Implementation of MobileNetV2 and OpenCV in a Real-Time Sign Language Recognition System

Marcellinus Ronan Narendra[1], Radius Tanone[2]
Universitas Kristen Satya Wacana
Jalan Diponegoro No. 52-60, Kel. Salatiga, Kec. Sidorejo, Kota Salatiga, Prov. Jawa Tengah, Indonesia, 50711
marcellinus205@gmail.com

Abstract— The main means of communication for the deaf is sign language, although communication hurdles are frequently caused by a lack of public comprehension. The goal of this project is to use the OpenCV library and MobileNetV2 architecture to create a real-time American Sign Language (ASL) recognition system. The ASL Alphabet from Kaggle, which has over 87,000 photos in 29 classes, is the dataset that was used. Convolutional Neural Networks (CNNs) based on MobileNetV2 with picture preprocessing and data augmentation were used to train the model. OpenCV was then used to combine the CNN with a camera for real-time implementation.
The evaluation results indicate an average F1-score of 83.6, recall of 83.8, and precision of 83.5. The system is responsive for direct interaction because it can operate at 18–22 frames per second (FPS) on a typical laptop. The system still has issues with complicated backdrops, low light levels, and gesture similarities between some letters. Overall, this study demonstrates the efficacy of MobileNetV2 and OpenCV in developing a lightweight, efficient real-time sign language recognition system that facilitates inclusive communication for individuals with hearing impairments.

Keywords— sign language, deep learning, MobileNetV2, OpenCV, real-time recognition

I. INTRODUCTION

When dealing with one another and the wider public, deaf and mute people mostly communicate through sign language. However, there are communication barriers and information inequality in Indonesia due to the public's very low comprehension of sign language. People with impairments find it difficult to participate in social life, education, and the workforce as a result of this circumstance[1]. Therefore, in order to close the communication gap between the general population and deaf persons, technology-based solutions are required. New technologies that allow deaf persons to effectively communicate with the broader public have been developed as a result of this issue. An automatic sign language recognition system that makes use of computer technology might be the answer. Recent years have seen significant advancements in the recognition of visual patterns, including hand gestures, thanks to image processing and deep learning. Convolutional Neural Networks (CNNs) are one potential method[2]. CNNs can accurately classify a variety of hand motions, according to earlier research. MobileNetV2 is a lightweight and effective CNN architecture that maintains competitive accuracy while being optimized for mobile devices and real-time deployment[3]. In this situation, creating a responsive and effective sign language recognition system is best accomplished by utilizing MobileNetV2. Without appreciably sacrificing accuracy, this model is made to be deployed on devices with computing constraints.

By utilizing depthwise separable convolution and inverted residuals, MobileNetV2 is able to drastically reduce the number of parameters and computational load[4]. The integration of MobileNetV2 with

OpenCV enables direct hand gesture recognition through a camera, thereby supporting real-time, interactive systems[5]. Additionally, support from OpenCV, an open-source library for image processing, enables direct camera integration, allowing for real-time sign language recognition. Thus, this system can not only recognize letters or gestures from static images, but also enables direct interaction through the device's camera [6]. Similar research is still lacking in Indonesia, particularly in relation to real-time implementation and the use of lightweight architectures, such as MobileNetV2. Therefore, this research is crucial in filling this gap by developing a real-time sign language recognition system using MobileNetV2 and OpenCV, and comprehensively evaluating its performance [7]. The research question in this study is how to implement the MobileNetV2 architecture for real-time American Sign Language (ASL) recognition using OpenCV [8], as well as how high the accuracy and efficiency of the system is in recognizing ASL hand gestures in real conditions. Based on this research question, this study aims to implement a real-time sign language recognition system using MobileNetV2 and OpenCV, as well as to evaluate the system's performance in supporting inclusive communication for people with disabilities in Indonesia [9]. It is anticipated that the study's findings will have practical, scholarly, and social implications. It is hoped that this system will make it easier for deaf people to interact with the community by using an automatic sign language recognition system. Academically, this work can be used as a guide for future research and development concerning the use of OpenCV and MobileNetV2 in real-time computer vision systems. This system is anticipated to empower individuals with disabilities and encourage inclusive communication through the use of artificial intelligence-based technology. With this system, it is hoped that communication between deaf people and the general public can be easier and more inclusive through the support of real-time sign language recognition technology. The next section will discuss the literature review that forms the theoretical basis for the development of this system [10].

