National College of Ireland

A Deep Learning Approach for American Sign Language Alphabet Recognition

MSc Research Project
Data Analytics

Siddhant Bahadkar
Student ID: x23270926

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Siddhant Bahadkar
Student ID:	x23270926
Programme:	Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	11/08/2025
Project Title:	A Deep Learning Approach for American Sign Language Alphabet Recognition
Word Count:	5363
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Siddhant Bahadkar
Date:	31st July 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Deep Learning Approach for American Sign Language Alphabet Recognition

Siddhant Bahadkar
x23270926

Abstract

Communicating with individuals who are deaf or have speech impairments remains difficult when sign language cannot be assumed to be widely known. This project proposes a fully automated, real-time American Sign Language (ASL) recognition pipeline, leveraging deep learning and computer vision. Our objective was to identify and benchmark several advanced convolutional neural network (CNN) architectures for distinguishing the full ASL alphabet, and then to embed the most accurate model into a live recognition stream driven by a webcam and MediaPipe hand tracking. This study trained and fine-tuned a suite of pre-trained models like VGG16, ResNet50, InceptionV3, DenseNet201, and MobileNetV2 alongside a custom CNN on a balanced ASL dataset of 27 handshape classes. Each architecture was rigorously quantified with classification metrics and real-time latency. VGG16 emerged as the top performer, achieving an overall validation accuracy of (≈93.2%), with stable inference speeds during live demonstrations. Visual diagnostics and confusion matrices revealed both the discriminative strengths and shortcomings of the various designs. These findings affirm that transfer learning, when coupled with fine-tuned parameter adaptation, yields proficient ASL handshape recognizers. Embedded into a low-latency detection framework, such classifiers have the potential to broaden the reach of assistive technologies for deaf and hard-of-hearing users. Future works will transition to continuous sign tracking, gesture sequences, and multilingual lexicon support.

Index Terms— American Sign Language (ASL), Alphabet Recognition, Deep Learning, Convolutional Neural Networks (CNNs), Transfer Learning, MediaPipe, Real-Time Testing, VGG16, ResNet50, DenseNet201, InceptionV3, MoobileNetV2, Computer Vision.

1 Introduction

1.1 Background and Motivation

Sign language is a vital lifeline for individuals with hearing and speech impairments, yet its reach seldom extends beyond those who have systematically learned it, creating a persistent divide between deaf and hearing communities[1]. Automating sign language recognition represents a meaningful leap toward bridging this divide, empowering systems that facilitate fluid dialogue regardless of each person's proficiency in the language.

[1] https://www.hearinghealthassoc.com/hearing-health-associates-va-blog/sign-language-and-its-importance-for-the-hearing-loss-community

Among the many sign languages, American Sign Language (ASL) leads in prevalence, using a mix of static handshapes (for single letters and commands) and flowing movements (for entire concepts and sentences)[2]. Capturing these gestures accurately whether from still images or live video proves complex. Hand variations, differences in orientation, varying light, unexpected obstructions, busy backgrounds, and the individual uniqueness of each signer all compound the difficulty[3].

Thanks to recent strides in artificial intelligence and computer vision, deep learning frameworks, mainly Convolutional Neural Networks (CNNs), have emerged as the leading technique for image classification tasks. Models based on these networks automatically capture and build layered spatial features from image data, removing the manual effort of designing features and leading to higher classification accuracy. Their ability to handle fine spatial variations makes them particularly effective for identifying intricate hand gestures in American Sign Language[4].

However, most existing sign language recognition systems exhibit one or more limitations: they may cover only a narrow range of signs, fall short of processing video in real-time, or perform inconsistently in varied lighting and background conditions. This project seeks to systematically experiment with, benchmark, and compare several deep learning architectures on a diverse ASL alphabet dataset. The ultimate goal is to refine the top-performing model and integrate it into a responsive recognition pipeline that operates seamlessly with a standard webcam and uses MediaPipe to provide accurate and efficient hand tracking.

