Sign Language Recognition Using Convolutional Neural Networks

Sohela Showrin
ID: 2104010202199
Department of Computer Science & Engineering
Chattogram, Bangladesh

Rahin Toshmi Ohee
ID: 2104010202204
Department of Computer Science & Engineering
Chattogram, Bangladesh

Jahirul Islam Rakib
ID: 2104010202209
Department of Computer Science & Engineering
Chattogram, Bangladesh

Durjoy Nath
ID: 2104010202222
Department of Computer Science & Engineering
Chattogram, Bangladesh

Abstract—**Sign language is an essential communication method for individuals with hearing impairments, enabling them to express thoughts, emotions, and needs effectively. However, a significant communication gap exists between sign language users and non-users. This paper presents a deep learning-based system that converts American Sign Language (ASL) gestures into corresponding letters using a highly optimized Convolutional Neural Network (CNN). The model is trained on a dataset containing 26 classes (A-Z) and achieves 99.03% of training accuracy and 90.03% testing accuracy, demonstrating its near-perfect precision in recognizing ASL hand gestures. The system preprocesses images, applies normalization, and performs translation, making communication more accessible. By optimizing hyperparameters and implementing batch normalization, dropout techniques ,one hot encoding, the model ensures high generalization across various hand gestures. This approach enhances daily interactions for individuals with hearing impairments and promotes inclusivity in social, educational, and professional settings. The project highlights the effectiveness of deep learning in bridging communication barriers and improving accessibility through technology.**

Index Terms—**Sign Language Recognition, Convolutional Neural Networks (CNN), American Sign Language (ASL), Accuracy, Categorical Crossentropy, Model Evaluation, Communication Barriers, Hearing Impairments.**

I. Introduction

Communication is a vital part of our daily lives. It helps us share our thoughts, feelings, and information with others. But for people who have trouble hearing or speaking, communication can be challenging. Many of us don't understand sign language, which creates a gap between those who use it and those who don't. To help close this gap, our project plans to create a system that can recognize sign language gestures. We'll use a type of deep learning called Convolutional Neural Networks (CNNs) to do this. Our system will take images of hand gestures and turn them into letters. This will make it easier for sign language users to communicate with people who aren't familiar with sign language. It will also help beginners learn sign language more easily.

CNNs are especially good for this task. They can automatically pick out important features from images, like the movements of hands and facial expressions. These are key parts of sign language. By analyzing these features, CNNs can accurately identify different gestures and translate them into corresponding letters or words. We are motivated by a strong desire to help the millions of people around the world who have hearing or speech impairments. These individuals often feel alone or frustrated because of communication barriers. This affects their personal and work lives. For example, in many communities, deaf people face challenges in accessing services and participating in social activities due to a lack of awareness and understanding of their needs. Key aspects of our project include:

- **Utilizing Convolutional Neural Networks (CNNs):** CNNs are especially good for this task. They can automatically pick out important features from images, like the movements of hands, which are key parts of sign language.
- **Addressing Communication Barriers:** We are motivated by a strong desire to help the millions of people around the world who have hearing or speech impairments. These individuals often feel alone or frustrated because of communication barriers, affecting their personal and work lives.
- **Enhancing Social Inclusion:** In many communities, deaf people face challenges in accessing services and participating in social activities due to a lack of awareness and understanding of their needs.
- **Educational Resource:** Our system can serve as a tool for beginners learning sign language, allowing them to practice and receive immediate feedback, thus enhancing their learning experience.
- **Real-World Application of Deep Learning:** This project demonstrates how deep learning can be used to solve real-world problems.

By using CNNs' ability to recognize visual patterns, we aim to build a user-friendly tool. This tool will improve communication for those who rely on sign language. Our project not only aims to make daily interactions better for people with hearing and speech disabilities but also shows how deep learning can be used to solve real-world problems. For instance, similar approaches have been used to develop systems that recognize American Sign Language gestures with high accuracy, demonstrating the potential of CNNs in this field. In addition to bridging communication gaps, our system can serve as an educational resource. Beginners learning sign language can use it to practice and receive immediate feedback, enhancing their learning experience. This dual functionality makes our project beneficial not only for those with hearing or speech impairments but also for anyone interested in learning sign language.