II. LITERATURE REVIEW

American Sign Language and the Challenges of Its Recognition

American Sign Language (ASL) is a visual-gestural language used by the deaf community in the United States, parts of Canada, and other countries that have adopted this sign language system. Unlike verbal languages that use sound, ASL utilizes facial expressions, hand shapes and positions, and body movements to convey meaning [11]. ASL has a complex linguistic structure, including its own syntax and grammar, which is not limited to the A–Z alphabet, but also includes vocabulary for words, phrases, and sentences.

In the world of technology, the main challenges in automatic ASL recognition are the diversity of hand shapes between users, differences in skin color, ambient lighting, complex image backgrounds, and the similarity of gestures between some letters [12]. For instance, the ASL gestures for the letters "M" and "N" are extremely similar and can only be differentiated by the number of fingers above the thumb. To detect the finger positions and contours, a very accurate system is needed.

Another challenge is the need for a system that can run in real time. This system must be able to recognize hand gestures in milliseconds to enable natural interaction, similar to human conversation[13]. Therefore, one of the main goals of this research is to create a lightweight, effective, and accurate model.

Image Processing and the Role of OpenCV

Image processing is a technique for processing digital images through computer algorithms in order to extract important information contained within them [14]. In the context of ASL recognition, image

processing is used to detect and segment hands, reduce noise, and prepare image data in an optimal format before it is fed into a machine learning model. OpenCV (Open Source Computer Vision Library) is a widely used open-source library for image processing and computer vision[15]. OpenCV provides various functions such as object detection, motion tracking, color conversion from RGB to grayscale, thresholding, Gaussian blur, and many more. In this study, OpenCV was used to access the camera, capture images in real-time, and perform hand area segmentation (Region of Interest), which was then processed by the CNN model.

One of the advantages of OpenCV is its cross-platform capability and integration with popular programming languages such as Python and C++. In addition, OpenCV is able to run quickly even on computers with low specifications, making it very suitable for use in systems that emphasize real-time performance, such as ASL recognition.

Deep Learning and CNN

Deep learning is a machine learning method that automatically learns data representations from unprocessed inputs using deep neural networks [16]. Convolutional neural networks (CNNs) are the best deep learning architectures for image processing because they can automatically extract spatial features [17].

CNNs are specifically made to identify visual patterns, including objects, lines, shapes, and textures. In the context of American Sign Language (ASL) recognition, CNNs are used to learn the characteristics of each letter. There are several main components of CNN, including:

1. Convolution Layer

This layer functions as an automatic feature extractor from images. A small filter (kernel), for example measuring 3×3, is shifted across the image to calculate new values called feature maps.[18]. The simple formula is:

$$Output = \sum (Picture\ Patch\ x\ Kernel)$$

This indicates that a small patch of the image and the kernel weights are multiplied to produce each value in the feature map. This filter automatically picks up on specific patterns, such as palm contours and finger edges.

2. Activation Function

After convolution, the result is passed to an activation function, the most common of which is ReLU (Rectified Linear Unit) to introduce non-linearity. The ReLU function is very simple but effective:

$$f(x) = \max(0, x)$$

3. Pooling Layer

Pooling is used to reduce data dimensions, thereby speeding up the training process and reducing the risk of overfitting[19]. The most common type of pooling is max pooling, which takes the maximum value from a small patch (e.g., 2x2). Without complicated formulas, pooling intuitively selects the most important parts of the detected features.

4. Fully Connected Layer & Softmax

After several convolutions and poolings, the final result is passed to a fully connected layer, which is a regular neural network layer for making the final prediction[20].

$$Class\ i\ Probability = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

Where:

- zi = score generated for class i,
- K = total number of classes (in this case 26 classes for letters A-Z)

MobileNetV2 Model

MobileNetV2 is a lightweight CNN model developed by the Google team to address the

need for fast processing on devices with limited computing power, such as smartphones, Raspberry Pi, or standard laptops. MobileNetV2 introduces two major innovations, namely:

- Inverted residuals with shortcut connections, which allow feature information to flow efficiently between layers.
- Depthwise separable convolution, significantly lowers the number of parameters and computational load by splitting the convolution process into two stages.

