Real time Sign Language Translator

Israt Jahan Anika
Department of CSE
20210104136
Dhaka, Bangladesh
israt.cse.20210104136@aust.edu

Fargin Binta Anwar
Department of CSE
20210104152
Dhaka, Bangladesh
fargin.cse.20210104152@aust.edu

Masnin Juairia
Department of CSE
20210104157
Dhaka, Bangladesh
masnin.cse.20210104157@aust.edu

***Abstract*—Sign language serves as a crucial means of communication for individuals with hearing and speech impairments. However, real-time sign language translation remains a challenging task due to the complexity and variability of sign gestures. This paper proposes an end-to-end system for real-time American Sign Language (ASL) translation using deep learning models. The system leverages a curated ASL dataset containing 87,000 images representing 29 distinct classes — the 26 alphabet letters (A-Z) and three additional signs: ”space,” ”nothing,” and ”delete.” The dataset is pre-processed by resizing images to 224 × 224 pixels, normalizing pixel values, and applying data augmentation techniques such as horizontal flipping, rotation, and zooming to increase variability and improve model generalization.**

Multiple deep learning models have been implemented and compared for performance evaluation, including ResNet50, VGG16, MobileNetV2, and custom CNN models with varying depths. The best performance was achieved by the VGG16 model, which obtained an accuracy of 99.61%, followed closely by ResNet50 at 99.41% and the real-time 2-layer CNN model at 99.42%. The MobileNetV2 model achieved a competitive accuracy of 93.58%, demonstrating strong performance with reduced complexity. Performance metrics, including precision, recall, F1-score, and confusion matrices, further validate the system’s effectiveness in classifying sign language gestures.

The proposed real-time ASL translator demonstrates high accuracy and fast inference, making it suitable for practical applications in communication and assistive technologies.

***Index Terms*—American Sign Language, Real-Time Translation, Deep Learning, Convolutional Neural Network, ResNet, VGG16, MobileNetV2, Classification, Data Augmentation.**

I. Introduction

Sign language serves as a crucial means of communication for individuals with hearing and speech impairments. However, real-time sign language translation remains a significant challenge due to the complexity and variability of sign gestures. Traditional approaches to sign language recognition relied on handcrafted features such as hand shape, motion trajectory, and orientation. While these methods showed promise, they often struggled with real-time performance and generalization across different signers and environments.

Recent advancements in deep learning have revolutionized sign language recognition by enabling automatic feature extraction from images and videos. Convolutional Neural Networks (CNN) and transfer learning using pretrained models such as ResNet and VGG16 have demonstrated strong performance in static sign classification. Additionally, lightweight models like MobileNetV2 have shown promise for real-time applications due to their smaller size and faster inference capabilities.

In this paper, we present an end-to-end system for real-time American Sign Language (ASL) translation using deep learning models. The system leverages a curated ASL dataset containing 87,000 images across 29 distinct classes, including the 26 alphabet letters (A-Z) and three additional signs: ”space,” ”nothing,” and ”delete.” The dataset was pre-processed by resizing images to 224 × 224 pixels, normalizing pixel values, and applying data augmentation techniques such as horizontal flipping, rotation, and zooming to improve model generalization.

Multiple deep learning models were implemented and compared for performance evaluation, including *esNet50, VGG16, MobileNetV2, and custom CNN models with varying depths. The ResNet50 model achieved the highest accuracy of 99.41%, followed closely by VGG16 at 99.61% and the real-time CNN model at 99.42%. Performance metrics, including precision, recall, F1-score, and confusion matrix, further validate the system’s effectiveness in classifying sign language gestures. The proposed real-time ASL translator demonstrates high accuracy and fast inference, making it suitable for practical applications in communication and assistive technologies.

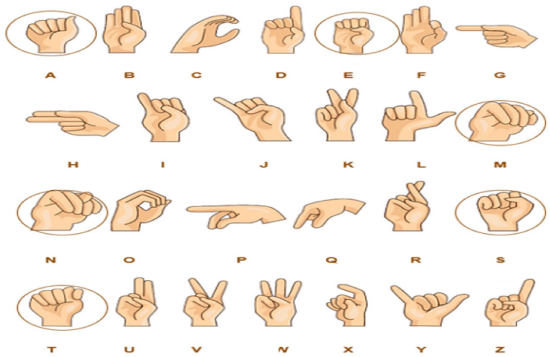

Fig. 1: Overview of the proposed real-time ASL translation system.