II. Related Work

Our project aims to create a sign language prediction system using Convolutional Neural Networks (CNNs) with Python and TensorFlow/Keras. The goal is to recognize American Sign Language (ASL) signs from images or videos. The model will focus on real-time prediction and be optimized for mobile devices. We will use data preprocessing methods like resizing, normalization, and data augmentation (flipping, rotation, and contrast adjustments). To make it work on mobile devices, we will use TensorFlow Lite and OpenCV for video detection. Kasapbaşı et al. (2022) [1] trained a CNN model on an ASL dataset with different lighting and distances, reaching 99.38helps improve accuracy. We will also use data augmentation but focus on real-time processing with OpenCV. Recent progress in sign language recognition has used transfer learning and CNNs to improve accuracy and speed. Sadik *et al.* [2] developed a system for recognizing Bangla sign language using transfer learning.This method combines pre-trained models with specific data to improve recognition without needing large datasets. This is especially helpful for languages with fewer available data.

In addition, Indian Sign Language recognition has also improved with machine learning. Katoch *et al.* [3] used SURF along with SVM and CNNs to recognize ISL signs. This technique works better than older methods by combining spatial features with deep learning models to improve performance.Barbhuiya *et al.* [4] focused on CNN-based feature extraction and classification for sign language. Their research shows how CNNs can learn important details from raw image data, making the recognition system faster and more accurate. Rastgoo *et al.* [5] reviewed many methods for sign language recognition and suggested improvements, including combining different data sources like vision, motion capture, and sensors. Current systems still face challenges. Murali et al. (2022) [6] built a real-time ASL system using HSV color segmentation and OpenCV for feature extraction. Their method was fast and used less computing power. We will use a similar real-time approach but improve accuracy with a custom CNN designed for mobile use. Garcia & Viesca (2020) [7] used transfer learning with GoogLeNet for better accuracy. However, transfer learning needs high computing power, which is not ideal for mobile devices. We will develop a lightweight CNN that works well in real-time on mobile phones. Das et al. (2020) [8] focused on static ASL signs and got 94.34validation accuracy. Their model worked well but could not recognize moving signs. Our model will work with both static and moving signs for better use. Huang et al. (2014) [9] used 3D CNNs with Kinect depth data for better ASL accuracy. Their method needed special hardware, but we will use 2D CNNs with normal cameras to make our system more accessible. Rathia et al. (2019) [10] used ResNet50 for ASL recognition and got 99.03will create a lightweight CNN for real-time mobile applications. Misra et al. (2011) [11] used Histogram of Oriented Gradients (HOG) with CNNs to improve recognition. HOG features are useful, but we will focus on deep learning-based feature extraction for better accuracy. Dong et al. (2015) [12] used a Kinect-based ASL system with depth and color data. It performed well but required extra hardware. Our system will work with regular cameras to make it easier to use and cheaper. Kala et al. (2014) [13] used Sparse Autoencoders for learning ASL features. While autoencoders are useful, CNNs are better for large ASL datasets. We will use CNNs for handling complex ASL signs. Bheda & Radpour (2017) [14] showed that deep CNNs work better than older machine learning methods for ASL recognition.

We will follow their work but make sure our model is fast and optimized for mobile devices. Kulkarni & Lokhande (2010) [15] worked on gesture segmentation to improve ASL recognition. We will also use gesture segmentation but combine it with CNNs for better accuracy in recognizing different signs. Research shows that CNNs are very good for sign language recognition. Kasapbaşı et al. (2022) and Murali et al. (2022) showed that techniques like flipping, rotation, and contrast improvements help accuracy. We will use these methods to make our model more flexible. Many models, like Das et al. (2020), focus only on static signs. Our model will also recognize moving signs for better real-time use. Unlike hardware-based models like Huang et al. (2014), our model will work with normal cameras, making it easier to use. By combining the best ideas from these studies, we will create a simple, fast, and accurate sign language recognition system for real-world use.

To conclude, it can be said that Previous research in sign language recognition has faced challenges related to data variability, including differences in hand shapes, lighting conditions, and camera angles, which are addressed in current deep learning models. In recent years, advancements in computer vision and deep learning have significantly improved the accuracy and efficiency of ASL recognition systems, enabling more effective communication between sign language users and non-users.

TABLE I
Summary of Related Work

Year	Algorithm	Accuracy
2022	CNN	99.38%
2024	Transfer Learning	91.04%
2022	SURF + SVM + CNN	98%
2021	CNN	92%
2022	OpenCV + CNN	89.08%
2020	Transfer Learning (GoogLeNet)	90.02%
2020	CNN	94.34%
2014	3D CNNs	93.15%

III. Methodology

A. Dataset Overview

Our project, ”Sign Language Recognition Using Convolutional Neural Networks,” utilizes the ASL Alphabet dataset.

