CITIZEN: Jurnal Ilmiah Multidisiplin Indonesia
Vol. 6 No. 1, 2026
ISSN: 2807-5994
https://journal.das-institute.com/index.php/citizen-journal

GESTURE TO SPEECH TRANSFORMATION: A REAL-TIME SIGN LANGUAGE ALPHABET RECOGNITION SYSTEM FOR DIGITAL INCLUSION OF THE SPEECH IMPAIRED COMMUNITY

Tigus Juni Betri
Informatics Department, UIN Raden Mas Said Surakarta
Corresponding email : tigusjuni.betri@staff.uinsaid.ac.id

DOI: 10.53866/jimi.v6i1.1247

Abstract

Access to communication is a fundamental right of every individual, including the speech-impaired community, which often faces limitations in social interaction. This study designs a real-time system that transforms sign language alphabet gestures into speech by utilizing Computer Vision and Deep Learning technologies. A sign language alphabet dataset is processed using a Convolutional Neural Network (CNN) to recognize visual hand patterns representing letters A–Z. The trained model is then integrated with OpenCV and Mediapipe for real-time hand gesture detection and connected to a speech synthesis engine so that the recognized letters can be automatically spoken. The results demonstrate the system's potential as a basic communication bridge that supports digital inclusion for the speech-impaired community. From a sustainable development perspective, this innovation is relevant to SDG 4 (Quality Education) and SDG 10 (Reduced Inequalities), as it enables more equitable, inclusive, and sustainable social interaction in the era of digital transformation. This innovation can also serve as a foundation for further development toward an automatic sign language translator capable of recognizing full words and complete sentences. Consequently, this system has strong potential to become a practical solution for ensuring equal access to communication across various sectors, including education, public services, and the workplace.

Keywords: CNN; Computer Vision; Sign Language; Speech-Impaired; Sustainable Development

1. Introduction

Communication is an essential element of human life as the primary means of conveying information, ideas, and emotions. However, for the speech-impaired community, limitations in verbal communication often create barriers to interacting with the wider society. Sign language serves as their main medium of communication; unfortunately, not everyone is able to understand or use it. This condition has the potential to create social disparities and reduce opportunities for speech-impaired individuals to gain access to education, employment, and public services (Haida & Wahyuningsih, 2024). The development of artificial intelligence (AI) and digital image processing (computer vision) technologies has created significant opportunities to provide inclusive solutions. One such solution is a sign language letter detection system based on Convolutional Neural Networks (CNN) that is capable of recognizing hand gestures and converting them into alphabetic characters. If the detected letters can be displayed visually and simultaneously pronounced automatically using a speech synthesis engine, a more universal communication medium that is easily understood by the general public can be achieved (Azmi et al., n.d.).

Based on these considerations, this study focuses on developing a system capable of detecting sign language letters by utilizing computer vision and deep learning, and then converting the detection results into speech in real time. This research is expected to address the need for alternative communication media for the speech-impaired community while simultaneously supporting the realization of digital inclusion in society (Raup et al., n.d.). The main objective of this study is to develop a sign language letter detection model using Convolutional Neural Networks (CNN), to build a real-time application that displays the detection results and simultaneously converts them into speech, and to provide a tangible contribution to the development of inclusive-friendly technology. Through this research, the benefits obtained are not only academic, by enriching the body of knowledge in the field of

informatics, but also practical, by providing an alternative communication medium for the speech-impaired community. From a social perspective, this study is expected to support the sustainable development agenda, particularly SDG 4 on quality education and SDG 10 on reducing inequalities, thereby helping to create a communication environment that is more just, equitable, and sustainable ("Abdimas Langkanae Jurnal Pengabdian Kepada Masyarakat," 2023).