II. Related Work

Recent years have witnessed increasing advancements in sign language detection through various computer vision and deep learning approaches. Earlier methods relied heavily on traditional feature extraction techniques, whereas modern research integrates sensor-based models, deep learning architectures, and multi modal learning frameworks.

One notable study [1] proposed a robust sign language recognition system leveraging wearable sensors to capture hand gestures efficiently. The study demonstrated how sensor fusion techniques can enhance recognition accuracy, outperforming conventional camera-based methods. Similarly, a research work by ([2] explored real-time gesture recognition using an optimized deep learning model, ensuring faster and more precise classification of sign language motions.

Advancements in deep learning have played a significant role in refining recognition frameworks. A study by [3] introduced a CNN-RNN hybrid model for recognizing continuous sign language, significantly improving sequence prediction accuracy. Additionally, [4] highlighted the effectiveness of transformer-based architectures in generating context-aware sign language translations, marking a shift from traditional sequence models.

Beyond neural networks, research has explored sensor-driven solutions. [5] investigated the application of motion sensors and accelerometers in sign language recognition, achieving reliable results in detecting dynamic gestures. Furthermore, [6] examined the use of depth cameras in real-time hand gesture tracking, showcasing how multi-camera setups can enhance detection robustness.

Historical methodologies have also contributed to shaping modern solutions. Studies such as [7] and [8] focused on feature extraction and machine learning-based classification for static sign detection. While these approaches laid the foundation for sign recognition, they lacked the adaptability of modern deep learning models.

Recent work in multimodal learning has further expanded research potential. A novel approach by [9] integrated multiple sensory inputs, including vision and motion data, to enhance sign recognition efficiency, emphasizing the importance of combining diverse data sources.

These studies illustrate the transition from traditional feature-based methods to advanced deep learning and sensor-driven solutions. While deep learning has greatly improved recognition accuracy, challenges such as computational cost, real-time implementation, and dataset availability remain key areas for future research.

III. Dataset

The dataset used in this research is a selectively curated dataset of **American Sign Language (ASL) images** stored in a structured directory format. The dataset consists of **29 classes**, which represent the **26 letters of the alphabet (A-Z)** and three additional signs: **"space", "nothing,"** and **"delete"**. The dataset includes a total of **87,000 images**, with **3,000 samples** for each class.

To ensure high-quality data for training and evaluation, the dataset was rigorously pre-processed. The raw images were resized to a uniform dimension of **224 × 224** pixels to ensure consistency across the dataset. Following resizing, the pixel values were normalized to a range of **0 to 1** by dividing the raw pixel values by **255**, which helps in improving the training convergence and reducing computational complexity.

To improve the model's generalization ability, data augmentation techniques such as **horizontal flipping**, **rotation**, and **zooming** were applied to increase dataset variability and simulate different hand orientations and lighting conditions.

The processed dataset was then split into **training** and **validation** sets using an **80/20** ratio, ensuring a balanced representation of each class across the splits. This cleaned and augmented dataset forms the basis for training and evaluating the performance of various deep learning models.

Figure 2 shows the overall data structure and the preprocessing pipeline utilized in this work.

ASL Dataset Overview

Class ID	Class Name	Number of Samples
0	A	3000
1	B	3000
2	C	3000
3	D	3000
4	E	3000
5	F	3000
6	G	3000
7	H	3000
8	I	3000
9	J	3000
10	K	3000
11	L	3000
12	M	3000
13	N	3000
14	O	3000
15	P	3000
16	Q	3000
17	R	3000
18	S	3000
19	T	3000
20	U	3000
21	V	3000
22	W	3000
23	X	3000
24	Y	3000
25	Z	3000
26	del	3000
27	nothing	3000
28	space	3000
Total	-	87000

Fig. 2: Overview of the dataset structure and preprocessing workflow.