MobileNetV2 is significantly faster and lighter than traditional models like VGG or ResNet, with very little loss in accuracy [21]. In real-time sign language recognition applications, this is an added value because it allows direct processing of camera frames without significant delays.

In addition to its simple design, MobileNetV2 fully supports the PyTorch ecosystem through the TorchVision library. Model development and implementation are made simpler as a result. Additionally, trained models can be exported in ONNX and TorchScript formats, enabling their deployment across multiple platforms. Considering these advantages, MobileNetV2 was chosen as the main architecture for this research. This architecture can be readily incorporated into actual application systems and allows for the quick and efficient classification of hand images from the ASL dataset.

Sign Language Dataset

In developing a deep learning-based sign language recognition system, the availability of a rich and structured dataset is crucial. The dataset used in this study is the ASL Alphabet Dataset published by a user named Grassknoted on Kaggle and developed by Massey University. (https://www.kaggle.com/datasets/grassknoted/asl-alphabet)

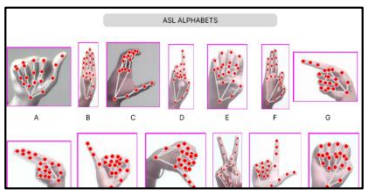

Figure 1. Example of ASL Dataset

More than 100,000 static hand images of American Sign Language (ASL) letters make up this dataset. There are three additional classes—"space," "delete," and "nothing"—that are frequently utilized for interaction in text-based applications or sophisticated gesture input, in addition to the 26 letters of the alphabet (A–Z).

This dataset is highly suitable for application with lightweight CNN models like MobileNetV2, as it is sufficiently representative and diverse in data distribution. The dataset structure is also appropriate, with each folder already separated by letter label, strongly supporting the supervised learning process. Furthermore, because the focus of this study is the recognition of static ASL letters, the use of this dataset is deemed appropriate for developing a real-time letter classification system that can be applied to desktop or mobile camera-based applications [22].

III. METHODS

This research focuses on developing a real-time sign language recognition system using a Convolutional Neural Network (CNN) model based on MobileNetV2 and the OpenCV library. This system is intended to recognize American Sign Language (ASL) letters from video input with high accuracy and fast processing speed. While the model was trained using the publicly available ASL Alphabet dataset from Kaggle, the system testing uses an integrated laptop camera.

This study is a form of applied research as it aims to produce a product in the form of a

sign language detection system. A quantitative experimental method is used, which includes data collection, model training, and numerical testing of system's performance. Figure 2 shows the workflow from dataset retrieval, data preprocessing, model training, and real-time implementation, to clarify the system's workflow in this study.

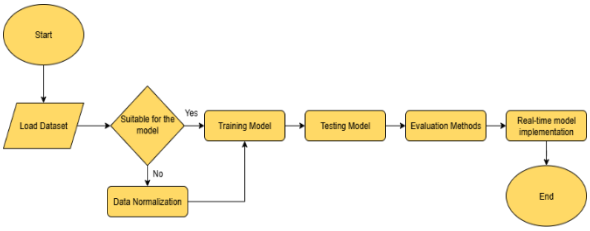

Figure 2. Workflow Diagram

Data Collection Process

The data used in this study is secondary data in the form of static hand images from a single open dataset source. The following table details the data source used:

Data Type	Source	Description
American Sign Language Alphabet (picture)	Kaggle (Grassknoted)	ASL Alphabet Dataset with 87.000 image and 29 classes

Table 1 Type of Data Used

The ASL Alphabet dataset from Kaggle consists of hand images across 29 classes (letters A–Z and three additional symbols). This dataset is used to train and test the performance of the CNN-based sign language recognition model. The dataset was downloaded directly from Kaggle via the "grassknoted" user repository.

