGB channels;

– resize and normalize: the image is resized to 224 × 224 pixels to fit the input architecture. The pixel values are normalized using ImageNet preprocessing (reducing the RGB mean, for example, (R: 123.68, G: 116.78, B: 103.94), resulting in an array of size (224, 224, 3);

– augmentation: for training, images may be rotated or re-illuminated, but for this example, it is possible to use the original processed images;

– input format: the images are converted to tensors with dimensions (1, 224, 224, 3) for batch size 1. After this, the data processing stage continues with an explanation of the stages for each architecture model.

The following explains the data processing stages with the workings of the VGG-19 architecture from the initial stage to the final stage:

– model initialization: VGG-19 (19 layers, 138 million parameters) is initialized with pre-trained weights from ImageNet. The last fully connected layer is replaced with a 26-neuron layer (for the alphabet A–Z) with softmax activation;

– feature extraction: The gesture image is processed through 16 convolution layers with 3 × 3 filters and 5 max pooling layers (2 × 2, stride 2). The convolution layers capture hierarchical features (finger edges to hand shapes), resulting in a 7 × 7 × 512 feature map after the last layer (block5_conv4);

– initial convolution layer: The image is processed through a "block1_conv1" layer (3 × 3 filters, 64 filters). The filters capture basic features such as the finger edges in the "A" gesture. Output: (224, 224, 64);

– first max pooling: The block1_pool layer (2 × 2 filters, stride 2) reduces the dimensionality to (112, 112, 64);

– next convolution layer: The process continues until "block5_conv4" (the last convolution layer), resulting in a (7, 7, 512) feature map after several convolution layers and max pooling. This feature includes complex patterns such as the hand shape for "A";

– last max pooling: The "block5" pool layer reduces the dimensionality to (3, 3, 512);

– flattening and classification: The feature map is flattened into a 1D vector (7 × 7 × 512 = 25,088 elements), then processed by a fully connected layer of 384 nodes with a dropout rate of 0.3 and a learning rate of 0.000099 (0.0001). The output layer produces probabilities for 26 classes (A–Z) using softmax;

– fully connected layer: This vector is processed by a fully connected layer, resulting in an output of 26 neurons (A–Z);

– initial training: The model is trained with the Adam optimizer, categorical cross-entropy loss function, dropout (0.3 until 0.7), and L2 regularization (coefficient 0.001). Initial accuracy is 99%;

– hyperparameter tuning: tuning is done on the learning rate (minimum = 0.00001, maximum = 0.001), dropout rate (minimum = 0.3, maximum = 0.7, stride 0.1), and L2 regularization. A small learning rate helps stable convergence, increasing accuracy to up to 100%;

– classification: to predict the class, the test image is processed through the trained model. The softmax output gives the probabilities for 26 classes (alphabet), and the class with the highest probability (e. g., "A") is selected as the prediction. The softmax output produces probabilities, for example: [0.92, 0.02, 0.01, ..., 0.01] (26 elements). The highest probability (0.92) is in class "A", so the gesture is predicted as the letter "A".

The working process of the MobileNetV2 architecture from the initial stage to the final stage is explained as follows:

– Model initialization: MobileNetV2 (3.5 million parameters) is initialized with ImageNet weights. The output layer is adjusted for 26 classes with softmax activation;

– feature extraction: MobileNetV2 uses depthwise separable convolution, which consists of depthwise convolution (3×3 filters per channel) and pointwise convolution (1×1 filter to combine channels). This architecture has an inverted residual block with shortcut connections, resulting in a $7 \times 7 \times 1024$ feature map before the final layer;

– initial layer: The image is processed through an initial convolution layer (3×3 filters, 32 filters). Output: (112, 112, 32) after stride 2;

– inverted residual blocks: using depthwise separable convolution (depthwise 3×3 and pointwise 1×1). The first block captures basic features such as the hand contour "A". The process continues through 17 blocks, producing a feature map of (7, 7, 1280) before the final layer;

– flattening and classification: The feature map is global average pooled into a 1D vector (1280 elements), then processed by a fully connected layer with an output of 26 classes (A–Z) using softmax;

– initial training: Trained with the Adam optimizer, categorical cross-entropy loss function, dropout, and L2 regularization, achieving 99% accuracy;

– hyperparameter tuning: Tuning the learning rate (0.00001 until 0.001) and dropout rate (0.3 until 0.7) improves the prediction probability distribution, despite high loss (19% in validation, 58% in testing);

– classification: The test image is processed, and the softmax output gives the probabilities for 26 classes. The class with the highest probability is selected as the prediction (e. g., "A"). Softmax output: [0.88, 0.03, 0.02, ..., 0.01]. The highest probability (0.88) is in class "A", so it is predicted as the letter "A".

The following is the working process of the ResNet50 architecture from the initial stage to the final stage explained as follows:

– model initialization: ResNet50 (50 layers, 25 million parameters) is initialized with ImageNet weights. The output layer is adjusted for 26 classes with softmax activation;

– feature extraction: using residual blocks with shortcut connections, each block has 1×1, 3×3, and 1×1 convolutions and batch normalization. The final feature map is $7 \times 7 \times 2048$ before global average pooling. After pooling, the feature map becomes $1 \times 1 \times 2048$;

– initial layers: the image is processed through initial convolution layers (7×7 filters, 64 filters, stride 2) and max pooling (3×3, stride 2). Output: (56, 56, 64);

– residual blocks: using residual blocks with shortcut connections (1×1, 3×3, 1×1 convolution). The first block captures the basic features of the gesture "A". The process continues through 50 layers, resulting in a feature map of (7, 7, 2048);

– flattening and classification: the feature map is global average pooled into a 1D vector (2048 elements), then processed by a fully connected layer with an output of 26 classes (A–Z) using softmax;

– initial training: trained with the Adam optimizer, categorical cross-entropy loss function, dropout, and L2 regularization, achieving 98% accuracy;

– hyperparameter tuning: tuning the learning rate (0.00001 until 0.001) and dropout rate (0.3 until 0.7) improves generalization;

– classification: the test images are processed, and the softmax output gives the probabilities for 26 classes. The class with the highest probability (e. g., "A") is selected as the prediction. Softmax output: [0.85, 0.04, 0.03, ..., 0.02]. The highest probability (0.85) is in class "A", so it is predicted as the letter "A".