The literature review serves to provide a theoretical foundation and an overview of previous studies relevant to the topic of sign language letter detection. This research is based on several key concepts, namely inclusive communication, sign language, computer vision, deep learning, and sustainable development. Sign language is a form of visual communication that uses hand movements, facial expressions, and body posture to convey messages. For the speech-impaired community, sign language is the primary medium of communication. However, the general public's limited ability to understand sign language often creates barriers in everyday interactions. Therefore, a technology-based system is needed to bridge communication between the speech-impaired community and the wider society (Mutmainah, 2025). Advances in the field of computer vision have enabled computers to recognize objects, patterns, and movements through digital image processing. One of the most widely used methods is the Convolutional Neural Network (CNN), a deep learning architecture that has proven effective in recognizing visual patterns. CNN is capable of extracting features from hand images and classifying them into specific categories, such as alphabet letters in sign language (Muhammad & Yulianto, n.d.) .

Previous studies have explored sign language recognition using various approaches. Some research employed manual feature extraction methods such as Histogram of Oriented Gradients (HOG) or Scale-Invariant Feature Transform (SIFT); however, these approaches tend to be less flexible in handling variations in gestures. With technological advancements, CNN has become the dominant method due to its ability to perform automatic feature extraction. Recent studies indicate that CNN can achieve high accuracy in classifying sign language alphabets, particularly when using the American Sign Language (ASL) dataset (Ahmad Fariz Fuady et al., 2025). In addition to CNN, speech synthesis technology also plays an important role in this study. By integrating a text-to-speech engine such as pyttsx3, the detected letters can be automatically spoken, allowing visual messages to be transformed into audio messages. This approach enables a more inclusive two-way communication medium, as people who do not understand sign language can still receive the message in spoken form (I Komang Setia Buana, 2020). From the perspective of sustainable development, this research is closely related to the principles of inclusivity and the reduction of inequalities. Sign language detection technology supports equal access to education for the speech-impaired community, creates opportunities for participation in the workforce, and strengthens social integration. Therefore, this study not only contributes to the advancement of information technology but also has social impacts that align with the goals of sustainable development. (Gholib Muzakki et al., n.d.).

2. Method

This study employs an experimental approach by utilizing Computer Vision and Deep Learning technologies to detect sign language letters and convert them into speech in real time (Nugroho et al., 2023). The research process is carried out through several interrelated stages, ranging from dataset preparation to the implementation of the detection system (Okpatrioka, 2023).

The initial stage involves setting up the development environment using the Python programming language executed in Visual Studio Code. To support this study, several essential libraries are used, including OpenCV for image processing, MediaPipe for hand landmark detection, TensorFlow for building and training the Convolutional Neural Network (CNN) model, and *pyttsx3* for converting the detected text into speech (Aditya Perdana et al., 2024). All libraries are installed within a virtual environment to ensure a more controlled experimental process (Aprilisa & Aulia, 2024).

Next, dataset preparation is conducted using a collection of sign language images representing the alphabet from A to Z. The dataset is obtained from public sources such as Kaggle and then organized into training and validation folders. The dataset is processed through a normalization stage so that the images have uniform dimensions and standardized pixel values, making them suitable for training the CNN model.

The next stage is the development of the CNN model. The model is designed with several convolutional layers to extract image features, followed by pooling layers to reduce dimensionality, and fully connected layers to produce letter predictions. The training process is conducted over a number of epochs until the model achieves optimal accuracy, while its performance is evaluated using validation

data. The trained model is then saved in h5 format so that it can be used in the implementation stage (Subur et al., 2024).

The system implementation stage is carried out by integrating a camera using OpenCV. Each frame captured by the camera is processed with MediaPipe to detect hand positions and generate a bounding box that isolates the hand region. The extracted hand image is then preprocessed and fed into the CNN model to be predicted as one of the letters A–Z (Syaddam et al., 2024). The prediction results are displayed visually on the computer screen and simultaneously converted into speech using pyttsx3. Thus, the system is able to provide real-time feedback in both text and audio forms (Andini et al., 2023).

Overall, this method is designed to provide a simple communication solution for the speech-impaired community. By detecting sign language letters and converting them directly into speech, this study is expected to serve as an initial bridge to support more inclusive and sustainable interaction within society.

3. Results and Discussion

This section presents the experimental stages in a systematic manner, including environment setup, dataset preparation, model development and training, model evaluation, real-time inference implementation (program), field testing, performance analysis, encountered challenges, and recommendations for improvement.