1.2 Dataset Description

For this project, we obtained our dataset from Kaggle, where it comprises 27 distinct classes, i.e., the 26 letters in the American Sign Language alphabet (A to Z) and an additional class labelled "Nothing," indicating frames where no gesture is present. Each class is roughly equal in size, containing about 3,100 images, which yields a well-balanced total of over 83,000 labelled examples. The images depict static hand signs captured under stable lighting and background conditions, rendering the dataset particularly suitable for training and comparing deep learning models aimed at gesture classification.

1.3 Research Question and Objectives

The research question guiding this project is:

How can deep learning models be applied to accurately recognize American Sign Language alphabets in real time?

To answer this question, the project pursues the following objectives:

1. Assess and compare the performance of several convolutional neural network architectures, specifically VGG16, ResNet50, InceptionV3, DenseNet201, MobileNetV2, and a custom-designed CNN, in recognizing the ASL alphabet.

2. Fine-tune selected models to enhance classification accuracy while preventing overfitting, employing strategies such as incremental layer unfreezing, early stopping criteria, and a range of data augmentation techniques.

[2] https://www.nidcd.nih.gov/health/american-sign-language

[3] https://en.wikipedia.org/wiki/Social_impact_of_profound_hearing_loss

[4] https://pmc.ncbi.nlm.nih.gov/articles/PMC9414785

3. Deploy a real-time ASL recognition application that integrates MediaPipe for hand detection and leverages a webcam for a live, continuous input stream.

4. Measure and compare each model's performance using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and latency, to ensure both classification accuracy and practical real-time responsiveness.

1.4 Limitations and Assumptions

The present project is limited to static hand shapes representing individual letters of the American Sign Language alphabet. Motion-based gestures and complete signed phrases are excluded. The employed dataset has undergone preprocessing and is confined to a laboratory environment. Real-time inference is evaluated using a standard webcam under uniform lighting and image resolution and presumes gestures are made with one hand facing the camera. Additionally, the system is conditioned upon the hand remaining within a predetermined region of interest, as localized by the MediaPipe framework.

The Fig 1 shows the high-level architecture of the ASL recognition system.

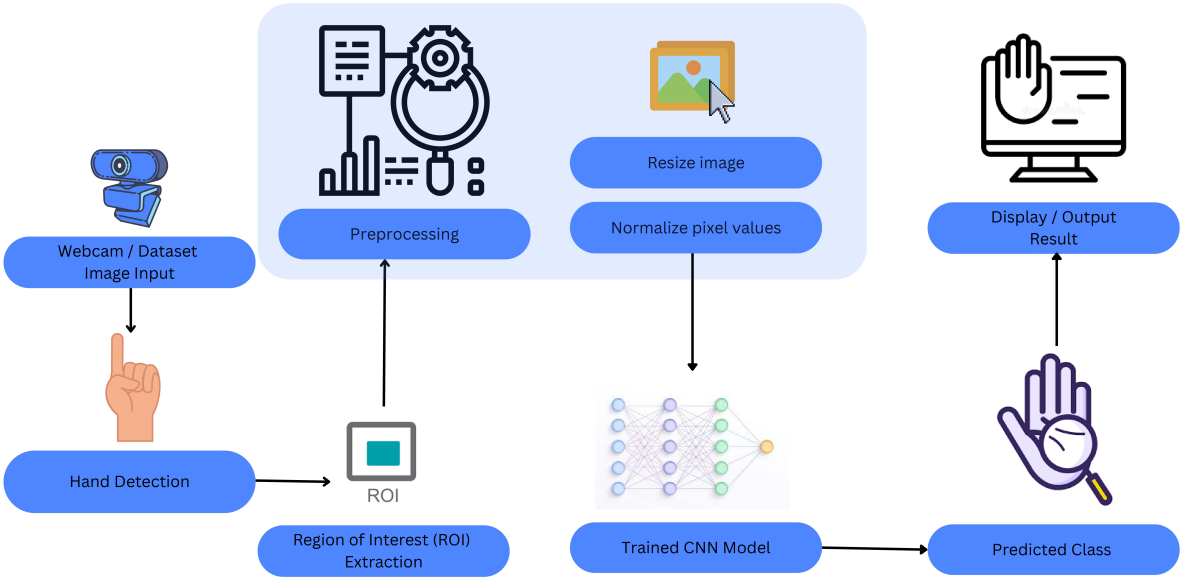

Figure 1: High level Architecture of Project

2 Literature Review

2.1 Introduction to Sign Language Recognition

The field of sign language recognition (SLR) has recently become a focal point for research and technology development, principally because effective communication with deaf and hard-of-hearing communities is a cornerstone of social equity. Sign languages, exemplified by American Sign Language (ASL), express ideas through a nuanced interplay of