And here is the work process of the InceptionV3 architecture from the initial stage to the final stage explained as follows:

– model initialization: InceptionV3 (24 million parameters) is initialized with ImageNet weights. The output layer is adjusted for 26 classes with softmax activation;

– feature extraction: using inception modules (1×1, 3×3, 5×5 convolutions in parallel) and factorized convolution. The final feature map is $7 \times 7 \times 2048$ before global average pooling;

– initial layer: the image is processed through an initial convolution layer (3×3 filters, 32 filters, stride 2). Output: (111, 111, 32);

– inception modules: using inception modules (1×1, 3×3, 5×5 convolutions in parallel). The first module captures the multiscale features of gesture "A". The process continues until the feature map is (5, 5, 2048);

– flattening and classification: the feature map is global average pooled into a 1D vector (2048 elements), then processed by a fully connected layer with an output of 26 classes (A–Z) using softmax;

– initial training: trained with the Adam optimizer, categorical cross-entropy loss function, dropout, and L2 regularization, achieving an accuracy of up to 99%. Hyperparameter Tuning: Tuning the learning rate (0.00001 until 0.001) and dropout rate (0.3 until 0.7) improves accuracy and stability;

– classification: the test image is processed, and the softmax output gives the probabilities for 26 classes. The class with the highest probability (e. g., "A") is selected as the prediction. Softmax output: [0.95, 0.01, 0.01, ..., 0.01]. The highest probability (0.95) is in class "A", so it is predicted as the letter "A".

5. 2. The results of comparing the accuracy performance of BISINDO gesture classification with confusion matrix

For the testing data prediction values, all models are tested using a confusion matrix, where predictions are synthesized and compared to actual values [3]. For example, Fig. 9, 10 show the confusion matrix of the VGG-19 and MobileNetV2 models correctly predicting all 40 images per alphabet. Assuming each class has 40 images in the test data (since total test data = 1.040 images, and 1040 images divided into 26 Classes resulting 40 images per class, this shows perfect accuracy. If each class has 40 images, the total number of correct predictions is

$$\text{Accuracy} = \\ = \left(\text{Correct Prediction/Number of Images}\right)\times 100\% = \\ = \frac{1040}{1040}\times 100\% = 100\%.$$

Fig. 11 shows that the model from the InceptionV3 architecture gives lower confusion matrix results compared to VGG-19 and MobileNetV2. Where in this model there are 4 images from 3 exception alphabet classes that do not match the prediction (1040 – 4 = 1036). So, the accuracy is calculated below

$$\text{Accuracy} = \\ = \left(\text{Correct Prediction/Number of Images}\right)\times 100\% = \\ = \frac{1036}{1040}\times 100\% = 99.61\%.$$

Furthermore, as seen in Fig. 12, it is very different from the three previous models. The ResNet50 model gives the worst confusion matrix results of the 4 models, where there are 18 images from 6 exception alphabet classes that do not match the predictions (1040 – 4 = 1036). So, the accuracy is calculated below

$$\text{Accuracy} = \\ = \left(\text{Correct Prediction/Number of Images}\right)\times 100\% = \\ = \frac{1022}{1040}\times 100\% = 98\%.$$

Recall or sensitivity to measure the quality of a machine learning model is very important in classification tasks. Recall, which is responsible for calculating the percentage of hits, is also known as sensitivity. F1 score allows combining Accuracy and recall into one weighted measure. If the F1 score is high, it means that false positives and false negatives are low [6]. A recapitulation of the comparison results of the confusion matrix test can also be seen in Table 1, where the column shaded in green indicates the highest value and the one shaded in red indicates the lowest value of the confusion matrix test variable from the 4 models.

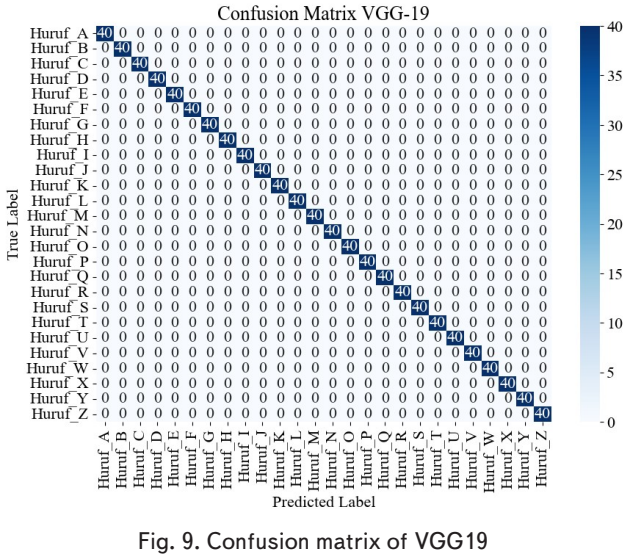

Fig. 9. Confusion matrix of VGG19

In Fig. 13 below, let's present the prediction results for the alphabet gesture image, because the results of the confusion matrix test state that the predictions of the 4 models have a percentage of above 98%, then the average prediction result for the alphabet gestures is stated to be correct and by the actual alphabet classification output.