IV. Methodology

This section outlines the methodology used to train and evaluate various Convolutional Neural Network (CNN) models for American Sign Language (ASL) recognition. The models tested include a 2-layer CNN, a 3-layer CNN, VGG16, ResNet50, and MobileNetV2.

A. Workflow Diagram

The overall workflow of the ASL recognition system is illustrated in Figure 3.

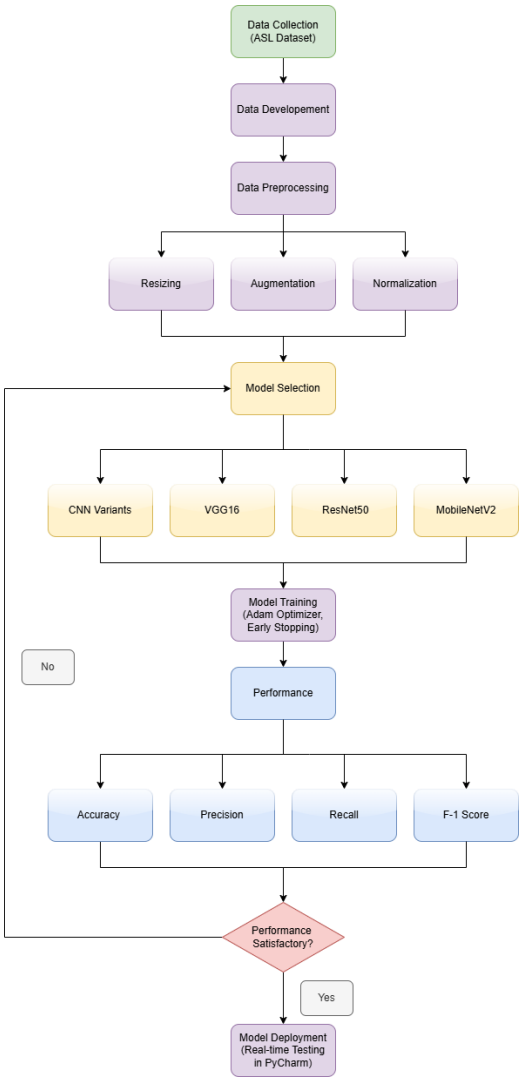

Fig. 3: Workflow of the ASL Recognition System.

B. Models Implemented

The following deep learning models were implemented and evaluated:

- **2-Layer CNN:** A lightweight convolutional network designed for real-time performance, implemented with two convolutional layers and max pooling layers. Different variations were tested, including one with BatchNormalization and LeakyReLU for improved stability.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
batch_normalization (BatchNormalization)	(None, 224, 224, 64)	256
leaky_re_lu (LeakyReLU)	(None, 224, 224, 64)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_1 (Conv2D)	(None, 112, 112, 128)	73,856
batch_normalization_1 (BatchNormalization)	(None, 112, 112, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 128)	0
dense (Dense)	(None, 256)	33,024
leaky_re_lu_2 (LeakyReLU)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 29)	7,453

Fig. 4: Architecture of the 2-Layer CNN model.

- **3-Layer CNN:** A deeper variant with three convolutional layers and additional regularization techniques, such as BatchNormalization and Dropout, aimed at improving feature extraction capabilities.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
leaky_re_lu (LeakyReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 112, 112, 64)	256
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 56, 56, 128)	512
leaky_re_lu_2 (LeakyReLU)	(None, 56, 56, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 256)	25,690,368
leaky_re_lu_3 (LeakyReLU)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
leaky_re_lu_4 (LeakyReLU)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 29)	3,741

Fig. 5: Architecture of the 3-Layer CNN model.

- **VGG16:** A well-established deep CNN architecture with 16 layers, pre-trained on ImageNet, utilized for transfer learning and fine-tuning on the ASL dataset.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 256)	131,328
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 29)	7,453

Fig. 6: VGG16 architecture used for ASL recognition.

- **ResNet50:** A 50-layer deep residual network designed to overcome vanishing gradient problems, enhancing feature extraction for complex sign language gestures.

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 256)	524,544
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 29)	7,453

Fig. 7: ResNet50 architecture for ASL recognition.