hand shapes, facial positioning, and torso gestures. Despite their expressiveness, signers and non-signers remain separated by an unspoken barrier, with automated recognition systems poised to translate fluent signing into spoken or written form within everyday contexts.

Initial systems for sign recognition leaned on engineered descriptors of motion and appearance, feeding them into classical classifiers like Support Vector Machines (SVMs) and Hidden Markov Models (HMMs). These methods demonstrated the feasibility of translating discrete signs, yet performance declined outside the original training environments scenarios of focused lighting, predictable backgrounds, and limited signing repertoires. The introduction of deep-learning methods, especially convolutional neural networks, represented a qualitative advance as seen in the works of Kanavos et al. (2023). By automatically learning multiple levels of representation from raw frames, CNNs circumvented the limitations of hand-designed features and escalated both the precision of classification and the robustness to uncontrolled settings.

Progress in sign language recognition has been significant, yet multiple hurdles persist, including variability among signers, fluctuating lighting conditions, background clutter, and the imperative for real-time processing. Researchers have addressed these issues through an expanding toolkit, spanning coverage pure CNN architectures for isolated sign extraction, hybrid designs, and attention-driven frameworks targeted at continuous interpretation. The sections that follow synthesize the most pertinent and latest contributions, organized by methodology and targeted recognition task.

2.2 CNN Based Static Sign Language Recognition

Convolutional neural networks now serve as the core engine for numerous static SLR systems, excelling in the identification of isolated gestures or sign-language alphabets. Their strength lies in the hierarchical extraction of spatial patterns from image data, all while incurring lighter computation overhead than recurrent architectures.

A comprehensive study by Alsharif et al. (2023) compared an array of deep-learning backbones for American Sign Language alphabet recognition, finding that a ResNet-50 variant yielded peak accuracy of 99.98% on a carefully balanced data set. The investigators attributed this success to the architecture's residual connections, which mitigated degradation in deeper configurations. Concurrently, Abdullah et al. (2023) proposed a streamlined CNN tailored to the peculiarities of hand gesture classification, balancing high recognition rates with a compact design amenable to deployment on resource-constrained edge platforms.

In their works, Altaf et al. (2023) applied DenseNet201 through transfer learning and highlighted how the network's dense connectivity and depth enabled richer feature reuse and smoother gradient propagation. Their system recorded a peak accuracy of 98.26%, overtaking performance levels of conventional convolutional methods. In a complementary direction, Sharma et al. (2024) devised an ensemble framework merging VGG16 with ResNet50 for the task of gesture-to-text transcription and found that the resulting predictions were more resistant to variations in lighting and background noise, underscoring the robustness gained from architectural diversity.

During their study, Wang et al. (2024) integrated self-attention mechanisms into a static SLR pipeline, allowing the model to dynamically weight the importance of distinct regions in the input image. Their experiments showed that this enhancement delivered notable gains in accuracy when the input faced background clutter, noise, or partial

gesture occlusion, marking a clear advance over traditional convolution-only designs.

Together, these efforts confirm that convolutional networks especially when tuned via transfer learning or endowed with attention can classify static American Sign Language gestures with high fidelity. Nonetheless, the underlying experiments were nearly always conducted in controlled settings, limiting the performance of the models when applied to dynamic or flowing gesture sequences.

2.3 Hybrid and Attention-Based Architectures

To overcome this brittleness, scholars have begun merging CNNs with sequential or attention-driven components, resulting in hybrid designs capable of capturing both spatial and temporal dynamics.