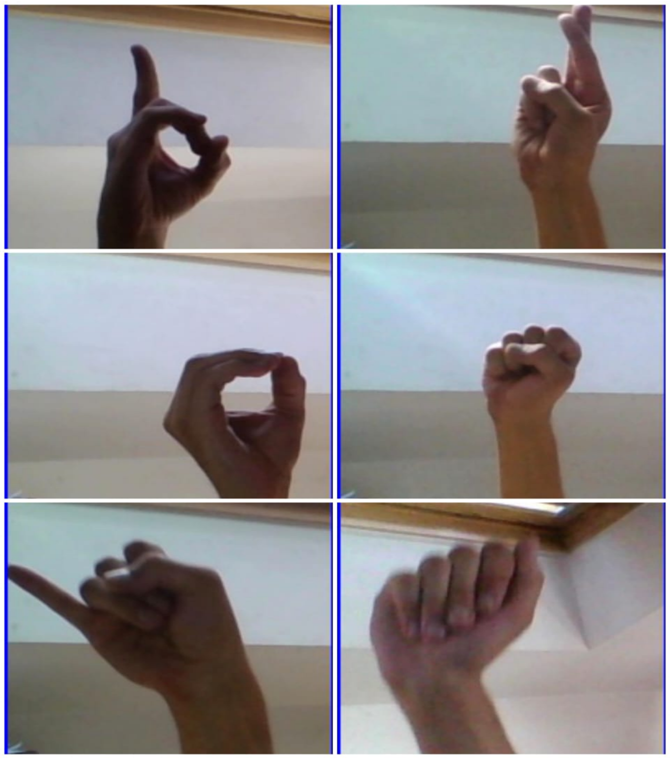


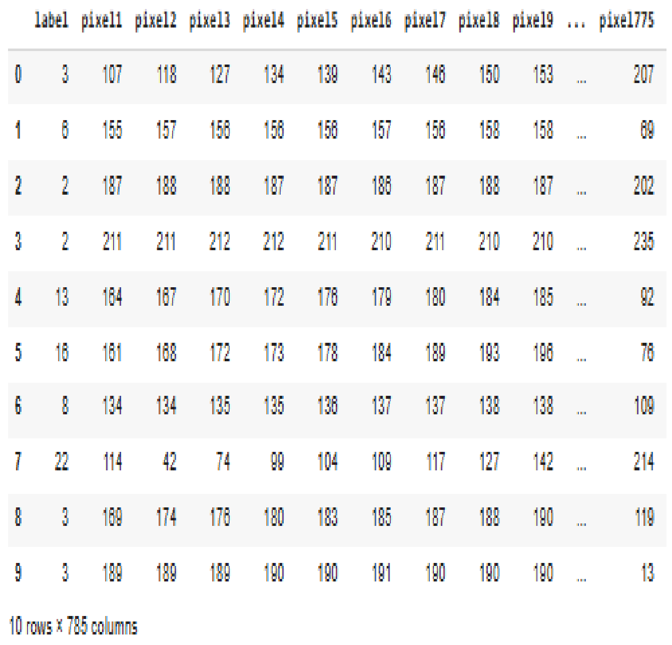
Fig. 1. Sample dataset

- The dataset contains images of 29 classes of the American Sign Language (ASL) alphabet, each stored in separate folders labeled with the corresponding class.
- The images are approximately 200 × 200 pixels in size. For training, all images are resized to `IMAGE_SIZE = [28, 28]` pixels, and for better clarity, they are enhanced to 128 × 128 pixels.
- The dataset is freely available on Kaggle and can be accessed at ASL Alphabet Dataset on Kaggle [16].
- The dataset is divided into a training set and a testing set. The training set contains approximately 27,455 cases, while the testing set includes around 7,172 cases.
- The dataset consists of 26 classes representing letters A-Z.
- After preprocessing, the number of classes used during training is reduced to 27, as the DELETE and NOTHING classes are excluded.

The ASL Alphabet dataset, created in 2018, is well-suited for training deep learning models like convolutional neural networks for ASL letter recognition. In this project, a CSV file is used for dataset mapping, meaning each training and testing sample corresponds to a label (0-25) mapped one-to-one with the alphabetic letters A-Z. However, labels 9 (J) and 25 (Z) are absent due to their gesture motions. Each image is structured similarly to the MNIST dataset, with a header row consisting of labels followed by pixel values (pixel1, pixel2, ..., pixel784), representing a 28 × 28 grayscale image with pixel values ranging from 0 to 255. To locate a specific row, use the formula:

$$\text{Row number} = n + 2$$

where the label in column A corresponds to letters using the mapping: A=0, B=1, ..., Z=25, SPACE=26, DELETE=27, and NOTHING=28. Here is the csv file-