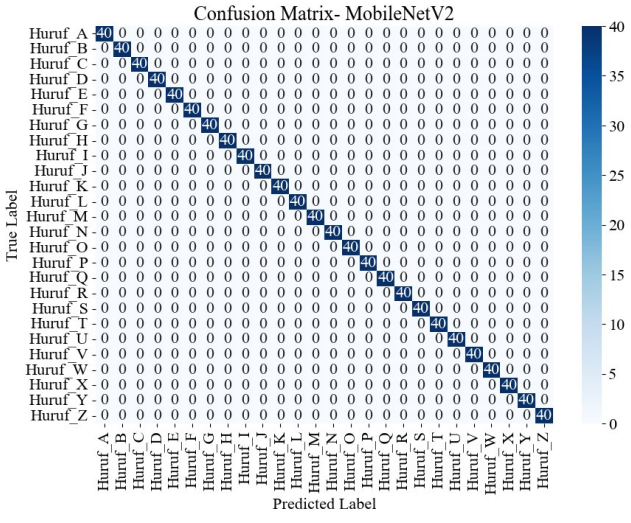

Fig. 10. Confusion matrix of MobileNetV2

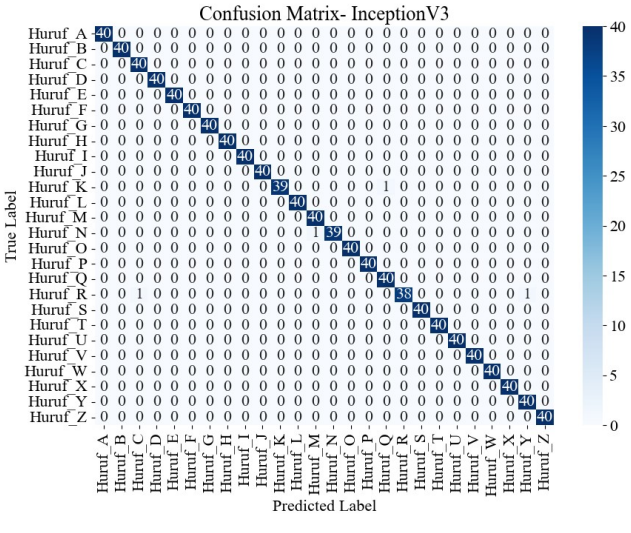

Fig. 11. Confusion matrix of InceptionV3

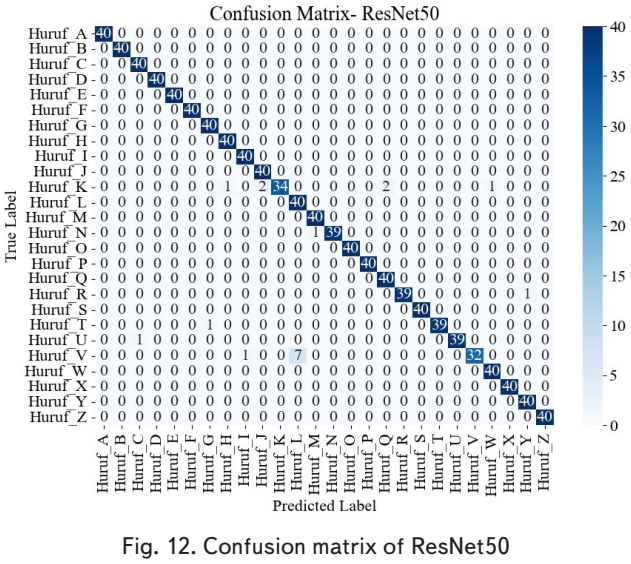

Fig. 12. Confusion matrix of ResNet50

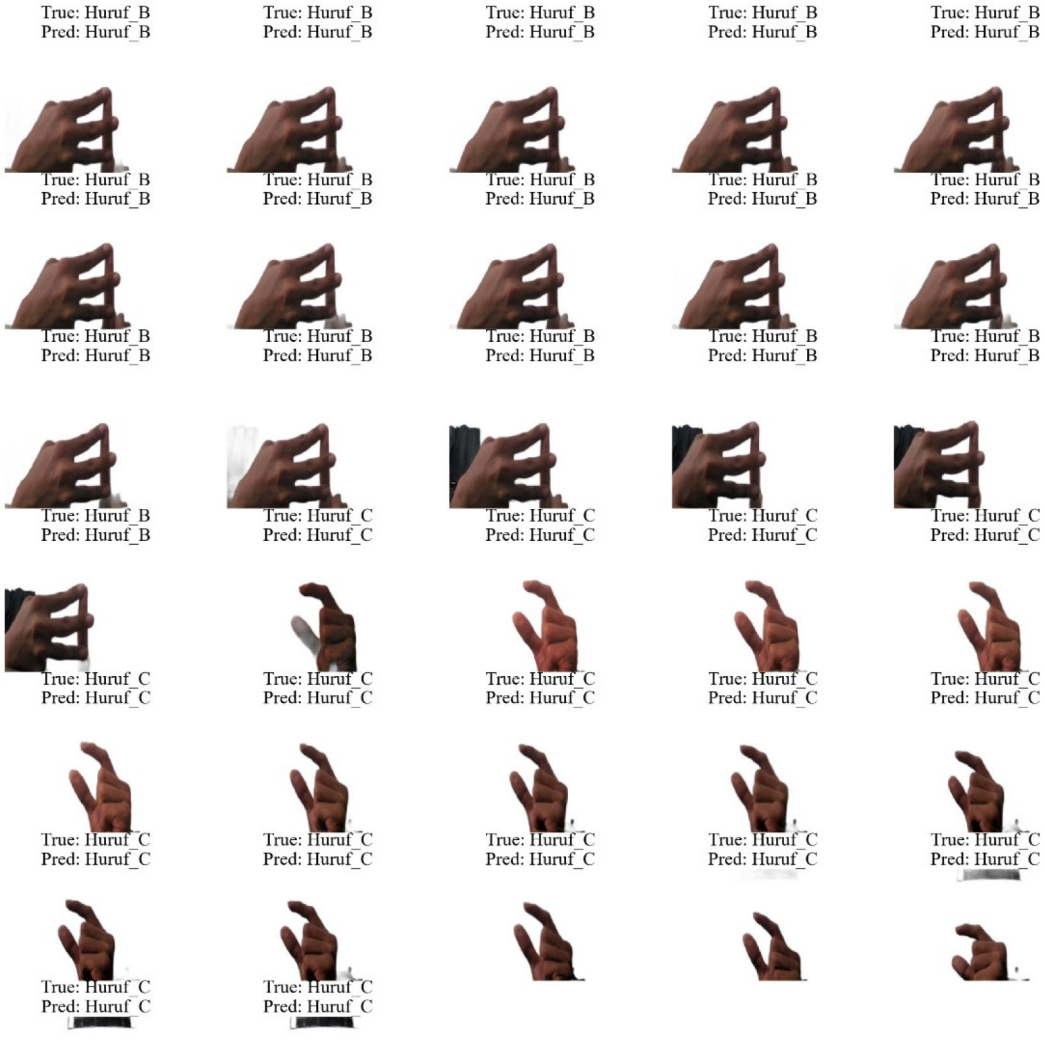

Fig. 13. Alphabet gesture image prediction results (*Prediction results are positioned above the image*)

5. 3. Results of comparing accuracy and loss values from training, validation, and testing

This section presents the results of four models that were trained, validated, and tested using a dataset of 7,280 Bisindo alphabet gesture images for training, 2,080 images for validation, and 1,040 images for testing. The same processing techniques, the same amount of data, the same number of hyperparameter partitions (learning rate and dropout) and the use of the same optimizer are used for all four models. Performance evaluation is measured by comparing the accuracy values of training, validation, and testing with the loss values. Time performance is also obtained, how quickly or how long the model can process data.

Graphs are also made to evaluate the performance of each model, along with the development of epochs for accuracy values and missing values in the training and validation processes. Fig. 14 displays the training set accuracy performance curve for each model. It can be seen that the MobileNetV2 model tends to be stable from 1st epoch, the network converges, and its performance is stable. Likewise, with VGG-19, although its performance seems to fluctuate slightly, it looks stable and converging from 1st epoch. Inception-V3 seems to converge after the 2nd network epoch, and the convergence speed is slightly lower than VGG-19 and MobileNetV2. The ResNet50 model doesn't seem to get maximum performance and doesn't seem to converge, but it's still above 80%. The rate of convergence is the slowest of all models.