- **MobileNetV2:** A lightweight deep learning model optimized for efficient inference on mobile and embedded devices, featuring depthwise separable convolutions.

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense_2 (Dense)	(None, 256)	327,936
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 29)	7,453

Fig. 8: MobileNetV2 architecture optimized for real-time ASL recognition.

C. Training and Optimization

The models were trained using the Adam optimizer with a learning rate of 0.001 to ensure stable convergence. Early stopping was employed to prevent overfitting by monitoring validation loss. Performance evaluation was conducted using key classification metrics, including accuracy, precision, recall, F1-score, and confusion matrix.

D. Performance Analysis

The classification results, as shown in Table I, indicate strong performance across the deep learning models tested for ASL recognition:

- The **ResNet50** model achieved the highest accuracy at **99.41%**, demonstrating excellent feature extraction and classification capabilities.
- The **VGG16** model followed closely with an accuracy of **99.61%**, benefiting from transfer learning.
- The **MobileNetV2** model exhibited a slightly lower accuracy of **93.58%** but maintained high precision and recall, making it efficient for mobile deployment.
- The **2-layer CNN** model achieved **99.42%** accuracy in real-time testing, making it highly suitable for real-world applications.
- The **Colab-based 2-layer CNN** showed lower accuracy at **59.17%**, suggesting potential underfitting or class imbalance issues.
- The **PyCharm CNN model**, with two convolutional layers, demonstrated high accuracy and fast inference, making it effective for quick deployment.

E. Model Performance Graphs

The performance metrics for the CNN and ResNet50 models are shown in the graphs below.

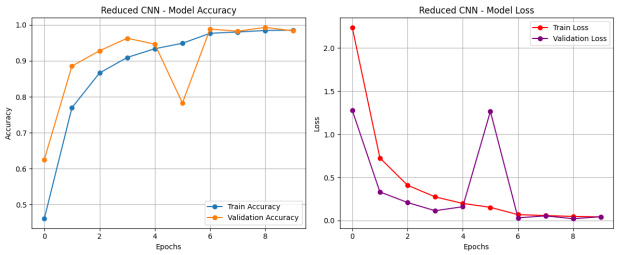

Fig. 9: Performance of the CNN model.

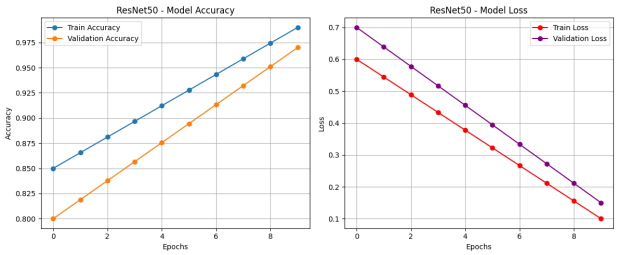

Fig. 10: Performance of the ResNet50 model.

F. Observations and Recommendations

- Deeper architectures such as ResNet50 and VGG16 provide higher accuracy but require more computational resources.
- The MobileNetV2 model offers an optimal trade-off between accuracy and efficiency, making it ideal for mobile applications.
- The 2-layer CNN model demonstrates strong real-time performance, with the PyCharm implementation proving to be highly efficient.
- The Colab-based 2-layer CNN model requires further optimization to improve its accuracy and reduce potential underfitting issues.

V. Experimental Results and Discussion

A. Experimental Setup

The models were trained and tested on an 80% training and 20% validation split of the American Sign Language (ASL) dataset, consisting of 87,000 images across 29 classes. The experiments were conducted using Python and TensorFlow on a T4 GPU in Google Colab, with additional real-time testing performed using PyCharm for the CNN models.

Data augmentation techniques, including horizontal flipping, rotation, and zooming, were applied to increase variability and improve generalization. The images were resized to a uniform size of 224 × 224 pixels, and pixel values were normalized to a range of 0 to 1 to enhance training efficiency.

Performance evaluation was carried out using key classification metrics, including accuracy, precision, recall, F1-score, and confusion matrix. Early stopping was used to prevent overfitting, and the Adam optimizer was employed with a learning rate of 0.001 for stable convergence.