Hardware and Tools

To build and test the real-time sign language recognition system, several supporting hardware and software tools are required. These tools were chosen because they support the computational needs for training deep learning models and processing images directly using a camera. The following are the details of the hardware and tools used:

Component	Description
Hardware	Laptop with Webcam
Programming Language	Python
Framework	PyTorch
Image Processing	OpenCV
IDE	Anaconda Prompt
Dataset	ASL Alphabet

Table 2 Hardware and supporting tool specifications

Research Stages and Pseudocode

The research stages were carried out sequentially so that the development of the real-time sign language recognition system could be done in a structured and systematic manner. Each stage plays an important role in ensuring the quality and effectiveness of the model being built. The following research stages were implemented:

Algorithm: ASL Recognition System **Input**: American Sign Language (ASL) hand gesture image dataset **Output**: Trained CNN model and real-time prediction of ASL letters

1. ***Import Dataset***

   The initial research stage begins with importing the dataset from Kaggle into the Python programming environment. The dataset is accessed and read using libraries such as Numpy, Pandas, Tensorflow, and OpenCV. At this stage, the folder structure, filenames, and class labels

are checked for use in the next preprocessing step.

2. ***Preprocessing Data***

   The first process begins by preparing the dataset to match the CNN model's input format. The steps taken include:

   - Resizing all images to 224x224 pixels.
   - Performing pixel normalization (0–255 to 0–1).
   - Converting labels into one-hot encoding form.
   - Dividing the dataset into two parts: 80% for training and 20% for testing.

3. ***Training CNN Model with MobileNetV2***

   After the data is ready, training is conducted using the MobileNetV2 architecture. This model was chosen because it is lightweight, fast, and accurate for image recognition. The steps performed in this phase include:

   - Adjusting the output layer to support 29 classes.
   - Using the ReLU activation function in the hidden layers and softmax in the output layer.
   - Using the Adam optimizer and categorical crossentropy loss function.
   - Training the model with a specific number of epochs and batch size.

4. ***Model Evaluation***

   After the training process, an evaluation is conducted using the test data to determine the model's performance. The evaluation is done by:

   - Measuring accuracy, precision, recall, and F1-score values.
   - Constructing a confusion matrix to see the classification distribution per class.
   - Displaying accuracy and loss graphs to monitor model stability during training.

5. ***Real-Time Implementation***

   The trained model is then applied to a real-time system using a webcam. This implementation involves:

   - Using OpenCV to capture video input.
   - Defining and extracting the Region of Interest (ROI) of the hand gesture.
   - Applying preprocessing directly to the camera image.
   - Classifying the ASL letter using the trained model.
   - Displaying the prediction directly on the application screen.

System Evaluation

The purpose of the system evaluation is to determine how effective and efficient the developed sign language recognition model is. Two primary methods were used for the evaluation: real-time system implementation testing and quantitative assessment of model performance. The assessment was carried out on test data to determine how well the model could be applied to new data.

Several evaluation metrics used include:

- **Classification Accuracy**: Calculating the percentage of correct predictions out of the total amount of data.
- **Confusion Matrix**: Utilized to assess each ASL letter class's classification

performance. This matrix can be used to determine which classes are most commonly misclassified because it shows the number of accurate and inaccurate predictions for each class.

- **Precision, Recall, and F1-Score**: These three metrics provide a more detailed picture of the model's performance, especially in cases of imbalanced classes. Precision measures the model's accuracy when predicting a class, recall measures how well the model detects all data from a class, and F1-score is the harmonic mean of precision and recall.

- **Frame per Second (FPS)**: FPS is a metric used to assess a system's real-time prediction capability. A high frame rate (FPS) shows that the system can react fast to video input from a webcam, which makes it easier to use interactively.

Visual evaluation is also conducted through training result graphs that show accuracy and loss trends in training and test data. During model training, these graphs can be used to detect overfitting or underfitting. By conducting this comprehensive evaluation, it is hoped that the system built will not only be accurate but also efficient and responsive when used in real-world conditions.

IV. RESULT AND DISCUSSION

A. Model Training

The training process was conducted for 100 epochs with a batch size of 32, using a learning rate of 0.0001. Rotation, flipping, and color jitter are examples of image augmentation techniques used during training to enhance data variation and strengthen the model.