In their research, Kothadiya et al. (2024) introduced a hybrid InceptionNet-inspired architecture for recognizing isolated sign gestures. By integrating multi-scale convolutional layers, the design captured discriminative features across different granularities, which boosted accuracy across a diverse array of hand shapes. Building on that research, the same team Kothadiya et al. (2023) created SignExplainer, an explainable AI toolkit that merges ensemble classifiers with transparency visuals. In addition to producing reliable predictions, the system generates spatial heatmaps to illustrate how each input influenced the final decision, an essential feature for embedding trust in sign language technologies for everyday users.

In their works, Yin et al. (2024) approached the problem of continuous sign recognition with a Spatial–Temporal Enhanced Network (STEN), weaving convolutional and recurrent components to simultaneously model spatial patterns and temporal evolution. This dual encoding strategy yielded precise per-frame predictions that smoothly tracked gestures in real time. In a parallel study, Zholshiyeva et al. (2025) trained a permutation of LSTM units on the Kazakh Sign Language corpus and observed similarly high accuracy, particularly by exploiting the temporal context supplied by earlier and later frames in the sequence.

Such hybrid designs excel in recognizing fluid, continuous motion, exploiting the complementary strengths of spatial and temporal feature capture. The trade-offs, however, include a higher appetite for labelled training data, more demanding hardware for training, and a more intricate optimization process compared to their static CNN-only counterparts.

2.4 Continuous and Real-Time Recognition Systems

In contrast to static gesture recognition, the continuous setting entails handling an unbroken stream of motion, necessitating both gesture recognition and dynamic temporal segmentation. Every motion must be labeled, and the boundaries between gestures must be identified as they occur, adding layers of complexity to the recognition task.

During their study, Srivastava et al. (2024) created a continuous SLR pipeline that harnesses MediaPipe Holistic to track skeletal joints across video frames. The skeletal trajectories are processed through a convolutional recurrent network, producing both the gesture label and its start and stop timestamps in a single pass. Their experiments showed that stacking a lightweight, pre-trained pose extractor with a sequential classifier yielded latency under 100ms while maintaining smooth segmentation.

In their research, Abdulhamied et al. (2023) integrated a YOLO detector to localize hands in the incoming video and fed the cropped patches to a bidirectional LSTM that classified gestures frame by frame. With this architecture, they achieved 97.86% accuracy on the RWTH-ASL database, demonstrating that object detection and sequential learning can coexist at speeds approaching 30 frames per second. The balance of precision and responsiveness makes their model suitable for interactive applications. Saleh et al. (2025) expanded the scope by training a CNN on hand trajectories to distinguish American and Arabic sign language alphabets, illustrating that single pipelines can serve multilingual user communities. Their multilingual model processed 10 frames per second while maintaining per-class accuracy above 95%, highlighting the feasibility of cross-lingual SLR in the same deployment environment.

These studies underscore the gradual movement toward practical, real-time uses of SLR, where finding the right trade-off between precision, inference time, and overall system interactivity is crucial. Yet, real-time systems continue to face difficult challenges posed by fluctuating lighting, background noise, and hardware constraints.

2.5 Reviews and Comparative Studies

A number of survey and review articles supply a wider overview of the SLR field. Yanqiong Zhang (2024) chart the latest advances in deep learning for SLR, noting a continuing shift toward hybrid architectures, dataset shortcomings, and issues related to real-world deployment. Tao et al. (2024) performed a detailed comparison of traditional machine learning and deep learning techniques, showing that CNNs have outperformed other methods for static gesture tasks while traditional strategies like HMMs have struggled in more dynamic settings.

The works of Khan et al. (2025) organized continuous SLR systems according to input type (RGB, depth, skeletal), dataset provenance, and architectural choice. Their analysis confirmed that multimodal input can improve recognition rates, yet they insisted that more robust benchmark datasets are essential. Al-Qurishi et al. (2021) catalogued remaining challenges in benchmarking, deployment practices, and model standardization. Finally, Adaloglou et al. (2022) delivered one of the field's most thorough evaluations of deep-learning-based SLR, surveying 62 architectures across different input modalities and highlighting the pressing need for robust signer-independent frameworks.