Fig. 15 shows the accuracy performance curve for each model on the validation set. The convergence performance of MobileNetV2, VGG19 and Inception-V3 on the validation set is not much different, and the network convergence speed is much higher than that of RessNet50. After 1st epoch on both the training set and validation set, the accuracy can reach above 99%, but on the training and validation set, MobileNetV2 still looks very stable. The accuracy of the training set and validation set of MobileNetV2, VGG19 and Inception-V3 is almost synchronous. The accuracy of the RessNet50 validation set has a level of performance that improves with increasing epochs but still looks below 90%, and the convergence is not as visible as the previous 3 models.

The graph in Fig. 16 shows the loss performance curve of each model on the training set, and the graph in Fig. 17 shows the loss performance curve of each model on the validation set. Fig. 16, 17 shows that the loss performance of VGG-19, MobileNet and Inception V3 decreases synchronously on the training set and validation set, and all look stable, which proves that the number of iterations of the epoch network is quite stable. However, of the 3 models, VGG-19 seems to converge with a loss below 30% for training and success under 20% during validation. For inception V3, although the loss performance decreases steadily, the percentage is relatively

high, namely above 60% in both the training and validation sets. MobileNetV2 is better than Inception V3 with loss performance below 60% but not as low as VGG-19 loss convergence for both the training set and validation set. In contrast to ResNet-50 for the training set, the loss percentage is still above 50%, but on the validation set the loss value improves to a value below 40%. However, the convergence loss performance is unstable, so it is said that the training and validation sets are not as good as the previous 3 models.

So, from the results of the accuracy and loss performance on the training data and validation data based on the explanation above, it can be concluded that the model with the VGG-19 architecture shows the best stability. because the resulting loss value is below 20%, with an accuracy of almost 100%. MobileNetV2 is the most reliable in terms of accuracy, but the loss value is still around and above 20%. And the other 2 models still provide relatively high loss performance even though the accuracy performance obtained is above 90%. So, from this it is possible to understand that the InceptionV3 and ResNet50 architectures are still vulnerable to this BISINDO alphabet gesture prediction, especially if later using more real test data.

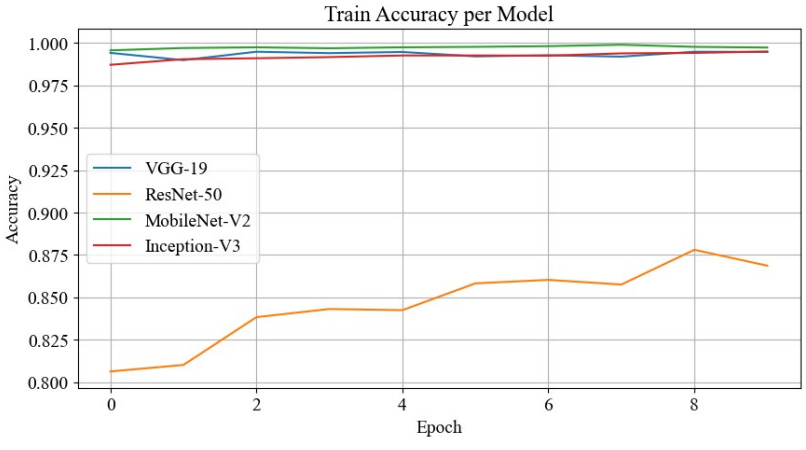

Fig. 14. Training accuracy performance comparison of VGG19, MobileNetV2, ResNet50 and InceptionV3

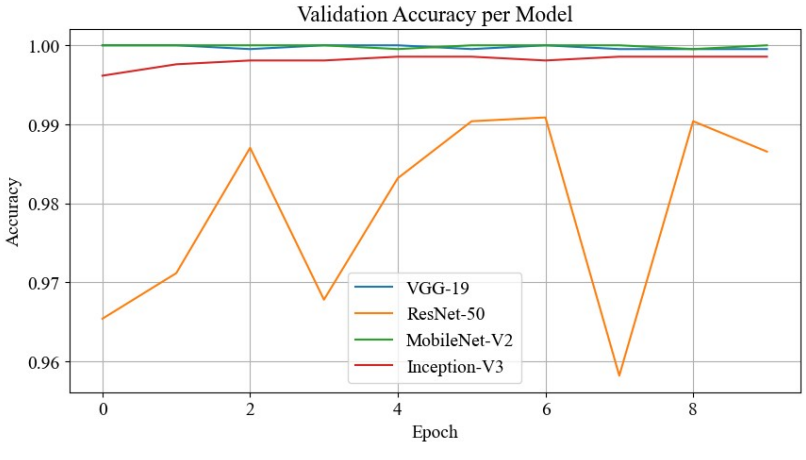

Fig. 15. Validation accuracy performance comparison of VGG19, MobileNetV2, ResNet50 and InceptionV3

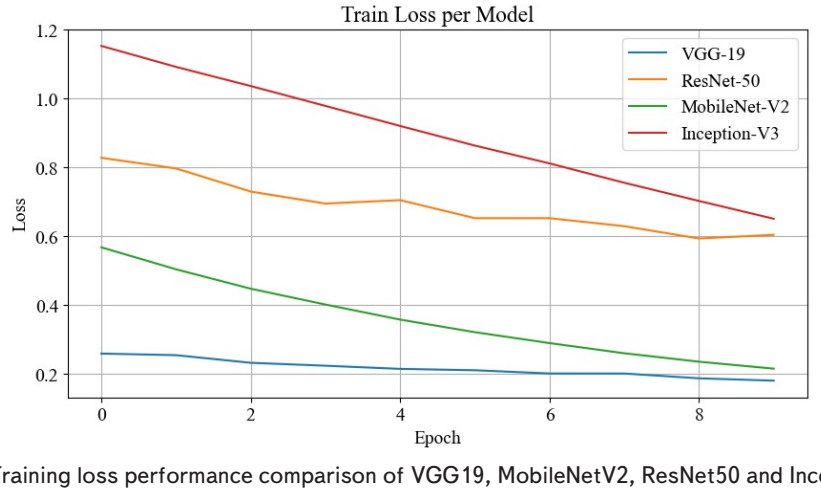

Fig. 16. Training loss performance comparison of VGG19, MobileNetV2, ResNet50 and InceptionV3