A positive trend can be seen in the training process's outcomes. During the first epoch, the training loss value drastically dropped from a very high level, above 3.0. Towards the end of training, the training loss leveled off and stabilized at approximately 1.0.

Concurrently, there was a notable decrease in the validation loss as well. The validation loss value, which ended at a lower level of about 0.65, was continuously lower than the training loss throughout the procedure.

In terms of accuracy, the training accuracy (Train Acc) began at a very low level (around 15%) and gradually increased, stabilizing at approximately 71%. On the other hand, the validation accuracy (Val Acc) increased much more rapidly and reached a higher level, stabilizing above 80% (ranging from 82-83%).

Interestingly, during the training process, the validation accuracy was consistently higher than the training accuracy, and the validation loss was also consistently lower than the training loss. This is a strong indicator that the model did not experience overfitting and was able to generalize well to the validation test data.

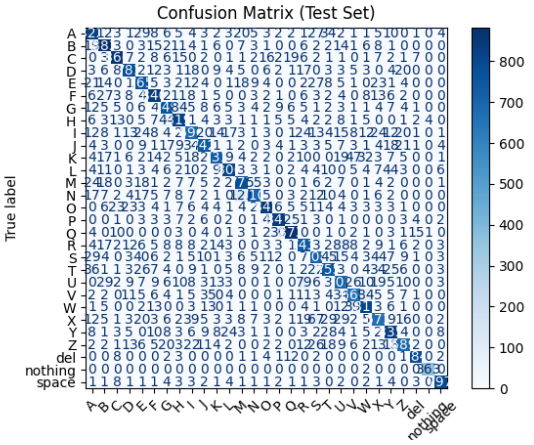

Figure 5. Confusion Matrix on the test dataset with 29 classes

The test results on the ASL Alphabet dataset containing 29 letter and symbol classes are shown in Figure 5. Most classes were recognized well, as indicated by the dominant diagonal values, but there were some classification errors for letters with similar hand shapes.

Evaluation was conducted on the test data using a classification report metric that

includes precision, recall, and F1-score for each letter class. The average evaluation results are shown in Table 3.

Metric	Value (%)
Accuracy	83.53
Precision	83.89
Recall	83.55
F1-Score	83.67

Table 3 Model Evaluation Metrics

The evaluation metrics used in this study follow standard definitions commonly applied in machine learning classification tasks.[23] The mathematical formulas are as follows:

- **Accuracy**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

- **Precision**

$$Precision = \frac{TP}{TP + FP} \times 100\%$$

- **Recall**

$$Recall = \frac{TP}{TP + FN} \times 100\%$$

- **F1-Score**

$$F1 - Score = 2\,\frac{Precision \times Recall}{Precision + Recall} \times 100\%$$

B. Model Evaluation

The quantitative results presented in Section A provide a comprehensive view of the model's performance. According to Table 3, the model's final test accuracy of 83.53% shows that it is generally successful in classifying the ASL gestures from the test set. The validation accuracy was consistently higher than the training accuracy and the validation loss was consistently lower than the training loss, according to an analysis of the training graphs. This is a clear sign that the model did not experience overfitting and performed well when applied to fresh data.

The weighted F1-Score of 83.57% attests to a strong balance between recall (83.53%) and precision (83.89%). This implies that the model is reasonably dependable, reducing both false positives (erroneous letter prediction) and false negatives (inability to recognize the correct letter).

Additionally, the confusion matrix (Figure 5) aids in identifying particular difficulties. It visually confirms that most errors happen between letters with very similar hand shapes, like "M" and "N" or "S" and "T," even though the majority of classes are easily recognized (strong diagonal line). This is an expected challenge in this domain, suggesting that the main cause of classification errors is these particular gestures.

C. Real-Time Implementation

The trained model was saved in the asl_mobilenetv2.pth file and combined with OpenCV. The system was tested using a webcam to show that it could directly receive input. The system is sufficiently responsive, as evidenced by speed testing, which revealed that it operates on a typical laptop with a GPU. However, dim lighting and complex backgrounds made predictions less accurate.

