Comparative Analysis of CNN Architectures for SIBI Image Classification

Yulrio Brianorman[1*], Dewi Utami[2]

[1]Informatics Engineering, Univeristy of Muhammadiyah Pontianak, Indonesia
[2]Communication Science, University of Tanjungpura, Indonesia

*corr_author: y.brianorman@unmuhpnk.ac.id

Abstract - The classification of images from the Indonesian Sign Language System (SIBI) using VGG16, ResNet50, Inception, Xception, and MobileNetV2 Convolutional Neural Network (CNN) architectures is evaluated in this paper. With Google Colab Pro, a 224 × 224-pixel picture dataset was used for the study. A five-stage technique consisting of Dataset Collection, Dataset Preprocessing, Model Design, Model Training, and Model Testing was applied. Performance evaluation focused on accuracy, precision, recall, and F1-Score. The results identified VGG16 as the top-performing model with an accuracy of 99.60% and an equivalent F1-Score, followed closely by ResNet50 with nearly similar performance. Inception, XCeption, and MobileNetV2 demonstrated balanced performance but with lower accuracy. This study sheds light on the best CNN models to choose for SIBI image classification, and it makes recommendations for further research that include using sophisticated data augmentation methods, investigating novel CNN architectures, and putting the models to practical use.

Keywords: sign language, pre-trained model, CNN, MnetV2, VGG16, ResNet50, xception, inception

I. INTRODUCTION

Human communication has changed over the ages, taking on diverse forms and techniques. A key element of this progression has been the employment of body language and complementing words, which can occasionally function as the main means of communication. This is especially important for deaf people, whose primary language is signing language, which consists of hand motions, body gestures, and facial expressions. Sign language is a rich and complex system with many emotional and contextual nuances inherent in each movement and expression. Comprehending and analyzing these motions offers a valuable understanding of how the deaf community engages and perceives its surroundings.

American Sign Language (ASL) is not limited to the US; it is widely used worldwide. Notably, the Indonesian Sign Language System (SIBI) has adopted several elements from ASL, particularly in terms of representing letters and numbers. The deaf community in Indonesia utilizes techniques derived from ASL when using SIBI to express the Latin alphabet letters from 'A' to 'Z' and numbers from '0' to '9'. This enables speakers of ASL and SIBI to spell out words manually, letter by letter, which proves highly useful. Sign language is flexible and rich in adapting elements of written language, as evidenced by the implementation of handshapes in both ASL and SIBI. This adoption highlights the global significance of ASL and how it aids the unique communication needs of the deaf community in many countries, including Indonesia.

The focus of this research is on the sign language found within the Indonesian Sign Language System (SIBI). A thorough understanding of letters and numbers in this sign system is crucial because many deaf communities in Indonesia are still underexposed to the representation of the Latin alphabet in SIBI, which is essential for their daily lives. The ability to express each letter and number clearly and accurately in SIBI is vital for the Indonesian deaf community, as they rely on SIBI hand movements to communicate the same information. This ability is akin to the capability of hearing individuals to comprehend sounds.

ConvNets, also known as Convolutional Neural Networks (CNN), are an effective machine-learning method for image processing. CNNs are better than regular neural networks because they are ideal for applications like image recognition since they can automatically and adaptively learn the spatial features of images. The convolutional layers, which perform feature extraction, are at the heart of CNN architecture. These layers can detect features such as edges, textures, and shapes in images. After the features are extracted, the results are usually processed through one or more fully connected layers. This process, often referred to as a multilayer perceptron, functions to classify or make decisions based on the extracted features [1]. With these capabilities, CNNs are now the standard for many deep-learning applications, especially those related to computer vision and image processing.

The utilization of artificial neural networks (ANN) for identifying finger alphabets in Sign Language was the subject of Rozani's research [2]. This application achieved an accuracy of 77.08% in identifying finger alphabets using the backpropagation method. Its strengths include the ability to recognize more than one alphabet in a single test and adapt to images taken from various distances. However, a limitation of this study is that it did not detail the CNN architecture used in the training for sign language recognition.

The research focus of Bagus et al. was on the recognition of numerical signs in SIBI (Sistem Isyarat Bahasa Indonesia) using a Convolutional Neural Network (CNN) [3]. This study involved a three-stage processing method, with epochs set at 25, 50, and 100. The results showed a significant increase in accuracy, reaching a peak of 96.44% in training and 98.89% in data prediction. These outcomes indicate that the CNN method is highly effective in recognizing SIBI numerical signs with a very low error rate. However, a limitation of this study is that it only focused on the recognition of numbers from 0 to 9.