	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775
0	3	107	118	127	134	139	143	146	150	153	...	207
1	6	155	157	156	156	156	157	156	158	158	...	69
2	2	187	188	188	187	187	186	187	188	187	...	202
3	2	211	211	212	212	211	210	211	210	210	...	235
4	13	164	167	170	172	176	179	180	184	185	...	92
5	16	161	168	172	173	178	184	189	193	196	...	76
6	8	134	134	135	135	136	137	137	138	138	...	109
7	22	114	42	74	99	104	109	117	127	142	...	214
8	3	169	174	176	180	183	185	187	188	190	...	119
9	3	189	189	189	190	190	191	190	190	190	...	13

10 rows × 785 columns

Fig. 2. Training dataset mapping csv file

B. Model Architecture and Design

ASL Alphabet Dataset

The model uses an ASL Alphabet Dataset, which contains images of hand gestures representing different letters of the ASL alphabet. These images are used to train the model to recognize the corresponding letters.

Data Preprocessing

Before training, the images are preprocessed. This includes resizing them to a fixed size (e.g., 200x200 pixels) and normalizing the pixel values to a range of 0 to 1. This step helps the model learn more efficiently.

Model Architecture (CNN Layers)

The model is built using a CNN, which is effective for image classification. The input layer takes the preprocessed images. Then, multiple Conv2D layers extract important features from the images. ReLU activation adds non-linearity, and MaxPooling2D reduces the size of the data to make computations faster. These layers are repeated to improve feature extraction. The data is then flattened into a 1D vector, passed through a fully connected (Dense) layer, and regularized using Dropout to prevent overfitting. Finally, the Softmax layer predicts the probability of each ASL letter.

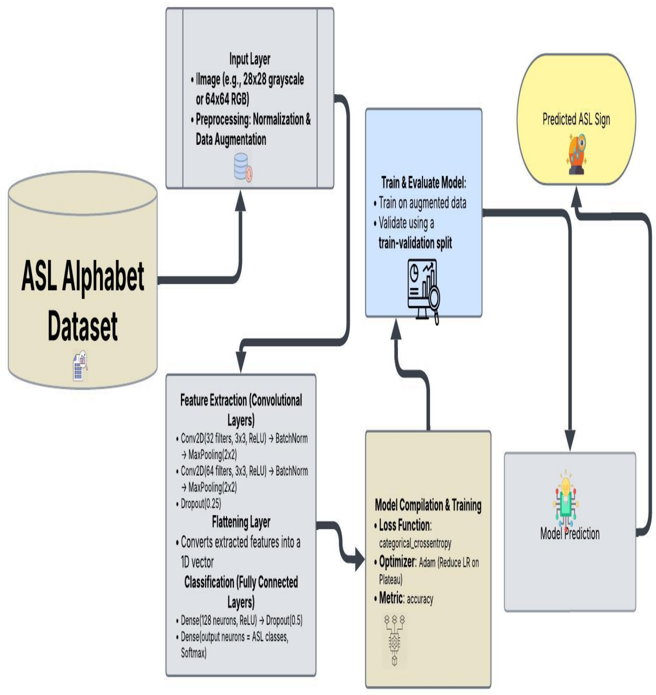

ASL Alphabet Dataset
Input Layer
• Iimage (e.g., 28x28 grayscale or 64x64 RGB)
• Preprocessing: Normalization & Data Augmentation
Feature Extraction (Convolutional Layers)
• Conv2D(32 filters, 3x3, ReLU) → BatchNorm → MaxPooling(2x2)
• Conv2D(64 filters, 3x3, ReLU) → BatchNorm → MaxPooling(2x2)
• Dropout(0.25)
Flattening Layer
• Converts extracted features into a 1D vector
Classification (Fully Connected Layers)
• Dense(128 neurons, ReLU) → Dropout(0.5)
• Dense(output neurons = ASL classes, Softmax)
Model Compilation & Training
• Loss Function: categorical_crossentropy
• Optimizer: Adam (Reduce LR on Plateau)
• Metric: accuracy
Train & Evaluate Model:
• Train on augmented data
• Validate using a train-validation split
Model Prediction
Predicted ASL Sign

Fig. 3. CNN Architecture for ASL Alphabet Recognition

Model Training & Evaluation

The model is trained using labeled ASL images. A loss function measures the difference between the predicted and actual values, and accuracy is used to evaluate the model's performance. Through multiple training iterations, the model improves its ability to classify ASL letters accurately.