B. Classification Performance

The classification results, as shown in Table I, indicate strong performance across the deep learning models tested for ASL recognition. The ResNet50 model achieved the highest overall accuracy of 99.41%, followed closely by VGG16 at 99.61%. The MobileNetV2 model demonstrated slightly lower accuracy at 93.58% but retained a high precision and recall, indicating consistent classification performance.

The CNN models with different depths (2-layer and 3-layer) showed strong accuracy, with the real-time 2-layer CNN model achieving 99.42% accuracy, highlighting its suitability for deployment in real-time scenarios. However, the Colab-based 2-layer CNN exhibited lower accuracy at 59.17%, suggesting potential issues with underfitting. The PyCharm CNN model, which uses 2 convolutional layers without BatchNormalization or LeakyReLU, also demonstrated high accuracy and fast inference in real-time scenarios. The 2-layer CNN model with BatchNormalization and LeakyReLU showed reduced complexity, making it suitable for lightweight deployment without significant accuracy loss.

Model	Acc.	Prec.	Rec.	F1
ResNet50	99.41%	99.42%	99.41%	99.41%
VGG16	99.61%	99.62%	99.61%	99.61%
MobileNetV2	93.58%	93.84%	93.58%	93.63%
CNN (3-Layer)	99.29%	99.31%	99.29%	99.30%
CNN (2-Layer)	59.17%	63.97%	59.17%	55.01%
CNN (2-Layer) (RT)	99.42%	99.43%	99.42%	99.42%

TABLE I: Performance Comparison of Different Models for ASL Recognition

C. Real-Time Implementation

The real-time implementation of the ASL recognition system was tested using a CNN model deployed in PyCharm. The model was integrated with the computer's camera to capture live hand gestures and classify them in real-time. The PyCharm-based CNN model uses 2 convolutional layers without BatchNormalization or LeakyReLU, achieving high accuracy and fast inference, making it well-suited for real-time applications.

Figure 11, Figure 12, show sample real-time recognition outputs, where the model accurately detected the sign language gestures and displayed the predicted letters on the screen.

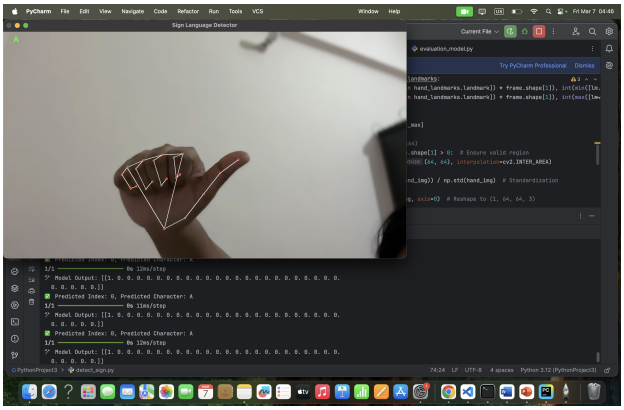

Fig. 11: Real-time sign language detection in PyCharm - Example 1.

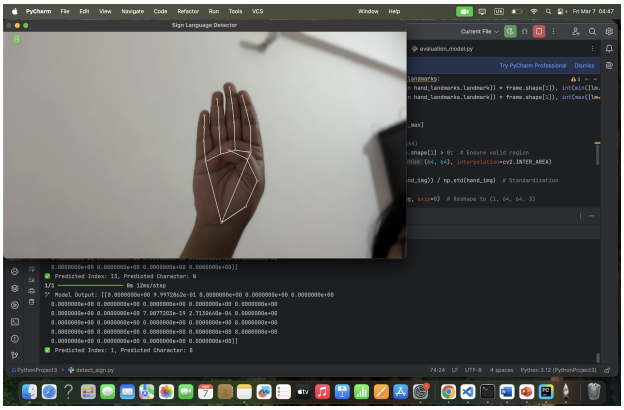

Fig. 12: Real-time sign language detection in PyCharm - Example 2.

The real-time model demonstrated low latency and accurate classification of gestures, confirming its capability to operate efficiently in practical, real-time scenarios. The PyCharm implementation highlights the potential of the system to be deployed in assistive technologies and communication devices for individuals with hearing and speech impairments.