Yolanda, Gunadi, and Setyati conducted additional research on the recognition of hand sign language, specifically the alphabet, commonly used by people with disabilities [4]. While there has been progress in processing static sign language images, challenges persist in processing dynamic images or videos. In their study, they employed Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) with video inputs. The CNN extracted spatial features, while the RNN correlated the frames extracted by the CNN over time. The results were displayed as a text representing the alphabet derived from the recognized sign language. Due to limitations in the supporting system for the developed architecture, real-time testing was unsuccessful, with an average accuracy of 60.58% for full letters.

Perdana, Putra, and Dharmadi. researched recognizing sign language, particularly numbers in American Sign Language (ASL), using a machine learning system [5]. This study developed a system that utilizes preprocessing methods to optimize results, addressing the common difficulty many people face in learning sign language. The research employed a Convolutional Neural Network (CNN) architecture with MobileNetV2. The results showed that the combination of preprocessing methods like Grayscale, HSV, and Global Threshold achieved the best recognition accuracy of 97%, indicating that the system is effective in understanding and classifying ASL sign language. The study is limited, though, because the information utilized in it only contains the 26 letters of the SIBI alphabet; it excludes digits 0 through 9.

Sholawati, Auliasari, and Ariwibisono utilized the Convolutional Neural Network (CNN) method to classify sign language demonstrations recorded in real-time by teachers and students [6]. The dataset included 416 digital images, encompassing images of the active movements of the J and Z sign language alphabets. The system developed in this study displayed class labels and probability values of classification results on a website through a webcam interface. The testing, involving two participants, yielded accuracy, recall, specificity, and sensitivity values of 80.76% based on the confusion matrix formula. One limitation of this research was that the architecture used was a proprietary creation of the researchers, meaning its generalizability has not yet been tested.

In their research, Fadhilah and Marpaung employed the Convolutional Neural Network (CNN) method, known for its effectiveness in deep learning for image recognition, to develop a learning medium for recognizing the SIBI alphabet [7]. The dataset used consisted of 2,600 images, divided into 20% for validation and 80% for training. After conducting ten experiments and comparing various parameters such as batch size and number of epochs, it was found that the best accuracy reached 85%. This result confirms the effectiveness of CNN in recognizing the SIBI alphabet. The study is limited, though, because the information utilized in it only contains the 26 letters of the SIBI alphabet; it excludes numbers 0 through 9.

Naufal and Kusuma compared the algorithms K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Convolutional Neural Network (CNN) with transfer learning [8]. The study utilized transfer learning algorithms such as Xception, ResNet50, VGG15, and MobileNetV2. The results showed that the CNN with the Xception architecture achieved the highest F1 score of 99.57% with an average training time of 1,387 seconds. Meanwhile, KNN with K = 1 had the fastest training speed, with a training time of only 0.03 seconds, and achieved an F1 score of 86.95%. This research offers crucial insights into algorithm selection for SIBI image recognition. A limitation of this study is that the pre-trained architectures did not use hyperparameter tuning to enhance performance. Additionally, the dataset did not include sign language numbers. Other related research on the use of CNN for classification beyond SIBI images includes the classification of Batik Garutan images [9], the recognition of images for improving the early warning system for tsunamis in Indonesia [10], and the identification of fashion images [11].

In this study, the main problem formulated involves assessing the effectiveness of various pre-trained Convolutional Neural Networks (CNN) architectures such as Inception, MobileNetV2, VGG16, and Xception in classifying images of the Indonesian Sign Language System (SIBI), which includes the alphabet from A to Z and numbers from 0 to 9. This includes investigating the relationship between training duration and classification effectiveness, as well as comparing the training time required by each CNN architecture on the SIBI dataset. Furthermore, this research aims to evaluate and compare performance metrics such as accuracy, precision, recall, and F1-Score for each model to determine their relative performance in the SIBI image classification task, thereby indicating which model is most suitable for this application.

The research methodology employed includes dataset collection, dataset pre-processing, training of classification models with hyperparameter adjustments, testing of classification models, and calculation of classification performance. This method is expected to enhance understanding of the effectiveness of different network architectures in classifying SIBI (Sistem Isyarat Bahasa Indonesia) images. The benefit of this research is anticipated to significantly improve comprehension of the effectiveness of various network architectures in classifying SIBI images, providing crucial insights that can be used to optimize applications and the development of image pattern recognition technologies in the future.

II. METHOD

In this study, four key research steps were executed. The first step involved the collection of the dataset. This was followed by the training of the classification model. Next, the trained classification model was tested. The final step entailed evaluating the classification performance. This performance evaluation also included a comparison of the performance across various algorithms. Fig. 1 illustrates the sequence of these research steps, which were followed systematically.