3.1 Research Results

3.1.1 Environment Preparation

Before the experiment begins, a stable environment must be prepared as follows::

1. Install Python version 3.10 or 3.11 (not versions 3.12 or 3.13, as some packages such as MediaPipe and TensorFlow do not yet support them).
2. Create and activate a virtual environment:
   A virtual environment in Python is an isolated working environment that allows a Python program project to be executed with its own specific dependencies and configurations

```
python -m venv venv
.\venv\Scripts\activate   # Windows PowerShell: .\venv\Scripts\Activate
```

3. Install dependencies
   Dependencies in Python are external libraries or packages required by a program to function properly.

```
pip install --upgrade pip
pip install opencv-python mediapipe tensorflow pyttsx3 matplotlib scikit-learn
```

3.1.2 Dataset Preparation

This step is essential to ensure that the model generalizes well:

1. Folder Structure
   The folders are used to store training data and validation data. Training data are used to enable the model to recognize differences among images. For example, if the task is to detect cat images, the training data are used to distinguish whether the detected data are more similar to cats or to other animals. Validation data, on the other hand, are used to evaluate the model's performance during the training process. In this study, the dataset is organized according to sign language letters A, B, C, and so on, with each letter stored in its corresponding folder. Each letter category contains more than 3,000 images

```
dataset/
  train/
    A/ B/ ... Z/
  val/
    A/ B/ ... Z/
```

2. Check the number of images per class. Balance them if necessary, using Python code

```
import os
for split in ["train","val"]:
    for cls in sorted(os.listdir(f"dataset/{split}")):
        print(split, cls, len(os.listdir(f"dataset/{split}/{cls}")))
```

3. Apply data augmentation to increase variability (lighting, rotation, zoom, and shift). An example using ImageDataGenerator is as follows:

```
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
```

3.1.3 Model Development and Training (Experiment in Python)

Technical steps and experimental decisions:

1. Model Architecture

```
model = Sequential([
    Conv2D(32,(3,3),activation='relu',input_shape=(64,64,3)),
    BatchNormalization(),
    MaxPooling2D(),
    Conv2D(64,(3,3),activation='relu'),
    BatchNormalization(),
    MaxPooling2D(),
    Conv2D(128,(3,3),activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(256,activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
```

2. Tested Hyperparameters (50 epochs). An epoch is a complete pass of the entire training dataset through the training process. In this study, training was conducted for 50 epochs, which took approximately 12–13 hours to complet.

```
early = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(train_ds, validation_data=val_ds, epochs=50, callbacks=[early])
model.save("asl_model.h5")
```

3.1.4 Model Evaluation (after training)

Performance exploration steps:

1. Validation accuracy and loss: obtained from history.history['val_accuracy'] and history.history['val_loss'].
2. Confusion matrix (to identify which classes are most frequently misclassified):

```
from sklearn.metrics import confusion_matrix, classification_report
y_pred = model.predict(val_images)       # array softmax
y_pred_labels = np.argmax(y_pred, axis=1)
print(classification_report(y_true, y_pred_labels, target_names=class_names))
cm = confusion_matrix(y_true, y_pred_labels)
```

Analysis of the confusion matrix reveals letters that are commonly confused with one another (e.g., M vs. N).

3. Per-class accuracy: this is important for identifying whether certain letters require more training data.

3.1.5 Real-Time Inference Implementation

This stage evaluates the model in a real-world pipeline (detect_asl.py):

1. Open the camera (OpenCV) and read frames sequentially.

2. Perform hand detection (MediaPipe Hands), which produces 21 landmarks; a bounding box is computed from these landmarks.
3. Preprocessing: crop the hand region → cv2.resize(..., (64,64)) → normalization by dividing by 255.0 → np.expand_dims.
4. Make predictions using the saved model:

```
pred = model.predict(hand_img)        # (1, num_classes)
label = class_names[np.argmax(pred)]
```