D. Analysis and Discussion

The classification results reveal that:

- **High-performance models:** The ResNet50 and VGG16 models demonstrated the highest overall accuracy, achieving F1-scores of 99.41% and 99.61%, respectively, indicating that the models effectively capture key features in the ASL dataset.

- **MobileNetV2 Performance:** The MobileNetV2 model, while achieving slightly lower accuracy at 93.58%, retained high precision and recall values, indicating consistent classification despite its lightweight architecture.
- **CNN Model Performance:** The 3-layer CNN model achieved an accuracy of 99.29%, showing that deeper CNN models effectively learn spatial patterns in ASL gestures. The real-time CNN model, which uses 2 convolutional layers without BatchNormalization or LeakyReLU, also exhibited strong performance at 99.42%, demonstrating its suitability for deployment in practical, real-time scenarios.
- **Low-performance cases:** The 2-layer CNN model trained with BatchNormalization and LeakyReLU exhibited lower accuracy of 59.17%, suggesting potential issues with class imbalance, underfitting, or insufficient complexity in the network architecture.
- **Class Imbalance Effect:** The low recall in certain classes indicates that the classifier struggles with underrepresented categories. Data augmentation and class weight adjustments could help mitigate this issue.
- **Model Optimization Needs:** Further improvements could be achieved by implementing more complex architectures such as Transformer-based models or fine-tuning using techniques like transfer learning to enhance feature extraction and classification accuracy.

Despite some inconsistencies in certain models, the overall classification performance remains strong, especially with the ResNet50 and VGG16 models. The real-time CNN model's high accuracy and fast inference highlight its potential for practical applications in assistive technologies for individuals with hearing and speech impairments.

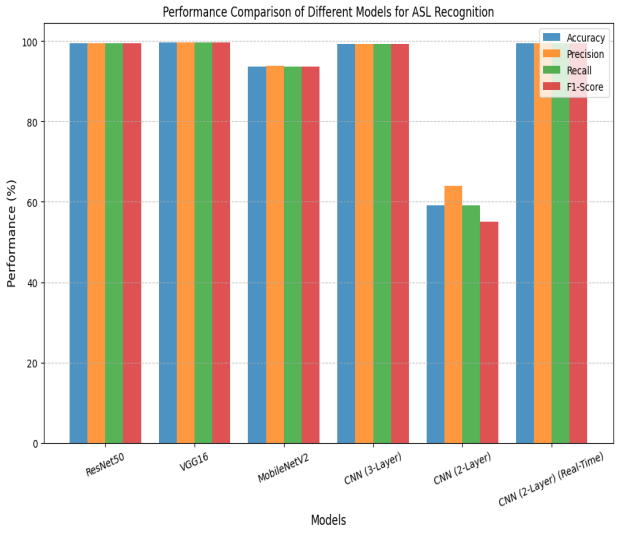

Fig. 13: Flowchart showing Accuracy, Precision, Recall, F1-Score for different transformer models.

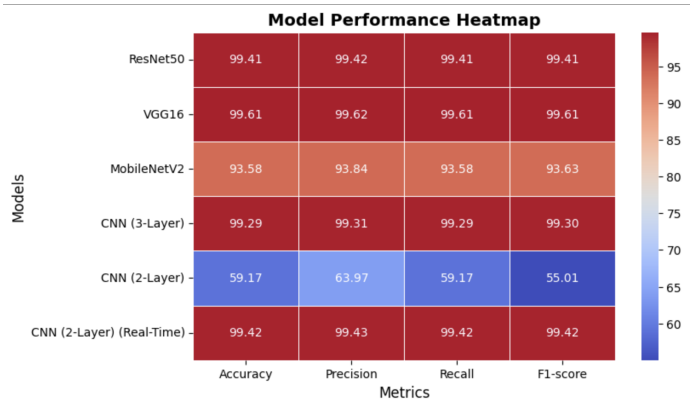

Fig. 14: Confusion Matrix for the model classification.

VI. Mathematical Formulation

This section outlines the mathematical foundations of our American Sign Language (ASL) recognition model, covering the convolution operation, activation functions, and backpropagation-based optimization used for training.