A. Dataset Collection

The SIBI image dataset used in this research was obtained from a public dataset source on Kaggle (https://bit.ly/3trwIlH). The decision to use this dataset was based on its open availability and accessibility to other researchers. Utilizing a public dataset facilitates comparisons with other methods that may be proposed by researchers in the future. This dataset comprises 36 classes representing the alphabet letters A to Z and numbers 0 to 9. Each class contains 70 images, except for the letter "T," which has 65 images, resulting in a total of 2515 images used in the study. Examples of images for each class are presented in Fig. 2.

Fig. 1. The flow of the research methodology in this study

Fig. 2 Image of SIBI alphabets and numbers

B. Dataset Preprocessing

The resolution of the captured images in the study was initially set at 400x400 pixels. Subsequently, these images were resized to 224x224 pixels to align with the input size requirements of the pre-trained models to be used in the research. This resizing is a crucial step to ensure compatibility with the chosen models. The image data was stored in the form of numpy arrays, which facilitates the learning process by eliminating the need for additional conversion from images to numpy arrays during training. The extension of these files is *.npy, indicating their numpy array format. The files can be accessed through the following link: https://bit.ly/citraSIBI.

C. Model Design

At this stage, each algorithm is configured with specific parameters, aiming to evaluate the impact of these parameters on classification performance. In the CNN algorithm, the parameters tested include the number of epochs, types of convolutional layers, types of activation functions, and the number of dense layers. Additionally, CNN utilizes transfer learning architectures such as InceptionV3, MobileNetV2, ResNet50, VGG16, and Xception. This approach helps in understanding how different settings and architectures influence the overall effectiveness of the model in classifying images.

InceptionV3, an evolution of the earlier Inception model, is designed to enhance image recognition accuracy and reduce overfitting. This model incorporates improvements such as the use of auxiliary classifiers, convolution factorization, and label smoothing regularization. With these features, InceptionV3 becomes more efficient and flexible in handling complex image recognition tasks [12].

MobileNetV2 enhances the performance of mobile architectures across various tasks and benchmarks. This architecture employs inverted residual blocks and linear bottlenecks, enabling higher memory efficiency, crucial for mobile applications. Techniques such as 1x1 convolutions, inverted residual blocks, batch normalization, and residual connections improve gradient propagation capabilities and reduce memory requirements [13].

ResNet50 is a 50-layer Convolutional Neural Network (CNN) that utilizes residual blocks, enabling the addition of more convolutional layers without encountering the problem of vanishing gradients, thanks to the use of shortcut connections. These connections "skip" some layers, transforming a regular network into a residual network [14]. Despite ResNet50's large number of parameters, it has a simpler architecture compared to VGGNet. It remains effective with fewer filters and reduced complexity, making it faster in training.

The VGG16 model is a Convolutional Neural Network (CNN) that supports 16 layers. It is designed to replace large filters with a sequence of smaller 3x3 filters, enhancing non-linear activation functions and allowing the network to converge quickly. VGG16 utilizes these smaller convolutional filters to reduce the tendency towards overfitting during training, making it the smallest model capable of understanding the spatial features of images [15].

Xception is a convolutional neural network architecture that fully relies on depthwise separable convolution layers. It represents an enhancement of the Inception architecture, modifying depthwise separable convolutions for improved performance. Xception posits that the mapping of cross-channel correlations and spatial correlations within the feature maps of convolutional neural networks can be entirely decoupled. Comprising 36 convolutional layers structured into 14 modules, all equipped with linear residual connections around them, the Xception architecture offers a simpler and more efficient approach compared to Inception V3. This efficiency is particularly evident in terms of the number of parameters and training performance [16].

The transfer learning technique employed involves omitting the output part of each architecture. Subsequently, this output section is replaced with an output layer consisting of 36 neurons used for the classification process, employing softmax activation. An illustration of the CNN architecture design using pre-trained ImageNet models is depicted in Fig. 3.

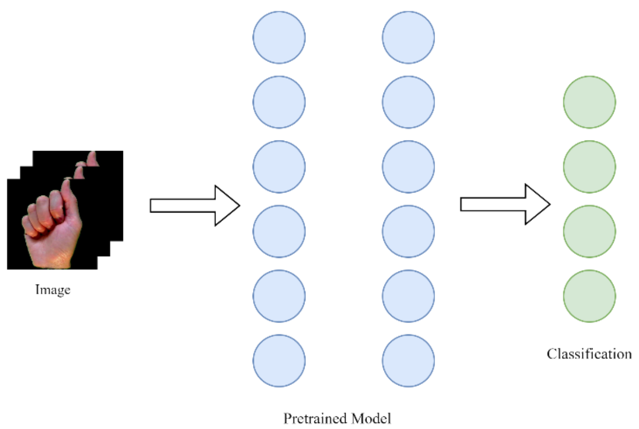


Fig. 3 The design of an architecture for classifying letters and numbers using pre-trained models