5. **Output stability (filtering):** use a buffer or majority voting so that speech is produced only when the prediction is stable:

```
from collections import deque
buf = deque(maxlen=10)
buf.append(label)
if buf.count(label) >= 8 and label != last_spoken:
   # speak using pyttsx3 (prefer non-blocking)
```

6. **Text-to-Speech:** *pyttsx3* is recommended to be executed in a separate thread so that it does not block the camera loop. An example of non-blocking usage is as follows:

```
import threading
def speak(text): engine.say(text); engine.runAndWait()
threading.Thread(target=speak, args=(label,), daemon=True).start()
```

3.1.6 Conducted Experiments and Testing Procedures

1. Experiment A (baseline): a simple architecture, epochs = 15, without data augmentation → record training and validation accuracy.
2. Experiment B (augmentation): apply data augmentation → examine changes in validation accuracy.
3. Experiment C (more epochs + EarlyStopping): epochs = 50 with early stopping → observe whether validation accuracy increases or overfitting occurs.
4. Experiment D (transfer learning): use MobileNetV2 pretrained on ImageNet and then fine-tune the model → this a pproach typically improves accuracy with limited data.
5. Field testing: run detect_asl.py in environments with different lighting conditions, different users, and different backgrounds → record correct and incorrect predictions for each letter.**3.1.7** How to Run the System:
   1. Training: python train_model.py → check the per-epoch training output and the generated file asl_model.h5.
   2. Inference: python detect_asl.py → perform real-time testing; record latency and whether the spoken output is correct

3.1.7 Performance Anlysis and Practical Metrics

1. Validation accuracy is used as the primary benchmark; val_loss is also monitored.
2. The confusion matrix reveals pairs of classes that are frequently confused. These results can be used to add more data or apply augmentation to weaker classes.
3. Latency / FPS: measure the prediction time per frame (time.time() before and after model.predict)-ideally, prediction should be under 100 ms so that the user interface feels real-time. If the system is too slow, reduce the input size, use a lightweight model (e.g., MobileNet), or utilize a GPU.
4. Speech stability: buffering or majority voting helps reduce noise and prevents repeated or unstable speech output.

3.1.9 Experimental Results

Below is the training dataset folder. Each class is organized into folders labeled A–Z. Each folder contains images of the corresponding sign language gestures, with approximately 3,000 images per letter.

CITIZEN: Jurnal Ilmiah Multidisiplin Indonesia
Vol. 6 No. 1, 2026
ISSN: 2807-5994
https://journal.das-institute.com/index.php/citizen-journal