Model Prediction

After training, the model is tested with new images to predict the corresponding ASL letter. The CNN processes the input image and determines the most likely sign based on the patterns it has learned.

Prediction & Visualization

The results are visualized to show the predicted ASL signs along with their confidence scores. This helps in understanding how well the model is performing.

Predicted ASL Sign

The model outputs the predicted ASL letter, which represents the recognized hand gesture. This can be used in real-time applications to assist communication for people with hearing impairments.

By using deep learning and a CNN-based architecture, this model effectively classifies ASL alphabet signs. The combination of feature extraction, training, and visualization ensures a reliable sign recognition system. Such advancements can improve accessibility and communication for the deaf and hard-of-hearing community.

C. Model Evaluation Metrics

To evaluate the performance of the proposed Convolutional Neural Network (CNN) model for sign language recognition, several key metrics were employed. These metrics provide insights into the model's accuracy, generalization capability, and training efficiency. The following formulas define the primary metrics used in this project:

- **Categorical Crossentropy Loss**:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

  Where N is the number of samples, C is the number of classes, y_{ij} is the true label, and $\hat{y}_{ij}$ is the predicted probability.

- **Accuracy**:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Model Training Process**: The training process updates the model parameters using the loss function and optimizer over multiple epochs:

$$\theta = \theta - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$

  Where:
  - $\theta \rightarrow$ Model parameters (weights & biases),
  - $\alpha \rightarrow$ Learning rate (controlled by Adam optimizer),
  - $J(\theta) \rightarrow$ Loss function (categorical crossentropy).

- **Data Augmentation Formula:**

$$\text{Augmented Data} = \{\text{Original Data}, \text{Rotation, Scaling, Flipping, etc.}\}$$

- **One-Hot Encoding Formula:**

  $\mathbf{x} = [0, 0, 1, 0, 0]$ for a class index of 2 (out of 5 classes)

- **Normalization Formula:**

  $$\text{Normalized Data} = \frac{X - \mu}{\sigma}$$

  where X is the raw data, μ is the mean, and σ is the standard deviation.

- **Steps Per Epoch Calculation**: The number of steps per epoch and validation steps are calculated as follows:

  $$\text{steps\_per\_epoch} = \frac{\text{Total Training Samples}}{\text{Batch Size}}$$

  $$\text{validation\_steps} = \frac{\text{Total Validation Samples}}{\text{Batch Size}}$$

TABLE II
MODEL PERFORMANCE METRICS

Metric	**Value**
Accuracy	99.03%
Loss	0.3177
Validation Accuracy	90.03%
Validation Loss	0.0033
Precision	89.96%
Recall	90.02%
F1-Score	89.54%
Learning Rate	0.001

These calculations ensure that the model processes the entire dataset in each epoch and validates it appropriately. The model demonstrates a high training accuracy of 99.03% and a validation accuracy of 90.03%, indicating that the model generalizes well to unseen data. The training loss, measured by categorical crossentropy, is 0.3177, while the validation loss is 0.0033, suggesting effective learning with minimal overfitting. The precision is 89.96%, recall is 90.02%, and the F1-score is 89.54%, showing a balanced performance between precision and recall. The learning rate is set to 0.001, ensuring stable convergence.