Fig. 4 illustrates a comparison of the number of parameters among five different CNN architectures: VGG16, InceptionV3, MobileNetV2, ResNet50, and Xception. From the graph, it is evident that VGG16 has the lowest number of parameters, 903,204, indicating a relative simplicity in its architecture. This suggests potential for faster training and lighter implementation, though it may come with limitations in accuracy or the ability to learn complex features. Conversely, Xception and ResNet50, each with 3,612,708 parameters, indicate higher complexity, often associated with better capability to learn detailed data features, potentially leading to higher accuracy. InceptionV3 and MobileNetV2, with parameter counts in between, offer a balance between complexity and efficiency. However, the number of parameters is not the sole determinant of model performance, as factors like data quality, specific architecture, training methods, as well as memory and computational time requirements, also play crucial roles.

D. Training Model

At this stage, all the image data used have a size of 224x224, consistent with the dimensions set during the collection of image data. The total image data consists of 2,515 images, which include 26 signals representing letters and 10 signals for numbers. All images trained in this study are 224x224 pixels in size, in line with the dimensions established when the image data was collected. The total number of images available is 2,515, comprising 26 signals for letters and 10 signals for numbers.

Next, the image data is divided into three categories: training data, validation data, and test data. This division process is done in two steps: first, all data is randomized, and then divided with a proportion of 80% for training data, 10% for validation data, and the remaining 10% for test data. The details of this data division can be seen in TABLE I.

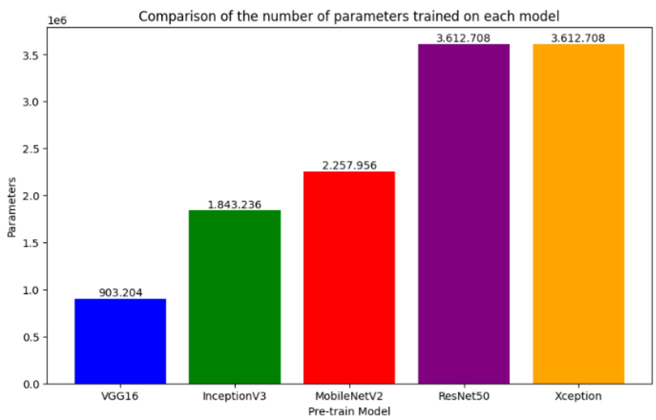


Fig. 4 Graph of the total number of parameters used in each pre-trained model

TABLE I
DIVISION OF TRAINING DATA, VALIDATION DATA, AND TEST DATA

Data	Percentage	Amount
Training Data	80%	2012
Validation Data	10%	251
Test Data	10%	252

The hyperparameter configuration for the CNN architecture can be seen in

		True/Actual Class	
		Positive(P)	*Negative(N)*
Predicted Class	*True (T)*	*True Positive (TP)*	*False Positive (FP)*
	False (F)	*False Negative (FN)*	*True Negative (TN)*
		P=TP+FN	*N=FP+TN*

Fig. 5 Confusion matrix

TABLE II. During the training process, the hyperparameters are adjusted as follows: the use of the Adam optimizer, a learning rate set at 0.001, the loss function utilizing categorical loss, training carried out for 30 epochs, and a batch size set to 32. Additionally, the learning process will be halted if there is no improvement in accuracy over 5 consecutive epochs. This is indicated by a hyperparameter of patience = 5.

E. Testing Model

The testing process conducted using the image data of SIBI letter and number signs aims to evaluate the classification performance at each stage, including training, validation, and testing. The performance of each stage will be measured using various metrics, including precision (1), recall (2), accuracy (3), and F1-Score (4). These metrics provide a comprehensive overview of the model's effectiveness in accurately classifying data. This testing will be consistently applied to each pre-trained CNN architecture used in this research, ensuring that each architecture can be evaluated and compared based on the same criteria. Thus, it will be possible to determine which architecture is most effective in recognizing and processing SIBI letter and number sign images. The confusion matrix was illustrated in Fig. 5.

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

$$F1\ score = \frac{Precision\ x\ Recall}{Precision+Recal} \tag{4}$$

		True/Actual Class	
		Positive(P)	*Negative(N)*
Predicted Class	*True (T)*	*True Positive (TP)*	*False Positive (FP)*
	False (F)	*False Negative (FN)*	*True Negative (TN)*
		P=TP+FN	*N=FP+TN*

Fig. 5 Confusion matrix

TABLE II
CNN ARCHITECTURE HYPERPARAMETERS

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0,001
Loss function	Categorical Cross Entropy
Epoch	30
Batch size	32
Patience	5

Fig. 5 explains the four main variables: TP, FP, FN, and TN, and visualizes four possible prediction scenarios. In the image, white boxes represent accurate predictions, while gray boxes indicate inaccurate predictions. The True Positive (TP) variable indicates a match between a correct prediction and a true reality. Conversely, True Negative (TN) indicates a match between a wrong prediction and a false reality. Meanwhile, a False Positive (FP) represents a situation where the prediction indicates something as true when it is false. Finally, False Negative (FN) refers to cases where the prediction indicates something as false when it is true.