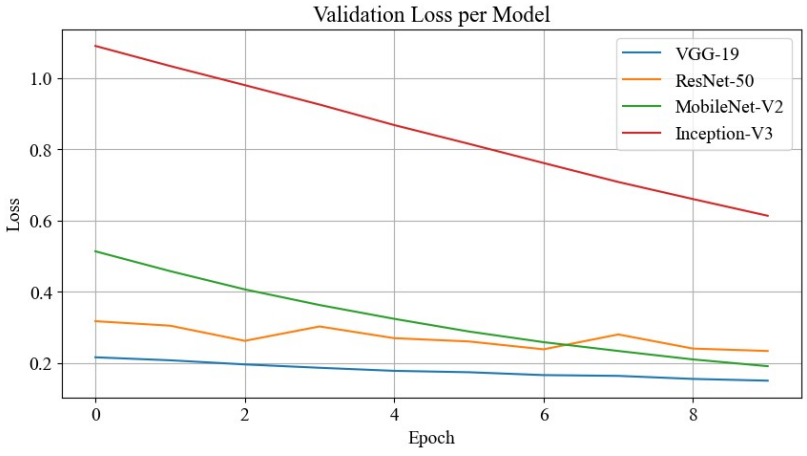

Fig. 17. Validation loss performance comparison of VGG19, MobileNetV2, ResNet50 and Inception-V3

5. 4. Overall performance comparison results of VGG-19, ResNet-50, MobileNetV2 and Inception-V3 models

Table 1 shows the recapitulation results of the 4 model tests, consisting of accuracy, recall, F1 score and precision values based on the confusion matrix results. It is shown that there are two models that get perfect prediction results of 100%, namely the VGG-19 and MobileNetV2 models with overall accuracy, recall, F1 score and precision values, which are also both 100%. For the other two models, due to prediction errors from 4 images in 3 alphabet classes (K, N, R) in InceptionV3, the accuracy is 99% with the same F1 score and precision, slightly different for recall which is 98% and prediction errors from 18 images in 6 alphabet classes (K, N, R, T, U, V) in ResNet50 make this model get the lowest accuracy, recall, F1 score and precision performance, namely 98% even though the value of 98% itself is considered to provide very good performance results.

Table 2 shows that from the training data, the highest accuracy performance was obtained by the MobileNetV2 model: 99.89%, while the lowest was by the ResNet50 model: 87.80%. For the loss value, the VGG19 model got the lowest value, namely 18.79%, with the highest being the InceptionV3 model, namely 65%. So, in this training data, the best performance can be concluded to be VGG-19.

For validation data, the highest accuracy performance is achieved by two models, VGG-19 and MobileNetV2, which reach a value of 100%. The lowest accuracy is achieved by ResNet, with a slight difference of 99%. The lowest loss performance is achieved by VGG-19, with 16.59%, and the highest is Inception-V3, at 61%. For validation data, it is also possible to observe the time performance, with Mobilenetv2 being the fastest at 38 seconds and VGG-19 being the longest at 310 seconds. So, it can be concluded that the best validation data performance is MobileNetV2. Next, for the test data, the highest accuracy performance of the 2 models is again VGG-19 and MobileNetV2, which are able to reach a value of 100%, and the lowest is ResNet50 with a slight decrease of 98%. The lowest loss performance is VGG-19, and the highest is Inception-V3, which reaches a value of 114%. For this test performance, it is also possible to see from the time performance that the fastest is on MobileNetV2, which is only 20 seconds, and the longest is on VGG-19, reaching 157 seconds. So, it can be concluded that the best test data performance is MobileNetV2, with the exception of the loss value and VGG-19, with the exception of the time value.

However, the almost perfect performance results of the four models require further validation with external test data under different conditions to ensure durability in the real world.

Table 1

Comparison of VGG-19, MobileNetV2, ResNet50 and InceptionV3 with confusion matrix

Model/Co mp Var	Confusion matrix			
	Accuracy	F1 Score	Recall	Precision
VGG-19	100%	100%	100%	100%
MobileNetV2	100%	100%	100%	100%
ResNet50	98%	98%	98%	98%
InceptionV3	99%	99%	98%	99%

Table 2

Comparison of VGG-19, MobileNetV2, ResNet50 and InceptionV3 for training, validation and testing performance

Model/Comp Var	Training		Validation			Testing		
	Accuracy	Loss	Accuracy	Loss	Time (seconds)	Accuracy	Loss	Time (seconds)
VGG-19	99.48%	18.79%	100%	16.59%	310s	100%	30%	157s
MobileNetV2	99.89%	26%	100%	19%	38s	100%	58%	20 s
ResNet50	87.80%	59%	99%	23.80%	118s	98%	30%	66s
InceptionV3	99.49%	65%	99.86%	61%	133s	99.60%	114%	67s

6. Discussion of the performance comparison of 4 CNN models for BISINDO alphabet gesture classification

Performance comparison in detecting BISINDO alphabet gestures using 4 CNN models: VGG-19, MobileNetV2, ResNet50 and InceptionV3 all use 7280 images for training, 2080 for validation and 1040 for testing, which are divided into 26 alphabet classes (Fig. 2). In this case, the number of data sets (images) that have been cleaned of the background can significantly increase the level of accuracy compared to ordinary normal images. The BISINDO gesture image is pre-processed (resize 224 × 224,

normalization), then its features are extracted by VGG-19 through 16 convolution layers (3×3 filters) to a feature map of $7 \times 7 \times 512$, MobileNetV2 with depthwise separable convolution up to $7 \times 7 \times 1280$, ResNet50 using residual blocks up to $7 \times 7 \times 2048$, and InceptionV3 with inception modules up to $5 \times 5 \times 2048$. This feature map is flattened or global average pooling, processed by a fully connected layer, and classified with softmax to produce a probability of 26 classes. Hyperparameter tuning (learning rate 0.00001–0.001, dropout 0.3–0.7) which functions to improve performance.

In the test results using the confusion matrix for VGG-19 in Fig. 9, result of MobileNetV2 in Fig. 10, result of ResNet50 in Fig. 11 and result of InceptionV3 in Fig. 12 it can be seen that alphabet prediction using the VGG-19 and MobileNetV2 architecture models can produce a perfect value of 100%, followed by InceptionV3 with 99% and ResNet50 with 98%. This shows that the results obtained are a very good achievement in overall performance besides considering the complexity that occurs in predicting finger gesture images, which often have features with various conditions.