figure 1. Dataset Folder

Figure 2. Picture Letter A

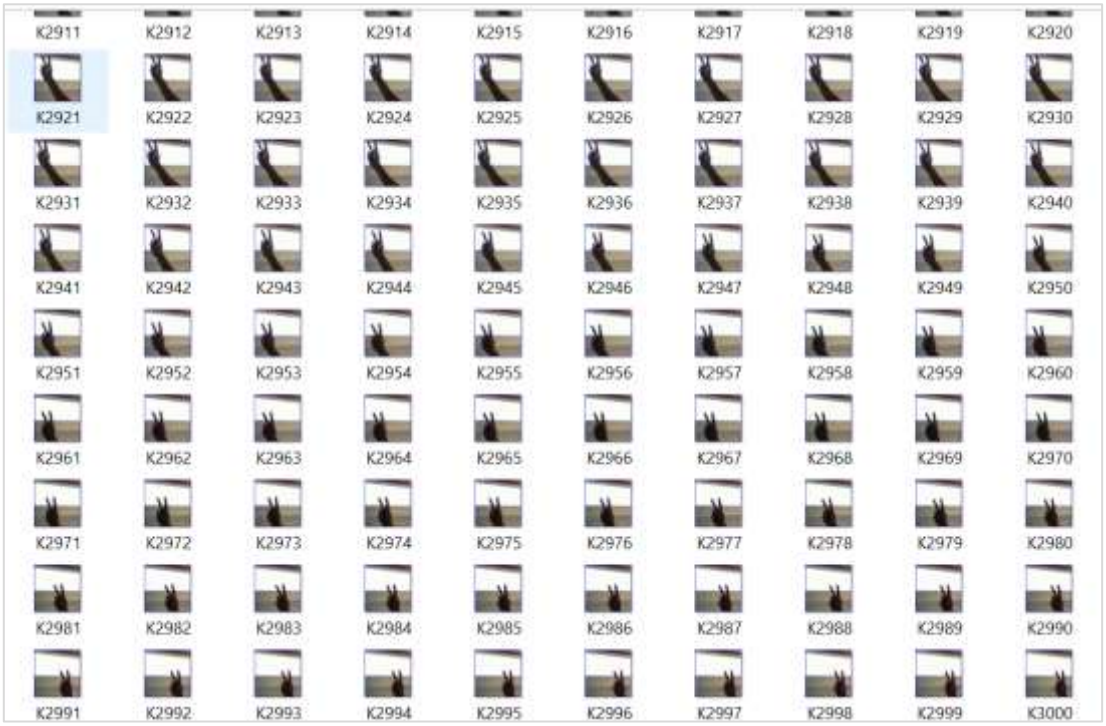

Figure 3. Picture Leter K

Below is the implementation of the system using Python with the Visual Studio Code software editor. The program consists of the following files:

1. rain_model.py: this file is used to train the deep learning model.
2. test_hand.py: this file is used to test the model using static hand image inputs.
3. test_camera.py: this file is used to test the model using a camera/webcam in real time.
4. asl_model.h5: this file contains the trained model generated from the training process.
5. detect_asl.py: this file serves as the main program for detecting hand gestures.

In terms of workflow, train_model.py is used to train the model, and the resulting model is saved as asl_model.h5. The trained model is then tested using static images with test_hand.py and evaluated in real-time using a camera with test_camera.py. If all tests run successfully, the main execution is carried out using detect_asl.py, which loads the model from asl_model.h5 to perform hand gesture detection.

CITIZEN: Jurnal Ilmiah Multidisiplin Indonesia
Vol. 6 No. 1, 2026
ISSN: 2807-5994
https://journal.das-institute.com/index.php/citizen-journal

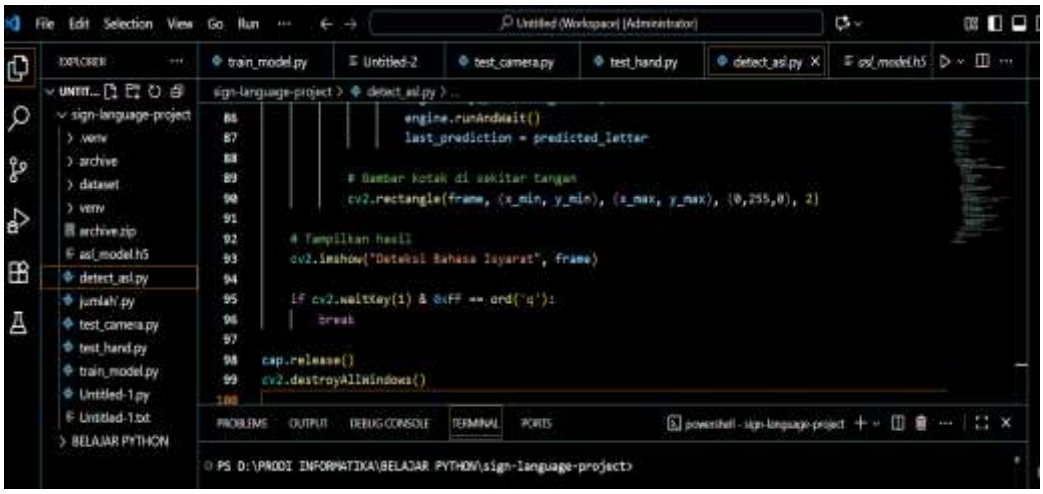

Figure 4. Source code view with Visual Studio Code

After executing the detect_asl.py file, the program will automatically activate the camera. The hand gestures performed in front of the camera will be detected by the system and spoken aloud through the speaker. The following figure shows an example of the execution result:

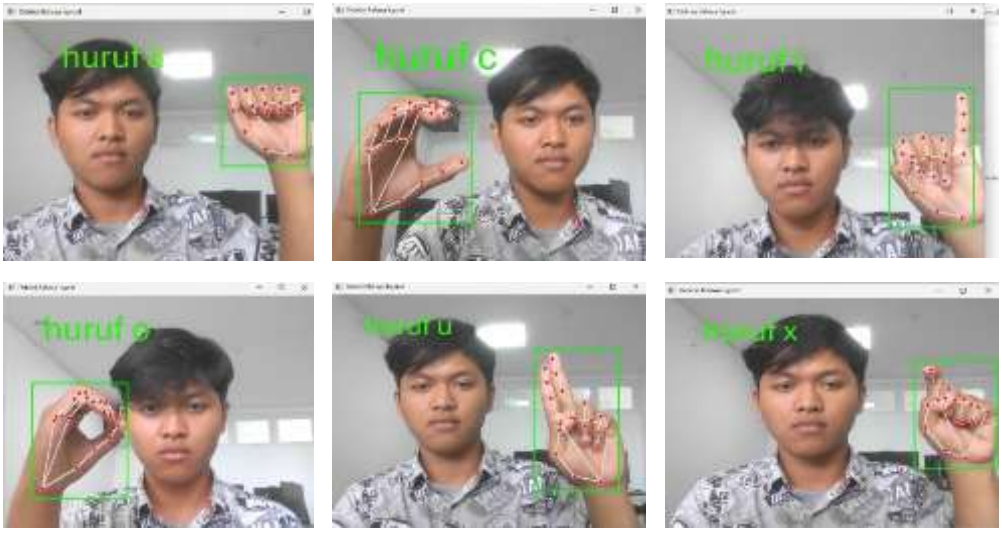

Figure 5. Execution Result

3.2 Discussion

The sign language letter detection program developed in this study using Python can be executed properly and has demonstrated results that meet the initial expectations. The system is capable of recognizing letters from sign language (ASL) through the training of a deep learning model based on Convolutional Neural Networks (CNN) and its application in various testing scenarios. This indicates that the integration of Python with machine learning libraries such as TensorFlow/Keras and computer vision libraries such as OpenCV provides a strong foundation for building sign language recognition applications.

During the training stage, the model was constructed using the train_model.py script and a dataset of hand images representing sign language letters. This training process produced a trained model that was saved in the asl_model.h5 file. From a series of experiments conducted, it was found that the model is able to recognize letters reasonably well; however, the achieved accuracy is strongly influenced by the number of epochs used during training. The greater the number of epochs applied, the better the model's ability to recognize hand image patterns, as the neural network weights are repeatedly updated to minimize prediction errors

However, a consequence of using a high number of epochs is the increased demand for computational time and hardware resources, especially when training is performed on computers with limited specifications. This results in a trade-off between training speed and model accuracy. When training is conducted with a low number of epochs, the model can be trained quickly, but it tends to suffer

from underfitting and yields suboptimal accuracy. Conversely, training with a high number of epochs can produce a model with better accuracy, but it requires several hours to complete, particularly when a GPU is not used.

Further evaluation was conducted using two main methods. First, the test_hand.py script was used to test the model with static image inputs. The results indicate that the model is able to predict sign language letters from a single image, although the accuracy is still limited under certain conditions, such as lighting and hand positions that are consistent with the training dataset. Second, the test_camera.py script was employed to test the model in real time using a camera or webcam. The results show that the system is capable of capturing the user's hand movements directly, performing preprocessing, and then generating letter predictions that are displayed on the screen.

The system's ability to operate in real time through a camera represents a significant achievement, as it demonstrates that the trained model can be integrated into practical and interactive applications. Nevertheless, challenges arise when the data acquisition conditions differ from those of the training data, such as low lighting, complex backgrounds, or hand positions that do not match the training samples. Under such conditions, the system's accuracy decreases and the predictions are not always reliable.