III. RESULT AND DISCUSSION

This section will focus on two crucial aspects: the discussion of results from training and validation, and the evaluation of testing outcomes. The training and validation process is essential for measuring the model's effectiveness in understanding data, while testing provides insights into the model's performance with new data. This research utilizes Google Colab Pro as the primary platform for training and testing, at a rental cost of $11.09 per month, allowing access to high computing resources. This ensures that the research is conducted with adequate resources, optimizing the performance and efficiency of the tested models.

A. Training and Validation Results

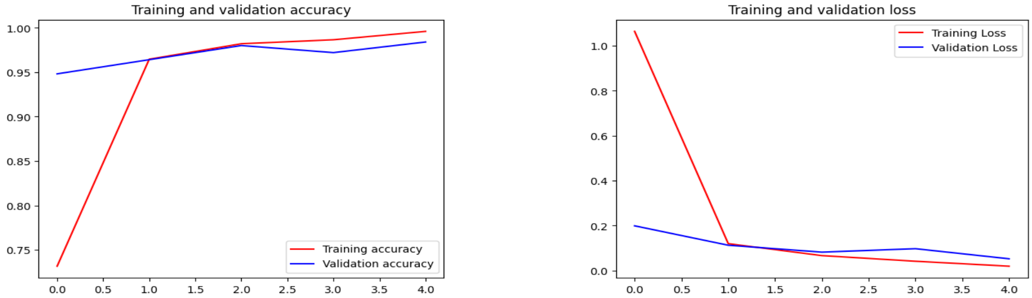
The training process was conducted using the concept of transfer learning from each pre-trained CNN model, namely MnetV2, VGG16, ResNet50, and Xception. This training was carried out for 100 epochs with hyperparameters as shown in Table II. However, the training would be stopped if there was no improvement in accuracy over 5 consecutive epochs. Fig. 6 shows the training results for each model used in this study.

The analysis of the performance of different pre-trained Convolutional Neural Networks (CNN) models reveals intriguing results. The VGG16 model, with a training accuracy of 99.60% and validation accuracy of 98.41% in just five epochs, demonstrates an exceptional ability to generalize learning from training data to validation data. The Inception model indicates that improvements in training strategies or architectural adjustments are needed to enhance efficiency and effectiveness. With a training accuracy of 100.00% in 12 epochs, MobileNetV2 achieved a validation accuracy of 95.62%, suggesting the potential for overfitting, which warrants further investigation. ResNet50, with 99.40% training accuracy and 97.21% validation accuracy over 26 epochs, showcases consistent and reliable performance, indicating its capability to balance learning and generalization. Lastly, XCeption also achieved perfect training accuracy at 100.00%, but its validation accuracy of 96.81% in 16 epochs suggests a possibility of overfitting.

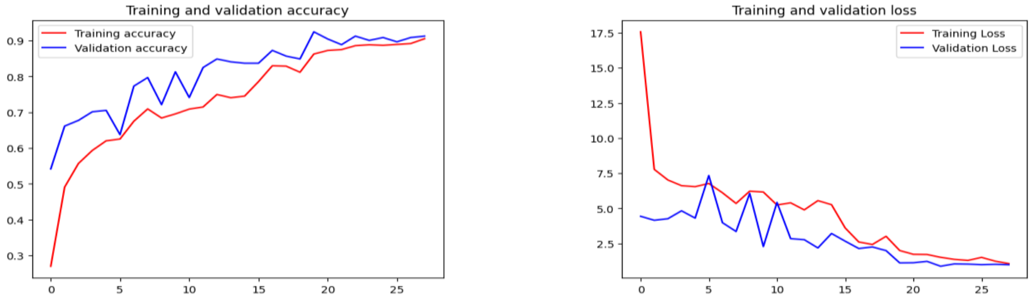
Overall, these findings emphasize the importance of measuring both training accuracy and validation performance to assess a model's generalization ability. While MobileNetV2 and XCeption require more careful handling to address overfitting issues, models like VGG16 and ResNet50 strike a good balance between learning and generalization. By evaluating accuracy, training efficiency, and generalization capability, this research provides critical insights on how to best select CNN architectures for specific image classification applications. The complete training and validation results are presented in TABLE III.