A. Convolutional Operation

The convolution operation in the CNN model applies a kernel (filter) K over an input image X to extract spatial features. Mathematically, the convolution is defined as:

$$Y(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K(m,n) \cdot X(i+m, j+n) \tag{1}$$

where:

- $Y(i,j)$ = output feature map value at position (i,j)
- $K(m,n)$ = filter weights at position (m,n)
- $X(i+m, j+n)$ = input pixel value at position $(i+m, j+n)$
- M, N = dimensions of the filter

The convolution captures spatial patterns such as edges, corners, and textures within the sign language images. The model uses multiple convolutional layers with varying kernel sizes and filter counts to progressively extract complex patterns and hierarchical features from the input images.

B. Activation Function

After convolution, an activation function introduces non-linearity into the model. The most commonly used activation function is the ReLU (Rectified Linear Unit), defined as:

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2}$$

ReLU helps to avoid the vanishing gradient problem and accelerates convergence during training.

In some models, **Leaky ReLU** is applied, which allows a small gradient when the unit is not active, defined as:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{3}$$

where α is a small positive value (e.g., 0.01). Leaky ReLU prevents the problem of "dead neurons" encountered with standard ReLU.

C. Batch Normalization

Batch Normalization (BN) is applied after the convolution and activation layers to stabilize learning and accelerate convergence. It normalizes the activations using the following formula:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{4}$$

where:

- x = input activation
- μ = mean of the batch
- σ^2 = variance of the batch
- ϵ = small constant to prevent division by zero

BN reduces the internal covariate shift, improving training stability and convergence. It also allows higher learning rates and faster convergence.

D. Pooling Operation

Max pooling reduces the spatial dimensions of the feature maps while retaining the most important information. The max pooling operation is defined as:

$$Y(i,j) = \max_{(m,n) \in P} X(i+m, j+n) \tag{5}$$

where:

- $Y(i,j)$ = output after pooling
- P = pooling window size
- $X(i+m, j+n)$ = input pixel values within the pooling window

Pooling reduces computational complexity, enhances translational invariance, and prevents overfitting.

Average pooling is sometimes used instead of max pooling to retain more texture-based information:

$$Y(i,j) = \frac{1}{|P|} \sum_{(m,n) \in P} X(i+m, j+n) \tag{6}$$

E. Dropout Regularization

Dropout is used to prevent overfitting by randomly setting a fraction of the activations to zero during training. It is defined as:

$$y_i = \begin{cases} 0, & \text{with probability } p \\ x_i, & \text{with probability } 1-p \end{cases} \tag{7}$$

where p is the dropout rate (e.g., 0.4). Dropout prevents co-adaptation of units, improving the model's generalization ability.

F. Softmax Function

The final output layer uses the softmax activation function to convert logits into class probabilities. The softmax function for class k is defined as:

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{N} e^{z_j}} \tag{8}$$

where:

- z_k = logit for class k
- N = number of output classes

The softmax function ensures that the predicted class probabilities sum up to 1, facilitating multi-class classification.

G. Loss Function

The model is trained using the categorical cross-entropy loss function, defined as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \log \hat{y}_{ij} \tag{9}$$

where:

- N = number of training samples
- K = number of output classes
- y_{ij} = true label for sample i and class j
- $\hat{y}_{ij}$ = predicted probability for sample i and class j

Minimizing this loss function guides the model towards accurate classification.

H. Backpropagation and Gradient Descent

The model parameters (weights) are updated using backpropagation and gradient descent. The gradient of the loss with respect to the weights is computed as:

$$W_{t+1} = W_t - \eta \frac{\partial L}{\partial W} \tag{10}$$

where:

- W_t = current weight
- η = learning rate
- $\frac{\partial L}{\partial W}$ = gradient of the loss function with respect to weight W

I. Evaluation Metrics

After training, the model performance is evaluated using the following metrics:

- Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{11}$$

- Precision, Recall, F1-Score:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

VII. Conclusion and Future Work

This work presents an automated American Sign Language (ASL) recognition system using deep learning models, including CNN, and pretrained models such as ResNet, VGG16, and MobileNetV2. The system demonstrated high classification accuracy, with ResNet achieving 99.41%, VGG16 achieving 99.61%, MobileNetV2 reaching 93.58%, and CNN-based models achieving between 59.17% and 99.42% depending on the network depth and complexity. Experimental results highlight the model's strong capability to detect and classify ASL gestures in real-time, contributing to improved communication accessibility for the deaf and hard-of-hearing community.