In addition to the number of epochs and dataset conditions, another limitation concerns the number of letter classes evaluated. In this study, the model focuses on sign language letters (ASL) only and does not yet cover full words or sentences. Thus, although the system can recognize letters individually, users still need to manually assemble them into words. This poses a particular challenge if the system is to be further into a more comprehensive sign language translation tool.

Overall, this discussion demonstrates that the Python-based sign language letter detection program has operated as designed, albeit with several limitations. The program is capable of detecting ASL letters in both images and real-time video; however, it still requires a high number of epochs for the model to achieve strong accuracy. The main challenges lie in dataset limitations, computational time, and the model's sensitivity to environmental conditions. With further improvements and development, this system has significant potential to be transformed into a more complete, accurate, and user-friendly sign language translation application for the broader community.

3.2.1 Keterbatasan eksperimen

1) The developed system is limited to detecting static letters (spell-by-letter) and does not yet support full words or sentences or dynamic gestures (e.g., J and Z, which require temporal tracking).
2) The system is sensitive to lighting conditions, skin color, hand size, and background. Therefore, a smaller framing area and a higher number of training epochs are required to improve performance.
3) The training data (e.g., ASL datasets from Kaggle) may not adequately represent local variations in environment, culture, and gesture styles.

4. Conclusion

Research and implementation of sign language letter detection using Python demonstrate that advances in artificial intelligence, particularly in the fields of computer vision and deep learning, can make a tangible contribution to supporting inclusive communication. Sign language is the primary means of communication for the deaf and mute community; however, the general public's limited understanding of sign language often becomes a barrier. Therefore, a Python-based sign language letter detection system serves as an alternative solution to bridge this communication gap.

In its implementation, the system is developed through several integrated stages. First, a model training process is carried out using the train_model.py script, which utilizes a dataset of sign language letters. The model is trained using a deep learning algorithm based on Convolutional Neural Networks (CNN), which has proven effective in recognizing visual patterns from hand images. The result of this training process is a model file with the .h5 extension (asl_model.h5), which stores the model's weights and architecture.

Next, the testing stage was conducted using two approaches. Static image–based testing using test_hand.py demonstrates how the model can recognize sign language letters from a single image after undergoing preprocessing. Meanwhile, real-time video–based testing using test_camera.py shows the system's ability to capture hand gestures directly from the camera and generate predictions dynamically. The final stage is implemented in detect_asl.py as the main program that integrates all functions, enabling users to perform sign language letter detection in a practical manner, either from images or via a camera.

From the series of experiments, it can be concluded that the Python-based sign language letter detection system operates effectively in recognizing ASL (American Sign Language) letters. The model is capable of performing classification with a satisfactory level of accuracy and providing rapid responses in real-time applications. This indicates that the integration of Python with libraries such as TensorFlow/Keras and OpenCV has strong potential for developing inclusive artificial intelligence–based applications.

Overall, this study demonstrates that sign language detection technology can serve as an important step toward supporting the Sustainable Development Goals (SDGs), particularly Goal 4 on Quality Education and Goal 10 on Reduced Inequalities. This system is not only applicable for academic and educational purposes but also has the potential to be further developed into an automatic sign language translation application that can be deployed on mobile devices and digital communication platforms. With further development—such as incorporating full word or sentence detection, integrating text-to-speech (TTS) technology, and expanding dataset diversity—this system is expected to become an increasingly useful solution for creating barrier-free communication for all communities.

Bibliografi

Langkanae, A. (2023). Abdimas Langkanae. *Jurnal Pengmas - Pendidikan Bahasa Dan Budaya*, *1*(1). https://pusdig.web.id/index.php/abdimas/index

Aditya Perdana, H., Lailiyah, S., & Wahyuni, dan. (2024). *Rancang Bangun Sistem Terjemahan Bahasa Isyarat Berbasis Web Secara Real-Time Menggunakan Tensorflow*. *2*. https://doi.org/10.46984/sebatik.v28i2.0000

Ahmad Fariz Fuady, Dwiky Oldi Amsyah, Muhammad Farhan, Rusma Riansyah, & M. Dayyan Dhiyaul Haq. (2025). Implementasi Algoritma Convolutional Neural Network (CNN) untuk Pengenalan dan Klasifikasi Buah Berdasarkan Citra Digital. *Jurnal Publikasi Ilmu Komputer Dan Multimedia*, *4*(2), 148–159. https://doi.org/10.55606/jupikom.v4i2.4116