Figure 6 Real-time prediction of ASL letter 'O'

Figure 7 Real-time prediction of ASL letter 'A'

Figure 8 Real-time prediction of ASL letter 'B'

Based on the visual results of the real-time implementation presented in the images, the American Sign Language (ASL) recognition system utilizing the MobileNetV2 architecture and OpenCV is proven to be highly effective. The user's hand is successfully contained within a Region of Interest (ROI) box, demonstrating the system's instantaneous recognition and classification of hand gestures. The Frame per Second (FPS) metric, which is shown directly on the screen and ranges from 18 to 22 FPS (e.g., 19 FPS when detecting the letters 'O' and 'A', and 21 FPS for 'B'), makes MobileNetV2's lightweight performance evident. This speed indicates that the system is very responsive and appropriate for face-to-face communication.

Additionally, the images demonstrate the system's capacity to identify and present hand landmarks (such as the identification of the letter 'A'), suggesting that the deep feature extraction procedure is operating at peak efficiency to differentiate intricate gesture shapes. Overall, these implementation results highlight how an effective model and OpenCV's quick image processing capabilities can be successfully combined to enable real-time ASL classification on common hardware.

D. Discussion

According to the test results, MobileNetV2 showed adequate speed and accuracy in real-time sign language recognition. Because the system can interact with a camera, this study is superior to earlier research that only examined static datasets.

Several significant benefits and drawbacks of this model for real-time application are highlighted by an analysis of the obtained results:

Advantages:

- Computational Efficiency and High Speed: The primary advantage of MobileNetV2 is its lightweight performance. As shown in the implementation results (Figures 6-8), the system is capable of running at 18–22 FPS on a standard laptop. This speed is crucial for a real-time application as it results in a responsive and smooth interaction for the user.
- Good Generalization (Anti-Overfitting): The training graphs showed that the validation loss was lower and the validation accuracy was consistently higher than the training accuracy. This demonstrates that the model did not overfit and can effectively generalize its patterns to fresh, unobserved data, which is crucial for anticipating a user's real-time hand gestures from a webcam.
- High Accuracy on Distinct Gestures: Table 3 shows a respectable overall accuracy of 83.53% for the model. This indicates high reliability for gestures that are visually distinct.

Disadvantages and Limitations:

- Sensitivity to Environmental Conditions: The real-time implementation test results explicitly noted that "dim lighting and complex backgrounds made predictions less accurate". The model can also fail to recognize gestures if other objects are behind it, such as a face. This dependency on good lighting and a clean background is a significant drawback for a real-world, real-time application.
- Difficulty with Similar Gestures: While the overall accuracy is good, the confusion matrix (Figure 5) indicates that most errors occur between letters with highly similar hand shapes (e.g., 'M'/'N', 'S'/'T'). In a real-time application, this would lead

to unstable or frequently incorrect predictions for those specific letters.

- Dependency on Dataset Poses (User Variation): The discussion notes the model is more accurate on hand poses that are similar to the dataset. For real-time implementation, where users may have different signing styles, angles, or hand shapes, this dependency can decrease accuracy.

V. CONCLUSION

This study successfully implemented an American Sign Language (ASL) recognition system based on a Convolutional Neural Network (CNN) with a MobileNetV2 architecture. The model was trained using the ASL Alphabet dataset , which includes 29 classes , with the data divided into 80% for training and 20% for testing. Test results showed a validation accuracy stabilizing around 82-83% (based on the training graphs) and a final test accuracy of **83.53%**. The real-time performance reached 18–22 frames per second (FPS). These findings prove that MobileNetV2 is capable of real-time sign language classification with good accuracy and good computational efficiency. The contribution of this research lies in the application of lightweight CNN for real-time ASL recognition based on hand images, which has the potential to be further developed as an alternative means of communication between deaf people and the general public.

ACKNOWLEDGMENT

The author would like to express his greatest gratitude and deepest appreciation for the assistance that was provided during the completion of this study, "Implementation of MobileNetV2 and OpenCV in a Real-Time Sign Language Recognition System." Without the research community, especially those whose prior work laid the groundwork for lightweight CNN applications and real-time computer vision systems, this study would not have been possible. Furthermore, the successful training and evaluation of the model heavily relied on the extensive ASL Alphabet Dataset, which is available through Kaggle. The author also thanks the institutional support from the Informatics Engineering Study Program, Faculty of Information Technology, Satya Wacana Christian University, for providing the necessary environment and resources to conduct this applied research.