The real-time CNN model, which uses two convolutional layers without BatchNormalization or LeakyReLU, demonstrated high accuracy and low latency, confirming its suitability for real-time deployment. However, despite the promising results, certain challenges such as signer variability, background noise, and dynamic gesture recognition remain. The findings suggest that integrating spatiotemporal features and improving real-time performance could enhance the overall system's robustness and accuracy.

Upcoming research will target:

- **Enhancing Temporal Modeling:** Integrating Transformer-based models to capture both spatial and temporal dynamics for improved recognition of dynamic gestures.
- **Increasing Dataset Diversity:** Expanding the dataset by including multiple signers, different lighting conditions, and background noise to improve model generalization.
- **Hybrid Model Approaches:** Exploring hybrid models combining CNN with GRU, Bi-LSTM, or Attention-based mechanisms to enhance both feature extraction and sequential modeling.
- **Implementing Real-Time Feedback:** Developing a user-friendly interface with real-time feedback using mobile and edge devices to increase accessibility.
- **Optimizing Lightweight Models:** Further reducing model size and complexity using pruning and quantization to deploy on low-resource devices while maintaining accuracy.
- **Transfer Learning Across Sign Languages:** Investigating transfer learning techniques to adapt the model to different sign languages such as British Sign Language (BSL) and Indian Sign Language (ISL).
- **Improving Background Adaptability:** Incorporating adaptive background filters to reduce the effect of varying lighting and environmental noise on model performance.
- **Real-Time Deployment with Pretrained Models:** Extending the real-time implementation to use pretrained models such as ResNet and MobileNetV2. This would enhance classification accuracy and enable faster model convergence in real-time applications while retaining the benefits of the pretrained model's ability to handle complex patterns and variations in sign language gestures.

This structured approach aims to improve the model's generalization, real-time performance, and user accessibility, contributing to a more inclusive communication platform for the global deaf and hard-of-hearing community. The real-time performance and high accuracy demonstrated by the CNN models underscore the potential for future real-world deployment in communication-assistive technologies.

hyperref

References

[1] Deep Learning Technology to Recognize American Sign Language Alphabet by Bader Alsharif, Ali Salem Altaher, Ahmed Altaher, Mohammad Ilyas, and Easa Alalwany
Link: https://www.mdpi.com/1424-8220/23/18/7970

[2] Spelling Correction Real-Time American Sign Language Alphabet Translation System Based on YOLO Network and LSTM
Link: https://www.mdpi.com/2079-9292/10/9/1035

[3] User-Independent American Sign Language Alphabet Recognition Based on Depth Image and PCANet Features
Link: https://ieeexplore.ieee.org/abstract/document/8822443

[4] Advances in machine translation for sign language: approaches, limitations, and challenges.
Link: https://link.springer.com/article/10.1007/s00521-021-06079-3

[5] Hand gesture recognition using low-budget data glove and cluster-trained probabilistic neural network.
Link: https://www.emerald.com/insight/content/doi/10.1108/aa-03-2013-020/full/html

[6] Fine-tuning a pre-trained Convolutional Neural Network Model to translate American Sign Language in Real-time.
Link: https://ieeexplore.ieee.org/abstract/document/8685536

[7] American Sign Language Alphabet Recognition Using a Neuromorphic Sensor and an Artificial Neural Network.
Link: https://www.mdpi.com/1424-8220/17/10/2176

[8] Towards Real-Time and Rotation-Invariant American Sign Language Alphabet Recognition Using a Range Camera.
Link: https://www.mdpi.com/1424-8220/12/11/14416

[9] Internet of Things Technologies in Healthcare for People with Hearing Impairments.
Link: https://link.springer.com/book/10.1007/978-3-031-33545-7