The results of the confusion matrix are also summarized in Table 1, which can explain that the four models show accuracy above 98%, F1 score above 98%, precision above 98%, and recall above 98%, which shows very superior and consistent performance in recognizing BISINDO gestures. Specifically, accuracy above 98% indicates that almost all gestures (98 out of 100) are correctly classified by the model, reflecting the ability of the four models to capture complex gesture features. Precision above 98% indicates that when the model predicts a gesture as a particular letter (e. g., "A"), above 98% of the predictions are correct, making it very reliable to minimize prediction errors. Recall above 98% indicates that the model is able to detect above 98% of all correct gesture instances for each letter, ensuring that almost no gestures are missed. F1 Score above 98% illustrates the perfect balance between precision and recall, confirming that the four models are not only accurate but also consistent in their performance. However, with accuracy, precision, recall and F1 score reaching 100%, making VGG-19 and MobileNetV2 a very reliable choice for BISINDO gesture recognition. With very high metrics, VGG-19 and MobileNetV2 show that these models are able to distinguish similar gestures (e. g., the letters "B" and "D" which are often confused) with no error rate (seen in the prediction results in Fig. 13), making them a powerful solution for future implementation in practical applications such as real-time gesture translation to support communication for the deaf community. This also answers previous research [2, 4] with the model only being able to produce performance below 90% for predicting BISINDO alphabet gestures. And in research [3], the model was only able to achieve an accuracy performance of 92.8%, which in fact is still below the performance of the four models studied.

For the results of the analysis of accuracy through comparative experiments in the training and validation process. Fig. 14 can discuss the training set accuracy performance curve for each model. MobileNetV2 and VGG-19 tend to be stable from the first epoch, the network converges, and its performance is stable. Inception-V3 appears to converge after the 2nd epoch but when it converges it is able to stabilize with an accuracy above 95%. The ResNet50 model does not get maximum performance with the slowest convergence rate of all models but still produces an accuracy level above 80%. Fig. 15 shows the accuracy performance curve for each model on the validation set. The convergence performance of MobileNetV2, VGG19, and Inception-V3 is not much different. After the 1st epoch the accuracy can reach 99%. The accuracy of the validation set on RessNet50 has an increased performance level but is still seen below 95% and convergence and stability are not seen like the other 3 models. In the end, the results of the analysis of the accuracy values of the training and validation sets are that MobileNetV2 is very stable. Meanwhile, the results of the loss performance analysis in the training and validation process. Fig. 16 shows the loss performance for the training set and Fig. 17 for the validation set. For both sets, the loss performance of VGG-19, MobileNet, and Inception V3 decreased simultaneously, which proves that the number of epoch network iterations is quite stable. Different from that shown by ResNet50 which tends to fluctuate (unstable). VGG-19 appears more convergent with a loss below 30% for training and success below 20% during validation. Inception V3, although the loss performance decreases steadily, the percentage is relatively high (above 60%) in both the training and validation sets. MobileNetV2 is better than Inception V3 with a loss performance below 60%. ResNet-50 for the training set, the loss performance is still above 50%, but in the validation set the loss value improves to a value below 40%. So, it is concluded that for this loss performance, VGG-19 has the best performance.

For efficiency performance in Table 2, it can also be seen that further model refinement greatly improves training efficiency and reduces the need for computing resources. As in the MobileNetV2 model, where the validation and testing process only takes less than 40 seconds (38 seconds and 20 seconds) but the VGG-19 model still provides the same high accuracy performance but the time required is longer than MobileNetV2, InceptionV3 and ResNet50 (Validation: 310 seconds and testing 157 seconds). So, for efficiency performance and process speed, the MobileNetV2 model is the most superior.

From the results of this overall comparison, it can be concluded that the best recommended model is MobilenetV2 which is slightly superior to VGG-19 due to the smaller number of parameters.

The distinction offered in this study is the high performance produced by the MobileNetV2, VGG-19 and Inception V3 models is influenced by the use of hyperparameter tuning, where the configuration of the learning rate range for model training allows the search for optimal hyperparameters by setting a minimum learning rate value of 0.00001 and a maximum value of 0.001. This range was chosen to ensure stable and efficient convergence, while handling the complexity of the BISINDO gesture, with small values for fine tuning and larger values for fast initial learning, which supports high model performance. The use of the Adam optimizer also helps to adjust the learning rate adaptively. This range gives Adam the flexibility to adjust the learning step within safe limits, preventing divergence (if the learning rate is too large) or convergence too slow (if it is too small). Dropout value setting is also required to enable optimal hyperparameter search in the range of minimum 0.3 to maximum 0.7 with step 0.1. This range is chosen to provide balanced regularization, preventing overfitting on the complex BISINDO dataset. Furthermore, support for the use of L2 regularization with a coefficient of 0.001 to add a penalty to the model weights, aims to prevent overfitting by keeping the weights small. The value of 0.001 provides balanced regularization, supporting the high performance of the model in BISINDO alphabet

gesture classification. However, the almost perfect performance results of the four models require further validation with external test data under different conditions to ensure robustness in the real world.

This study has several advantages over previous similar studies, namely: a comprehensive comparison of 4 VGG-19 architecture models, MobileNetV2, ResNet50 and InceptionV3, focusing on BISINDO as a local sign language, the best efficiency performance provided by MobileNetV2 can be applied to practical applications, the use of modern optimization techniques, systematic hyperparameter tuning, contributions to social inclusion, and detailed evaluation metrics. These advantages make this study an important step in the development of BISINDO gesture recognition technology, with great potential to support communication for the Deaf community in Indonesia. This study also provides a strong foundation for further development, such as expansion into dynamic gestures or the forerunner of integration with mobile devices.

Of course, this study is not free from limitations and weaknesses, including several limitations in terms of dataset, gesture coverage, computing resources, hyperparameter tuning, and model generalization. The disadvantages include low efficiency, limited representation of BISINDO culture, varying accuracy, and high time and cost. To address this, future research can focus on collecting larger datasets, using efficient models, expanding gesture coverage, optimizing hyperparameters, real-world testing, and collaborating with the Deaf community. These steps will result in a more robust and practical BISINDO gesture classification system.

Potential developments of this study include expansion to dynamic gestures, optimization for mobile devices, increasing robustness, multimodal integration, and building a public dataset. However, these processes will face mathematical (model complexity), methodological (dataset collection), experimental (real-world testing), computational (complex model), socio-cultural (collaboration with the community), and ethical (privacy and bias) difficulties. With solutions such as transfer learning, data augmentation, cultural collaboration, and fairness techniques, these developments can result in a more inclusive, robust, and practical BISINDO recognition system to support the Deaf community in Indonesia.