TABLE III
TRAINING ACCURACY AND VALIDATION RESULTS

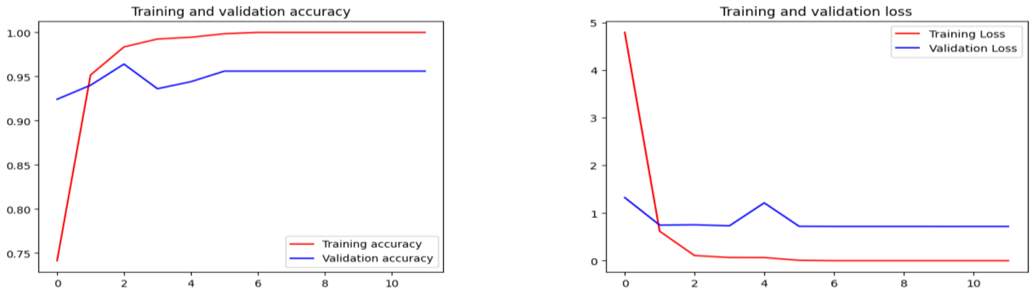
Pre-trained Model	Accuracy	Validation	Epoch
VGG16	0.9960	0.9841	5
Inception	0.9051	0.9124	28
MobileNetV2	1.0000	0.9562	12
ResNet50	0.9940	0.9721	26
XCeption	1.0000	0.9681	16



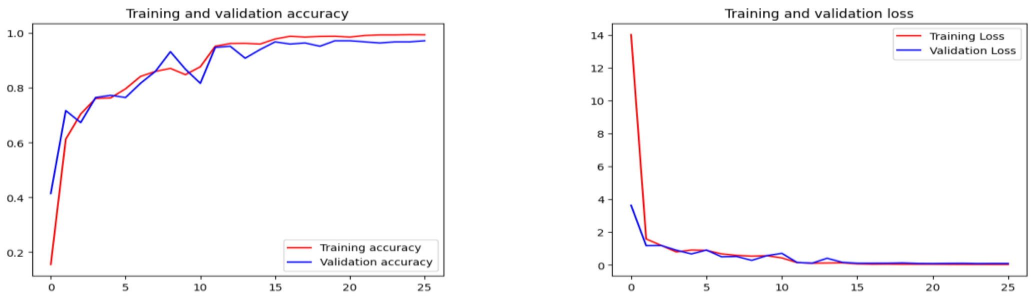
a. Model VGG16



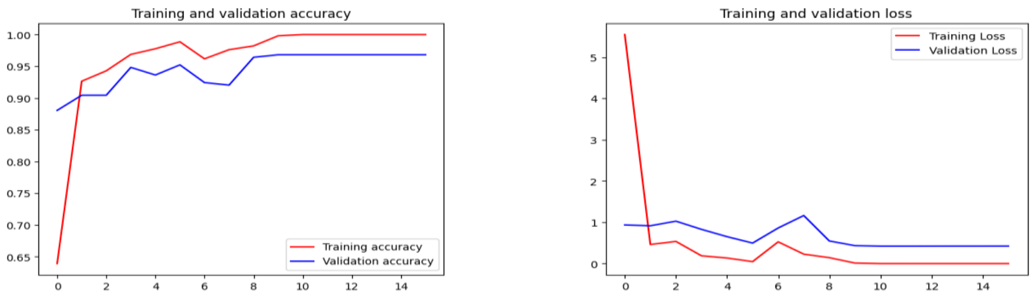
b. Model Inception



c. Model ModelNetV2



d. Model ResNet50



e. Model Xception

Fig. 6 Graph of training results for each pre-trained

The significance of selecting an appropriate CNN architecture for SIBI image classification is shown by the training and validation outcomes. The high accuracy of models like VGG16 and ResNet50 can be attributed to their architectural strengths—VGG16's deep layers effectively capture detailed visual features, while ResNet50's residual connections facilitate deeper learning without gradient vanishing, crucial for the nuanced differentiation required in sign language images. This analysis not only aligns with our research objectives but also highlights the effectiveness of specific network architectures in addressing the challenges of SIBI image classification, paving the way for future advancements in the field.

B. Test Results

If validation data is used as a performance check during the training process and is consistently observed at each epoch, then test data represents data that the pre-trained model has never seen during the training process. In other words, this data can be used as a benchmark for the actual implementation of the system. A model's training outcome might perform well in experiments, but it may not show good accuracy on data it has never seen during the training process. This underscores the importance of testing a model using test data. The division of the dataset is illustrated in TABLE IV.