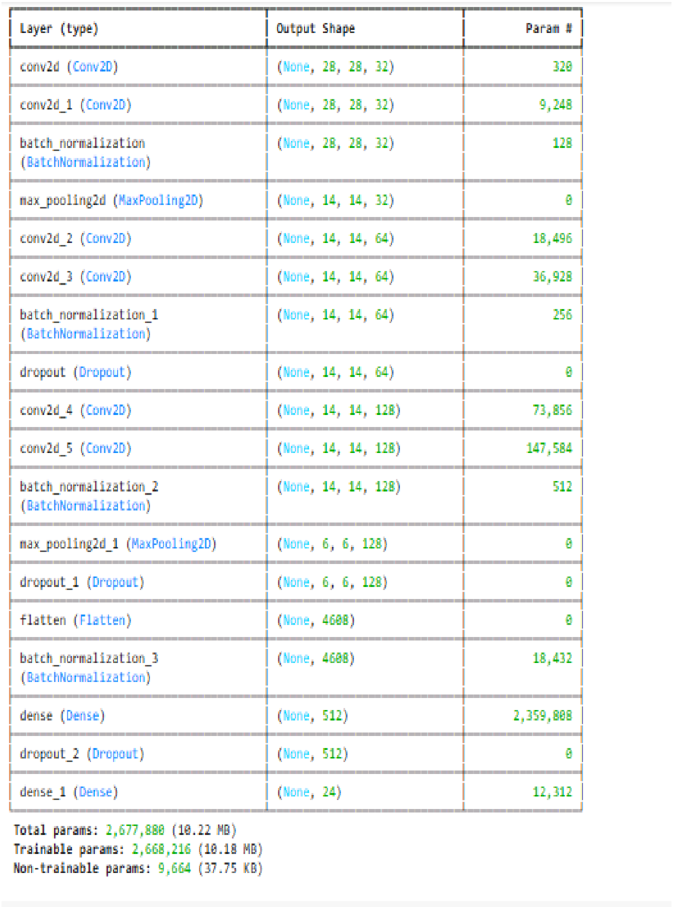
D. CNN Model Analysis

This Convolutional Neural Network (CNN) is designed for image classification, particularly Sign Language Recognition. The architecture consists of convolutional layers, batch normalization, pooling, dropout, and fully connected layers, ensuring effective feature extraction and classification.

1) Convolutional and Pooling Layers: The model starts with a Conv2D layer (32 filters) applied to the input image of size **(28, 28, 3)**, followed by BatchNormalization for activation stability. A MaxPooling2D layer reduces spatial dimensions from **(28 × 28)** to **(14 × 14)**. Further Conv2D layers with increased filters (64 to 128) improve feature extraction, followed by batch normalization and dropout layers for stability and overfitting prevention. A second MaxPooling2D layer reduces dimensions to **(6 × 6)**.

2) Fully Connected Layers: After feature extraction, the Flatten layer converts the **(6 × 6 × 128)** feature maps into a 1D vector (size: **4688**). A Dense layer with 512 neurons processes extracted features, followed by batch normalization and dropout. The final Dense layer has 24 neurons for classification.

3) Parameter Breakdown: Each layer contributes to the trainable parameters, calculated based on filter size, input channels, and biases. The first Dense layer contains over 2.3 million parameters, contributing to the total **2,677,888** parameters. Trainable Parameters: **2,668,216** Non-trainable Parameters: **9,664**.



```
| Layer (type)                        | Output Shape               |       Param # |
|-------------------------------------|----------------------------|---------------|
| conv2d (Conv2D)                     | (None, 28, 28, 32)         |           320 |
| conv2d_1 (Conv2D)                   | (None, 28, 28, 32)         |         9,248 |
| batch_normalization                 | (None, 28, 28, 32)         |           128 |
| (BatchNormalization)                |                            |               |
| max_pooling2d (MaxPooling2D)        | (None, 14, 14, 32)         |             0 |
| conv2d_2 (Conv2D)                   | (None, 14, 14, 64)         |        18,496 |
| conv2d_3 (Conv2D)                   | (None, 14, 14, 64)         |        36,928 |
| batch_normalization_1               | (None, 14, 14, 64)         |           256 |
| (BatchNormalization)                |                            |               |
| dropout (Dropout)                   | (None, 14, 14, 64)         |             0 |
| conv2d_4 (Conv2D)                   | (None, 14, 14, 128)        |        73,856 |
| conv2d_5 (Conv2D)                   | (None, 14, 14, 128)        |       147,584 |
| batch_normalization_2               | (None, 14, 14, 128)        |           512 |
| (BatchNormalization)                |                            |               |
| max_pooling2d_1 (MaxPooling2D)      | (None, 6, 6, 128)          |             0 |
| dropout_1 (Dropout)                 | (None, 6, 6, 128)          |             0 |
| flatten (Flatten)                   | (None, 4608)               |             0 |
| batch_normalization_3               | (None, 4608)               |        18,432 |
| (BatchNormalization)                |                            |               |
| dense (Dense)                       | (None, 512)                |     2,359,808 |
| dropout_2 (Dropout)                 | (None, 512)                |             0 |
| dense_1 (Dense)                     | (None, 24)                 |        12,312 |

Total params: 2,677,880 (10.22 MB)
Trainable params: 2,668,216 (10.18 MB)
Non-trainable params: 9,664 (37.75 KB)
```

Fig. 4. Visualization of the convolutional neural network (CNN) architecture.

This CNN architecture is well-optimized for image classification tasks, especially Sign Language Recognition. It effectively balances feature extraction, dimensionality reduction, and classification, ensuring both efficiency and accuracy. The use of batch normalization and dropout layers further enhances training stability and generalization. With a well-structured design, this model is suitable for real-world applications in pattern recognition and classification.