REFERENCES

[1] S. Apendi, C. Setianingsih, and M. W. Paryasto, "Deteksi Bahasa Isyarat Sistem Isyarat Bahasa Indonesia Menggunakan Metode Single Shot Multibox Detector | Apendi | eProceedings of Engineering," *Vol 10, No 1*, vol. 10, no. 1, pp. 249–255, 2023, [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/19322

[2] R. Saputra *et al.*, "Sistem Klasifikasi Alfabet Bahasa Isyarat Indonesia … 237," pp. 237–248, 2025, [Online]. Available: https://data.mendeley.com/datasets/ywnjpbcz8m/1.

[3] I. M. Pramono, Z. Niswati, and A. Agustina, "Model Penerjemah Bahasa Isyarat Indonesia Dengan Metode Convolutional Neural Network (Cnn)," *Semnas Ristek (Seminar Nas. Ris. dan Inov. Teknol.*, vol. 8, no. 01, pp. 1–5, 2024, doi: 10.30998/semnasristek.v8i01.7124.

[4] C. Han, J. Zhang, and H. Wu, "Fall Detection System Based on YOLO Algorithm and MobileNetV2 Model," *Proc. - 2024 10th Int. Conf. Syst. Informatics, ICSAI 2024*, pp. 1–5, 2024, doi: 10.1109/ICSAI65059.2024.10893853.

[5] C. A. Daniel, N. M. Liam, Y. Habchi, and S. Basnet, "Enhancing Language Learning with Real-Time Sign

Language Recognition and Feedback," *URTC 2024 - 2024 IEEE MIT Undergrad. Res. Technol. Conf. Proc.*, pp. 1–5, 2024, doi: 10.1109/URTC65039.2024.10937578.

[6] H. Y. A. Swasono, A. R. Himamunanto, and F. Maedjaja, "Implementasi YOLO11 dan OpenCV untuk Pengenalan Frasa dalam Video Real-Time Bahasa Isyarat Tangan," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 5, no. 3, pp. 1061–1073, 2025, doi: 10.57152/malcom.v5i3.2130.

[7] F. Zaelani and Y. Miftahuddin, "Perbandingan Metode EfficientNetB3 dan MobileNetV2 Untuk Identifikasi Jenis Buah-buahan Menggunakan Fitur Daun," *J. Ilm. Teknol. Infomasi Terap.*, vol. 9, no. 1, pp. 1–11, 2022, doi: 10.33197/jitter.vol9.iss1.2022.911.

[8] A. H. Gustsa and G. S. Permadi, "Sistem Deteksi Bahasa Isyarat Secara Realtime Dengan Tensorflow Object Detection dan Python Menggunakan Metode Convolutional Neural Network," *Inov. J. Ilm. Inov. ...*, vol. 7, no. 2, pp. 1–10, 2023, [Online]. Available: https://ejournal.unhasy.ac.id/index.php/inovate/article/view/4116

[9] Agus Nugroho, R. Setiawan, A. Harris, and Beny, "Deteksi Bahasa Isyarat Bisindo Menggunakan Metode Machine Learning," *J. Process.*, vol. 18, no. 2, pp. 152–158, 2023, doi: 10.33998/processor.2023.18.2.1380.

[10] A. R. Ardiansyah, A. H. Nur'azizan, and R. Fernandis, "Implementasi Deteksi Bahasa Isyarat Tangan Menggunakan OpenCV dan MediaPipe," *Stain. (Seminar Nas. Teknol. Sains)*, vol. 3, no. 1, pp. 331–337, 2024.

[11] J. Eckardt-taing, "Real-Time American Sign Language Recognition Using Machine Learning," *2024 2nd Int. Conf. Artif. Intell. Blockchain, Internet Things*, pp. 1–5, doi: 10.1109/AIBThings63359.2024.10863041.