TABLE IV
TEST ACCURACY RESULTS

Pre-Trained Model	Accuracy	Precision	Recall	F1-Score
VGG16	0.9960	0.9965	0.9960	0.9960
Inception	0.8769	0.9125	0.8769	0.8827
MobileNetV2	0.9880	0.9894	0.9880	0.9880
ResNet50	0.9880	0.9896	0.9880	0.9880
XCeption	0.9523	0.9592	0.9523	0.9526

The testing results using test data reveal significant performance differences among various pre-trained Convolutional Neural Networks (CNN) models. One of the best-performing models, VGG16 (Fig. 7), shows exceptional performance; it achieved an accuracy of 87.69% and an almost perfect F1-Score of 88.27%, indicating VGG16's highly effective capability in classifying test data with high accuracy. In contrast, the Inception model exhibits lower performance, with an accuracy of 87.69% and an almost perfect F1-Score, all above 99%. These results suggest that, compared to other models, Inception may be less effective in generalizing learning to test data. Meanwhile, MobileNetV2 and ResNet50 demonstrate impressive and nearly identical results, with accuracy, precision, recall, and F1-Scores around 98.80%, indicating high reliability in classifying test data. Lastly, XCeption, with an accuracy of 95.23% and an F1-Score of 95.26%, shows solid performance but is slightly below the standard set by MobileNetV2 and ResNet50. In conclusion, while each model has its strengths, VGG16 and ResNet50 emerge as highly effective choices for applications requiring high accuracy and consistency in image classification.

The VGG16 architecture is a prominent deep convolutional neural network that is both deep and straightforward in design, consisting of thirteen convolutional layers that use 3x3 filters, the smallest practical size for capturing image features such as patterns, edges, colors, and textures. It processes input images of 224x224 pixels resolution through these convolutional layers, each followed by a ReLU activation function to add non-linearity. The network employs 2x2 max pooling layers to reduce spatial dimensions between convolutions. It concludes with three densely connected layers, boasting 4096, 4096, and 1000 neurons respectively, leading to a softmax output layer that classifies images into 1000 categories, which is typical for ImageNet competition tasks. This design has positioned VGG16 as a powerful tool for feature extraction and image classification.

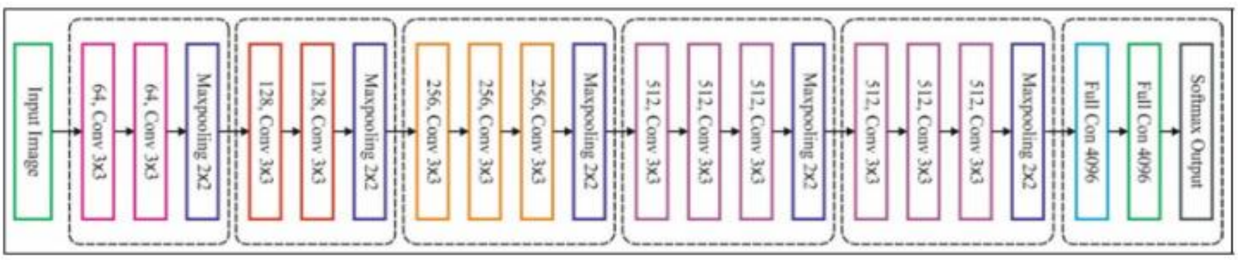


Fig. 7 VGG16 architecture

Moreover, the VGG16 model's success in various applications can be attributed to its ability to generalize well from large pre-trained datasets like ImageNet to specific tasks with limited data availability. This transfer learning potential is particularly beneficial in fields like sign language recognition and medical image analysis where collecting annotated datasets is challenging. VGG16's adaptability and resilience as a feature extractor are highlighted by its efficacy in diverse fields, including dementia severity assessment, cancer prediction, and bone anomaly identification. Such achievements are documented in studies like "Bone Abnormalities Detection and Classification Using Deep Learning-Vgg16 Algorithm" [17] and "VGG16 Feature Extractor with Extreme Gradient Boost Classifier for Pancreas Cancer Prediction," [18] underscoring the model's broad applicability and exceptional performance in image classification tasks.

IV. CONCLUSION

The testing results across various Convolutional Neural Networks (CNN) architectures indicate that VGG16 and ResNet50 stand out in evaluating different CNN models. VGG16 excels with nearly perfect accuracy (99.60%) and similar scores in precision, recall, and F1-Score, demonstrating its exceptional ability to classify data with high precision. In contrast, ResNet50 also shows strong performance with accuracy, precision, recall, and F1-Score around 98.80%, indicating its effectiveness in classifying data accurately and effectively. Both models exhibit high suitability for applications that require high levels of accuracy and consistency. Meanwhile, XCeption and Inception offer balanced performances, with XCeption recording accuracy and F1 Scores above 95%, and Inception showing potential for improvement. MobileNetV2, despite its high accuracy and recall, has a lower F1-Score, indicating potential challenges in balancing the identification of true positives and avoiding false positives. In conclusion, while VGG16 and ResNet50 stand out as the best models for applications demanding high levels of accuracy and precision, other models like XCeption and Inception remain relevant for specific needs, whereas MobileNetV2 requires further evaluation in aspects of precision and recall. There are opportunities to test new CNN architectures such as EfficientNet or Transformer-based models, as well as to implement more advanced methods for data augmentation and transfer learning in future research. To combat overfitting, it's also crucial to focus on regularization techniques and hyperparameter optimization. Furthermore, conducting investigations into the interpretation and transparency of models can provide valuable insights, especially for determining their relevance in various practical situations.