Andini, T. D., Arifin, J., . S., Irsyada, A. E., & Indahsari, R. D. (2023). Pelatihan Pemrograman Bahasa Python Pada Jurusan Perangkat Lunak Dan GIM SMKN 12 Malang. *Jurnal Pengabdian Masyarakat - Teknologi Digital Indonesia.*, *2*(2), 42. https://doi.org/10.26798/jpm.v2i2.880

Aprilisa, S., & Aulia, R. (2024). Penerapan Metode Prototype dalam Pengembangan Sistem Informasi Inventory Barang Berbasis Web. *Jurnal Teknik Industri Terintegrasi*, *7*(1), 333–340. https://doi.org/10.31004/jutin.v7i1.24749

Azmi, K., Defit, S., & Putra Indonesia YPTK Padang Jl Raya Lubuk Begalung-Padang-Sumatera Barat, U. (n.d.). *Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat*. *16*(1), 2023.

Gholib Muzakki, A., Pratiwi, A., & Kumala, F. N. (n.d.). *Pada Anak Penyandang Difabel Tunawicara*. http://journal.unesa.ac.id/index.php/paramasastra

Haida, N., & Wahyuningsih, N. (2024). Implementasi Sustainable Development Goals (SDGS) Di Indonesia Perspektif Ekonomi Islam. *Jurnal Ilmu Ekonomi Dan Keislaman*, *11*, 108–107.

I Komang Setia Buana. (2020). Implementasi Aplikasi Speech to Text untuk Memudahkan Wartawan Mencatat Wawancara dengan Python. *Jurnal Sistem Dan Informatika (JSI)*, *14*(2), 135–142. https://doi.org/10.30864/jsi.v14i2.293

Muhammad, R., & Yulianto, S. (n.d.). Penerapan Pemrograman Python Dalam Menentukan Waktu Overhoul Kondensor Turbin UAP Application of Python Programming in Determining the Overhoul Steam Turbine Condenser. *Jurnal Konversi Energi Dan Manufaktur*, *8*.

Mutmainah, N. (2025). Implementasi Deep Learning pada Pembelajaran Matematika. *Jurnal Ilmiah Pendidikan Dasar*, *10*(1).

Nugroho, A., Setiawan, R., Harris, A., Dinamika Bangsa, U., & Jendral Sudirman, J. (2023). *Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer Deteksi Bahasa Isyarat Bisindo Menggunakan Metode Machine Learning*. *18*(2). https://doi.org/10.33998/processor.2023.18.2.1308

Okpatrioka. (2023). Research and Development - Penelitian yang Inovatif. *Jurnal Pendidikan Dan Bahasa*.

Raup, A., Ridwan, W., Khoeriyah, Y., Yuliati Zaqiah, Q., & Islam Negeri Sunan Gunung Djati Bandung, U. (n.d.). *Deep Learning dan Penerapannya dalam Pembelajaran*. http://Jiip.stkipyapisdompu.ac.id

Subur, J., Taufiqurrohman, M., Novian Reza Al Hafizh, dan, Studi Teknik Elektro, P., Teknik dan Ilmu Kelautan, F., Hang Tuah Jl Arif Rahman Hakim No, U., Surabaya, S., & Timur, J. (2024).

CYCLOTRON : Jurnal Teknik Elektro Pemanfaatan Teknologi Computer Vision untuk Deteksi Ukuran Ikan Bandeng dalam Membantu Proses Sortir Ikan. *Jurnal Teknik Elektro* .

Syaddam, S., Soeksin, S. D., & Nizar, R. (2024). Teknologi Computer Vision untuk melakukan Deteksi dan Penentuan Kualitas Bibit Ayam Day Old Chicks. *Jurnal Informatika: Jurnal Pengembangan IT*, *9*(3), 296–305. https://doi.org/10.30591/jpit.v9i3.7923