7. Conclusion

1. The workflow of each model in predicting and classifying BISINDO alphabet gestures (A-Z) is summarized as follows: The gesture image is processed through pre-processing (resize 224×224, normalization), then its features are extracted by VGG-19 using a layered 3×3 filter up to a $7 \times 7 \times 512$ feature map, MobileNetV2 with depthwise separable convolution up to $7 \times 7 \times 1280$, ResNet50 through residual blocks up to $7 \times 7 \times 2048$, and InceptionV3 with inception modules up to $5 \times 5 \times 2048$. The feature map is flattened or global average pooling, processed by a fully connected layer, and classified with softmax to produce 26 class probabilities. Hyperparameter tuning (learning rate 0.00001–0.001, dropout 0.3–0.7) improves accuracy, with MobileNetV2 and VGG-19 reaching 100% (accurate and efficient), Inception-V3 up to 99% and ResNet50 reaching 98%, enabling accurate BISINDO gesture prediction for Deaf community communication.

2. Comparing the accuracy performance of BISINDO gesture classification using four deep learning models (VGG-19, ResNet-50, MobileNetV2, and Inception-V3) to identify the most accurate model, the confusion matrix results show that MobileNetV2 and VGG-19 achieve the highest accuracy of 100% on the test data, followed by Inception-V3 with 99% accuracy, and ResNet-50 with 98%. In certain scenarios, Inception-V3 even achieves 99% accuracy, precision, recall, and F1 Score, demonstrating its excellent ability in recognizing various BISINDO gestures, mainly due to the inception modules that capture features at various scales.

3. The accuracy and loss performance on the training and validation data can be concluded that the model with the VGG-19 architecture shows the best stability. because the resulting loss value is below 20%, with an accuracy of almost 100%. MobileNetV2 is the most reliable in terms of accuracy, but its loss value is still around 20% and above. And the other 2 models still provide relatively high loss performance even though the accuracy performance obtained is above 90%. So the InceptionV3 and ResNet50 architectures are still vulnerable to this BISINDO alphabet gesture prediction, especially when using real test data.

4. Based on the overall performance and computational efficiency of each model, MobileNetV2 and VGG-19 are identified as the best models with 100% accuracy and high computational efficiency, making them ideal solutions for practical applications. However, MobilenetV2 is slightly superior to VGG-19 due to the smaller number of parameters. Meanwhile, Inception-V3 offers very high accuracy (up to 99%), but its computational requirements are larger, making it more suitable for scenarios that prioritize maximum accuracy. In contrast, ResNet-50 is considered the worst among the 3 models above due to its low efficiency and tendency to overfit on the limited BISINDO dataset. In terms of efficiency, MobileNetV2 is proven to be the most efficient where the time required for the validation and testing process is no more than 40 seconds, much faster than VGG-19 (310 and 157 seconds), ResNet-50 (118 seconds and 66 seconds), and Inception-V3 (133 seconds and 67 seconds). This success is supported by the use of optimization techniques such as Adam optimizer, dropout (range 0.3 to 0.7), L2 regularization (coefficient 0.001), and hyperparameter tuning (e.g. learning rate 0.00001 to 0.001), which ensure the model is not only accurate but also able to generalize well. Thus, this study provides a strong foundation for the development of an inclusive BISINDO gesture recognition system, with MobileNetV2 as the main choice for real-time applications that are accessible to the Deaf community in Indonesia.

Conflict of interest

The authors declare that they have no conflicts of interest, financial, personal, authorial or otherwise, that could have influenced the study and the results presented in this paper.

Financing

The study was performed without financial support.

Data availability

The manuscript has associated data in a data repository.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgments

The authors wish to thank all the editors and anonymous reviewers for their constructive advice.