REFERENCES

[1] Md Zahangir Alom , Tarek M. Taha , Chris Yakopcic , Stefan Westberg , Paheding Sidike , Mst Shamima Nasrin ,Brian C Van Essen , Abdul A S. Awwal , and Vijayan K. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.01164

[2] A. Rozani, "Penerapan Metode Jaringan Syaraf Tiruan Pada Aplikasi Pengenalan Bahasa Isyarat Abjad Jari," *Jurnal Mahasiswa Teknik Informatika*, vol. 1, no. 1, pp. 311–317, 2017, doi: https://doi.org/10.36040/jati.v1i1.1897.

[3] M. Bagus, S. Bakti, and Y. M. Pranoto, "Pengenalan Angka Sistem Isyarat Bahasa Indonesia Dengan Menggunakan Metode Convolutional Neural Network," in *Prosiding SEMNAS INOTEK*, 2021, pp. 11–16. doi: https://doi.org/10.29407/inotek.v3i1.504.

[4] D. Yolanda, K. Gunadi, and E. Setyati, "Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real-Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network," *JURNAL INFRA*, vol. 8, no. 1, 2020.

[5] I. P. I. Perdana, I. K. G. D. Putra, A. Dharmaadi, and I. Putu, "Classification of Sign Language Numbers Using the CNN Method," *Jurnal Ilmiah Teknologi dan Komputer*, vol. 2, no. 3, pp. 485–493, 2021.

[6] M. Sholawati, K. Auliasari, and FX. Ariwibisono, "Pengembangan Aplikasi Pengenalan Bahasa Isyarat Abjad Sibi Menggunakan Metode Convolutional Neural Network (CNN)," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 1, pp. 134–144, Mar. 2022, doi: 10.36040/jati.v6i1.4507.

[7] Z. Fadhilah and N. L. Marpaung, "Pengenalan Alfabet SIBI Menggunakan Convolutional Neural Network sebagai Media Pembelajaran Bagi Masyarakat Umum," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 8, no. 2, pp. 162–168, May 2023, doi: 10.30591/jpit.v8i2.5221.

[8] M. F. Naufal and S. F. Kusuma, "Analisis Perbandingan Algoritma Machine Learning dan Deep Learning untuk Klasifikasi Citra Sistem Isyarat Bahasa Indonesia (SIBI)," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 10, no. 4, pp. 873–882, Aug. 2023, doi: 10.25126/jtiik.20241046823.

[9] L. Tresnawati and D. B. Sukriyansah, "Image Classification on Garutan Batik Using Convolutional Neural Network with Data Augmentation," *Jurnal*

Informatika, vol. 11, no. 1, pp. 107–115, 2023, [Online]. Available: www.kaggle.com/datasets/ionisiusdh/indones

[10] H. A. Nugroho, S. Hasanah, and M. Yusuf, "Seismic Data Quality Analysis Based on Image Recognition Using Convolutional Neural Network," *Jurnal Informatika*, vol. 10, no. 1, pp. 67–75, 2022.

[11] C. Sri, K. Aditya, V. Rahmayanti, S. Nastiti, Q. R. Damayanti, and G. B. Sadewa, "Implementation of Convolutional Neural Network Method in Identifying Fashion Image," *Jurnal Informatika*, vol. 11, no. 2, pp. 195–202, 2023.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[15] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, IEEE, Nov. 2015, pp. 730–734. doi: 10.1109/ACPR.2015.7486599.

[16] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 1800–1807. doi: 10.1109/CVPR.2017.195.

[17] V. S. Navale, "Bone Abnormalities Detection and Classification Using Deep Learning-Vgg16 Algorithm," *Int J Res Appl Sci Eng Technol*, vol. 11, no. 7, pp. 122–129, Jul. 2023, doi: 10.22214/ijraset.2023.54582.

[18] W. Bakasa and S. Viriri, "VGG16 Feature Extractor with Extreme Gradient Boost Classifier for Pancreas Cancer Prediction," *J Imaging*, vol. 9, no. 7, p. 138, Jul. 2023, doi: 10.3390/jimaging9070138.