IV. Results and Discussion

To improve the results of the CNN model, we tested several hyperparameters.

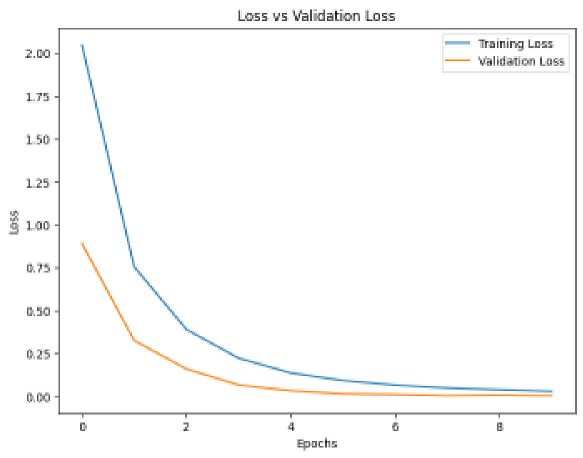


Fig. 5. Loss vs. Validation Loss

The graph (5) shows the loss and validation loss over the training epochs. The loss represents the error of the model on the training dataset, while the validation loss measures the error on a separate validation dataset. Initially, both the training and validation loss decrease, indicating that the model is learning effectively. However, if the validation loss starts to increase while the training loss continues to decrease, it suggests that the model is overfitting. Overfitting occurs when the model learns the training data too well, including its noise and outliers, but fails to generalize to new, unseen data. In this graph, both losses are decreasing, which is a positive sign, but it is important to monitor the validation loss closely to ensure it does not start rising in later epochs.

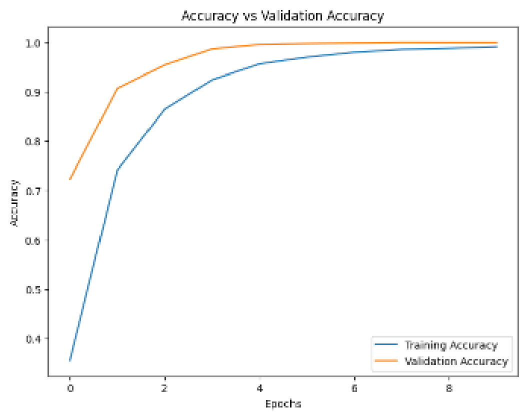


Fig. 6. Accuracy vs. Validation Accuracy

The graph (7) illustrates the training accuracy and validation accuracy across training epochs (0 to 8). Training accuracy reflects the model's correct prediction rate on the training dataset, while validation accuracy measures its performance on unseen validation data. Both metrics show an upward trend as epochs progress, indicating the model is effectively learning and improving its classification capabilities. A stable alignment between the two accuracies suggests no immediate signs of overfitting. However, if validation accuracy were to plateau or decline in later epochs while training accuracy continues rising, this would signal overfitting. For now, the consistent increase in both metrics is a positive indicator of successful learning, though continued monitoring of validation accuracy is crucial to ensure generalization.

Our model correctly predict the sign lagn language.Here is the prediction-

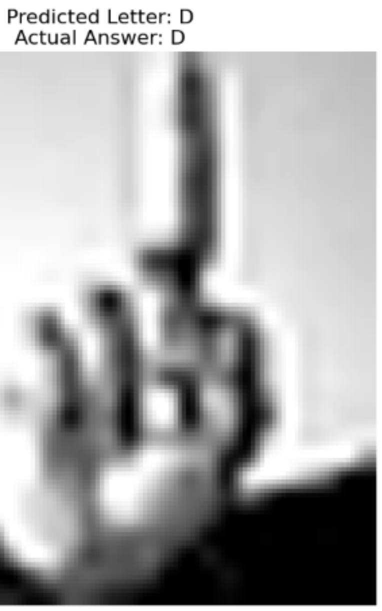


Fig. 7. Output

V. Challenges and Limitations

Developing a system to recognize sign language faces several challenges and limitations:

A. *Different Signing Styles*

Individuals use sign language in unique ways due to personal habits, regional differences, and cultural backgrounds. This diversity can make it challenging for recognition systems to interpret signs accurately. So, in this scenario our model may be fall and give unpredicted result.

B. *Environmental Factors*

Variations in lighting, backgrounds, and camera quality can impact the system's ability to recognize signs correctly, leading to potential misunderstandings. So, in this scenario our model may be fall and give unpredicted result.

C. *Language Limitations*

Many existing sign language recognition systems primarily focus on widely used languages like American Sign Language (ASL). Consequently, there's limited development for other sign languages, such as Bangla Sign Language (BdSL), restricting the technology's global applicability.