[12] I. B. A. Peling, I. M. P. A. Ariawan, and G. B. Subiksa, "Deteksi Bahasa Isyarat Menggunakan Tensorflow Lite dan American Sign Language (ASL)," *J. Krisnadana*, vol. 3, no. 2, pp. 90–100, 2024, doi: 10.58982/krisnadana.v3i2.534.

[13] A. Akhilesh, S. Sivakumar, M. A. R, G. Saranya, and K. Kumaran, "Real-Time Sign Language Interpretation and translation to speech Using CUDA and Machine Learning," *2025 Int. Conf. Data Sci. Bus. Syst.*, pp. 1–5, 2025, doi: 10.1109/ICDSBS63635.2025.11031893.

[14] S. Ashillah, S. Adventino Gulo, Y. Rahmi, and D. Yandra Niska, "Implementasi Algoritma Pengolahan Citra Digital untuk Perbaikan Kualitas Gambar," *J. SISKOM-KB (Sistem Komput. dan Kecerdasan Buatan)*, vol. 8, no. 3, pp. 274–280, 2025, doi: 10.47970/siskom-kb.v8i3.823.

[15] J. Ulfah and N. Nurdin, "Implementasi Metode Deteksi Tepi Canny Untuk Menghitung Jumlah Uang Koin Dalam Gambar Menggunakan Opencv," *J. Inform. dan Tek. Elektro Terap.*, vol. 11, no. 3, pp. 420–426, 2023, doi: 10.23960/jitet.v11i3.3147.

[16] G. F. Hanin and T. Dewayanto, "Peran Machine Learning Dan Deep Learning Dalam Pendeteksian Pencucian Uang-a Systematic Literature Review," *Diponegoro J. Account.*, vol. 13, no. 3, pp. 1–11, 2024, [Online]. Available: http://ejournal-s1.undip.ac.id/index.php/accounting

[17] R. Muhammad, M. Y. Andi, and R. Abdul, "Deteksi Citra Daun Untuk Klasifikasi Penyakit PadiMenggunakan Pendekatan Deep

Learning Dengan Model Cnn," *J. Teknol. Terpadu*, vol. 10, no. 1, pp. 56–62, 2024.

[18] R. Indraswari, W. Herulambang, and R. Rokhana, "Deteksi Penyakit Mata Pada Citra Fundus Menggunakan Convolutional Neural Network (CNN)," *Techno.Com*, vol. 21, no. 2, pp. 378–389, 2022, doi: 10.33633/tc.v21i2.6162.

[19] Y. Pratama, E. Rasywir, F. Fachruddin, D. Kisbianty, and B. Irawan, "Eksperimen Layer Pooling menggunakan Standar Deviasi untuk Klasifikasi Dataset Citra Wajah dengan Metode CNN," *Build. Informatics, Technol. Sci.*, vol. 5, no. 1, pp. 200–210, 2023, doi: 10.47065/bits.v5i1.3604.

[20] L. F. S. Scabini and O. M. Bruno, "Structure and performance of fully connected neural networks: Emerging complex network properties," *Phys. A Stat. Mech. its Appl.*, vol. 615, p. 128585, 2023, doi: 10.1016/j.physa.2023.128585.

[21] E. Shavna Gracia, N. Anisa Sri Winarsih, and D. Nuswantoro, "Comparison of VGG16, MobileNetV2, InceptionV3, ResNet50, and Custom CNN Architectures for Furniture Image Classification," *Infotekmesin*, vol. 16, no. 01, pp. 24–30, 2025, doi: 10.35970/infotekmesin.v16i1.2500.

[22] J. J. Pangaribuan, R. Romindo, A. R. Mitra, O. P. Barus, and S. Prashanth, "American Sign Language Alphabet Recognition Using Convolutional Neural Network," *Proc. - 2024 2nd Int. Conf. Technol. Innov. Its Appl. ICTIIA 2024*, pp. 1–6, 2024, doi: 10.1109/ICTIIA61827.2024.10761663.

[23] C. Miller, T. Portlock, D. M. Nyaga, and J. M. O. Sullivan, "A review of model evaluation metrics for machine learning in genetics and genomics," no. September, pp. 1–13, 2024, doi: 10.3389/fbinf.2024.1457619.