References

1. Types of Hearing Loss. American Speech-Language-Hearing Association. Available at: https://www.asha.org/public/hearing/Types-of-Hearing-Loss
2. Altiarika, E., Sari, W. P. (2023). Pengembangan Deteksi Realtime untuk Bahasa Isyarat Indonesia dengan Menggunakan Metode Deep Learning Long Short Term Memory dan Convolutional Neural Network. Jurnal Teknologi Informatika Dan Komputer, 9 (1), 1–13. https://doi.org/10.37012/jtik.v9i1.1272
3. Caraka, R. E., Supardi, K., Kurniawan, R., Kim, Y., Gio, P. U., Yuniarto, B., Mubarok, F. Z., Pardamean, B. (2024). Empowering deaf communication: a novel LSTM model for recognizing Indonesian sign language. Universal Access in the Information Society, 24 (1), 771–783. https://doi.org/10.1007/s10209-024-01095-1
4. Aziz, A. N. (2021). Image Recognition Alfabet Bahasa Isyarat Indonesia (Bisindo) Menggunakan Metode Convolutional Neural Network. Yogyakarta. Available at: https://dspace.uii.ac.id/handle/123456789/32137
5. Shams, M. Y., Hassan, E., Gamil, S., Ibrahim, A., Gabr, E., Gamal, S. et al. (2025). Skin Disease Classification: A Comparison of ResNet50, MobileNet, and Efficient-B0. Journal of Current Multidisciplinary Research, 1 (1), 1–7. https://doi.org/10.21608/jcmr.2025.327880.1002
6. Reka, S. S., Murthy Voona, V. D., Sai Nithish, P. V., Paavan Kumar, D. S., Venugopal, P., Ravi, V. (2023). Performance Analysis of Deep Convolutional Network Architectures for Classification of Over-Volume Vehicles. Applied Sciences, 13 (4), 2549. https://doi.org/10.3390/app13042549
7. Younis, H., Obaid, M. (2024). Performance Comparison of Pretrained Deep Learning Models for Landfill Waste Classification. International Journal of Advanced Computer Science and Applications, 15 (11). https://doi.org/10.14569/ijacsa.2024.0151166
8. Sanjaya, S. A., Faustine Ilone, H. (2023). BISINDO Sign Language Recognition: A Systematic Literature Review of Deep Learning Techniques for Image Processing. Indonesian Journal of Computer Science, 12 (6). https://doi.org/10.33022/ijcs.v12i6.3539
9. Aryananda, I. G. A. O., Samopa, F. (2024). Comparison of the Accuracy of The Bahasa Isyarat Indonesia (BISINDO) Detection System Using CNN and RNN Algorithm for Implementation on Android. MALCOM: Indonesian Journal of Machine Learning and Computer Science, 4 (3), 1111–1119. https://doi.org/10.57152/malcom.v4i3.1465
10. Pin, K., Ho Chang, J., Nam, Y. (2022). Comparative Study of Transfer Learning Models for Retinal Disease Diagnosis from Fundus Images. Computers, Materials & Continua, 70 (3), 5821–5834. https://doi.org/10.32604/cmc.2022.021943
11. Cui, S., Su, Y. L., Duan, K., Liu, Y. (2022). Maize leaf disease classification using CBAM and lightweight Autoencoder network. Journal of Ambient Intelligence and Humanized Computing, 14 (6), 7297–7307. https://doi.org/10.1007/s12652-022-04438-z
12. Kumar, J. S., Anuar, S., Hassan, N. H. (2022). Transfer Learning based Performance Comparison of the Pre-Trained Deep Neural Networks. International Journal of Advanced Computer Science and Applications, 13 (1). https://doi.org/10.14569/ijacsa.2022.0130193
13. Mendes, J., Lima, J., Costa, L., Rodrigues, N., Pereira, A. I. (2024). Deep learning networks for olive cultivar identification: A comprehensive analysis of convolutional neural networks. Smart Agricultural Technology, 8, 100470. https://doi.org/10.1016/j.atech.2024.100470
14. Wahyuningsih, W., Nugraha, G. S., Dwiyansaputra, R. (2024). Classification Of Dental Caries Disease In Tooth Images Using A Comparison Of Efficientnet-B0, Mobilenetv2, Resnet-50, Inceptionv3 Architectures. Jurnal Teknik Informatika (Jutif), 5 (4), 177–185. https://doi.org/10.52436/1.jutif.2024.5.4.2187
15. Candra, A., Rosmalinda, Intan, T. K., Purnamasari, F., Liyanto, H., Nugraha, A. T., Ewaldo. (2024). Development of machine learning-based sign language translator for Bahasa Isyarat Indonesia (BISINDO). Proceedings Of The 6th International Conference On Computing And Applied Informatics 2022, 2987, 020070. https://doi.org/10.1063/5.0199747
16. Harumy, T. H. F., Br Ginting, D. S., Manik, F. Y., Alkhowarizmi, A. (2024). Developing an early detection model for skin diseases using a hybrid deep neural network to enhance health independence in coastal communities. Eastern-European Journal of Enterprise Technologies, 6 (9 (132)), 71–85. https://doi.org/10.15587/1729-4061.2024.313983
17. Díaz-Gaxiola, E., Morales-Casas, Z. E., Castro-López, O., Beltrán-Gutiérrez, G., Vega-López, I. F., Yee-Rendón, A. (2019). Estudio comparativo de arquitecturas de CNNs en hojas de Pimiento Morrón infectadas con virus PHYVV o PEPGMV. Research in Computing Science, 148 (7), 289–303. https://doi.org/10.13053/rcs-148-7-22
18. Kunjachan, S., Remya, R., Kumar, S., Asish, G. R., Sheela, K., Kala, S. (2023). Comparative study of convolutional neural networks for leaf classification in Ayurveda. IET Conference Proceedings, 2023 (11), 18–23. https://doi.org/10.1049/icp.2023.1756
19. Iparraguirre-Villanueva, O., Guevara-Ponce, V., Paredes, O. R., Sierra-Liñan, F., Zapata-Paulini, J., Cabanillas-Carbonell, M. (2022). Convolutional Neural Networks with Transfer Learning for Pneumonia Detection. International Journal of Advanced Computer Science and Applications, 13 (9). https://doi.org/10.14569/ijacsa.2022.0130963
20. Panthakkan, A., Anzar, S. M., Al Mansoori, S., Mansoor, W., Al Ahmad, H. (2022). A systematic comparison of transfer learning models for COVID-19 prediction. Intelligent Decision Technologies, 16 (3), 557–574. https://doi.org/10.3233/idt-220017

21. Ahmed, N., Rahman, M., Ishrak, F., Joy, I. K., Sabuj, S. H., Rahman, S. (2024). Comparative Performance Analysis of Transformer-Based Pre-Trained Models for Detecting Keratoconus Disease. arXiv. https://doi.org/10.48550/arXiv.2408.09005
22. Ahmad, N., Wijaya, E. S., Tjoaquinn, C., Lucky, H., Iswanto, I. A. (2023). Transforming Sign Language using CNN Approach based on BISINDO Dataset. 2023 International Conference on Informatics, Multimedia, Cyber and Informations System (ICIMCIS), 543–548. https://doi.org/10.1109/icimcis60089.2023.10349011
23. Hossain, Md. B., Iqbal, S. M. H. S., Islam, Md. M., Akhtar, Md. N., Sarker, I. H. (2022). Transfer learning with fine-tuned deep CNN ResNet50 model for classifying COVID-19 from chest X-ray images. Informatics in Medicine Unlocked, 30, 100916. https://doi.org/10.1016/j.imu.2022.100916
24. Agarwal, C., Vishwakarma, V. P. (2022). Comparison of Different Deep CNN Models for Leukemia Diagnosis. Proceedings of the International Conference on Cognitive and Intelligent Computing, 659–672. https://doi.org/10.1007/978-981-19-2350-0_63
25. Gulzar, Y. (2023). Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique. Sustainability, 15 (3), 1906. https://doi.org/10.3390/su15031906
26. Indraswari, R., Rokhana, R., Herulambang, W. (2022). Melanoma image classification based on MobileNetV2 network. Procedia Computer Science, 197, 198–207. https://doi.org/10.1016/j.procs.2021.12.132
27. Theckedath, D., Sedamkar, R. R. (2020). Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks. SN Computer Science, 1 (2). https://doi.org/10.1007/s42979-020-0114-9
28. Cai, W., Li, M., Jin, G., Liu, Q., Lu, C. (2024). Comparison of Residual Network and Other Classical Models for Classification of Interlayer Distresses in Pavement. Applied Sciences, 14 (15), 6568. https://doi.org/10.3390/app14156568
29. Patra, P., Singh, T. (2022). Diabetic Retinopathy Detection using an Improved ResNet 50-InceptionV3 and hybrid DiabRetNet Structures. 2022 OITS International Conference on Information Technology (OCIT), 140–145. https://doi.org/10.1109/ocit56763.2022.00036
30. Wang, C., Chen, D., Hao, L., Liu, X., Zeng, Y., Chen, J., Zhang, G. (2019). Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model. IEEE Access, 7, 146533–146541. https://doi.org/10.1109/access.2019.2946000
31. Wang, X., Li, J., Tao, J., Wu, L., Mou, C., Bai, W. et al. (2022). A Recognition Method of Ancient Architectures Based on the Improved Inception V3 Model. Symmetry, 14 (12), 2679. https://doi.org/10.3390/sym14122679
32. Harumy, T. H. F., Zarlis, M., Lydia, M. S., Efendi, S. (2024). Image classification of diseases in rice using deep neural neural network and incevtion V3. Proceedings Of The 6th International Conference On Computing And Applied Informatics 2022, 2987, 020046. https://doi.org/10.1063/5.0200203