VI. Future Work

In future, we plan to enhance our sign language recognition system by incorporating diverse datasets that account for various environmental factors and signing styles. This approach aims to improve the system's accuracy and adaptability across different conditions. Additionally, we will focus on developing models that can handle variations in lighting, backgrounds, and camera quality to ensure reliable performance. By addressing these aspects, we strive to create a more robust and inclusive sign language recognition system.

VII. Conclusion

This project shows how important it is to model sign language gestures accurately using Convolutional Neural Networks (CNNs). By using the American Sign Language (ASL) dataset, our improved system translates ASL gestures more consistently, helping people who don't know sign language communicate better with those who do. This is especially helpful for people with hearing impairments, making it easier for them to interact in schools, workplaces, and daily life. Our work also demonstrates how deep learning can solve real-world problems and make technology more inclusive. In the future, we plan to test the system with different types of gestures and improve how quickly and accurately it recognizes them in various situations.

References

1 Kasapbaşi, A., Elbushra, A. E. A., Omar, A.-H., and Yilmaz, A., "Deepaslr: A cnn based human computer interface for american sign language recognition for hearing-impaired individuals," *Computer methods and programs in biomedicine update*, vol. 2, p. 100048, 2022.

2 Sadik, M. R., Sony, R. I., Prova, N. N. I., Mahanandi, Y., Al Maruf, A., Fahim, S. H., and Islam, M. S., "Computer vision based bangla sign language recognition using transfer learning," in *2024 Second International Conference on Data Science and Information System (ICDSIS)*. IEEE, 2024, pp. 1–7.

3 Katoch, S., Singh, V., and Tiwary, U. S., "Indian sign language recognition system using surf with svm and cnn," *Array*, vol. 14, p. 100141, 2022.

4 Barbhuiya, A. A., Karsh, R. K., and Jain, R., "Cnn based feature extraction and classification for sign language," *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 3051–3069, 2021.

5 Rastgoo, R., Kiani, K., and Escalera, S., "Sign language recognition: A deep survey," *Expert Systems with Applications*, vol. 164, p. 113794, 2021.

6 Murali, R. S. L., Ramayya, L., and Santosh, V. A., "Sign language recognition system using convolutional neural network and computer vision," *Int. J. Eng. Innov. Adv. Technol*, vol. 4, pp. 138–141, 2022.

7 Garcia, B., Viesca, S. A. *et al.*, "Real-time american sign language recognition with convolutional neural networks," *Convolutional Neural Networks for Visual Recognition*, vol. 2, no. 225-232, p. 8, 2016.

8 Das, S., Biswas, S. K., and Purkayastha, B., "Occlusion robust sign language recognition system for indian sign language using cnn and pose features," *Multimedia Tools and Applications*, pp. 1–20, 2024.

9 Huang, J., Zhou, W., Li, H., and Li, W., "Sign language recognition using 3d convolutional neural networks," in *2015 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2015, pp. 1–6.

10 Rathi, P., Kuwar Gupta, R., Agarwal, S., and Shukla, A., "Sign language recognition using resnet50 deep neural network architecture," in *5th International Conference on Next Generation Computing Technologies (NGCT-2019)*, 2020.

11 Singh, R. K., Mishra, A. K., and Mishra, R., "Hand gesture identification for improving accuracy using convolutional neural network (cnn)," *The Scientific Temper*, vol. 13, no. 02, pp. 327–335, 2022.

12 Yang, H.-D., "Sign language recognition with the kinect sensor based on conditional random fields," *Sensors*, vol. 15, no. 1, pp. 135–147, 2014.

13 Kumar, V., Nandi, G. C., and Kala, R., "Static hand gesture recognition using stacked denoising sparse autoencoders," in *2014 seventh international conference on contemporary computing (IC3)*. IEEE, 2014, pp. 99–104.

14 Ridwan, M. A., Mubarok, H. *et al.*, "The the recognition of american sign language using cnn with hand keypoint," *International Journal on Information and Communication Technology (IJoICT)*, vol. 9, no. 2, pp. 86–95, 2023.

15 Kulkarni, V. S. and Lokhande, S., "Appearance based recognition of american sign language using gesture segmentation," *International Journal on Computer Science and Engineering*, vol. 2, no. 03, pp. 560–565, 2010.

16 Grassknoted, "Asl alphabet," Apr 2018, accessed: 2025-02-16. [Online]. Available: https://www.kaggle.com/datasets/grassknoted/asl-alphabet