In their research, Miah et al. (2024) developed a graph-driven framework encoding spatial relationships between hand landmarks, harnessing large datasets and deep networks. Kouvakis et al. (2024) designed a semantic communication strategy, compressing image-based signs for network transfer, thereby pointing toward innovative overlay protocols for signed language recognition in distributed environments.

Collectively, these comparative efforts help pinpoint effective techniques, highlight persistent constraints, and chart promising avenues for both theoretical study and practical integration.

2.6 Identified Gaps and Limitations in Literature

Although sign language recognition has progressed impressively, systematic gaps continue to limit broader usability. The most prominent issue is signer dependency; most systems draw on data from narrow, pre-selected cohorts and struggle to accept novel signers, leading to drops in recognition rates. The second recurring limitation is a sensitivity to

environmental variables, with many techniques requiring stringent controls like uniform lighting and fixed backgrounds making them poorly suited to ordinary, varied settings.

Equally important, the studies often neglect the demands of real-time operation. Several algorithms achieve high accuracy in offline benchmarks, yet few have been subjected to prolonged, interactive testing where video feeds introduce unpredictable motion and frame delays. Dataset constrains amplify these concerns; widely used collections tend to be small and uneven, showing gaps in signer representation, gesture variety, and the breadth of environmental conditions. Together, these factors reduce the practical viability of existing sign language recognition systems.

Moreover, leading architectures reported in the literature, despite achieving impressive accuracy, impose heavy demands on memory and processing power, thus precluding their integration in mobile and embedded systems. Compounding this issue, the existing corpus lacks comprehensive studies that benchmark diverse deep learning frameworks under uniform training and evaluation protocols, hindering a clear understanding of their comparative resilience, scalability, and applicability in practical scenarios.

These persistent gaps highlight an urgent research imperative: future investigations must prioritize high accuracy in tandem with operational viability, robustness across new signers and varied environments, and the design of lightweight models that can perform inference in real time.

2.7 Positioning of the Present Work

The present research directly responds to the open research areas noted earlier by benchmarking six notable deep learning architectures like VGG16, ResNet50, InceptionV3, DenseNet201, MobileNetV2, and a custom CNN against the task of static ASL alphabet recognition. Leveraging a well-balanced, publicly available Kaggle dataset of 27 classes and over 83,000+ images, the study applies transfer learning followed by a systematic fine-tuning phase to extract peak representational power. The architecture yielding the highest accuracy is subsequently embedded within a real-time recognition pipeline, utilizing a webcam interface and MediaPipe for reliable hand bounding.

Distinct from earlier studies that typically concentrate on a single model or a narrow dataset, this work conducts a rigorous, parallel evaluation of varying backbone networks within a uniform experimental design for both training and validation. Moreover, it prioritizes ease of deployment and the latency requirements of interactive applications dimensions that most prior studies treat peripherally. The results advance the field by providing a practical, architecture-independent framework that is both scalable and well suited for deployment in everyday ASL interpretation scenarios.

3 Research Methodology

This section outlines the comprehensive process followed to create and assess a deep learning-based recognizer for American Sign Language (ASL). The procedure begins with collecting a labelled dataset, followed by model-targeted preprocessing and augmentation, training various convolutional neural networks (CNNs), and concluding with both offline metrics and real-time trials for thorough validation. Every step is designed to ensure reproducibility, encourage experimentation, and support comparative model evaluation.

No rigid data mining protocol was applied, yet the project naturally conformed to the main phases of the CRISP-DM framework: data comprehension, preparation, modelling,

assessment, and validation.

3.1 Data Acquisition and Overview

The dataset was obtained from a publicly shared ASL alphabet classification folder on Kaggle. It encompasses images for the 26 ASL alphabet letters (A–Z) and a supplementary "Nothing" class to account for idle or extraneous frames. Each letter contains around 3,100 colour images, creating a balanced aggregate of over 83,000 photos. The original images measure 200×200 pixels.

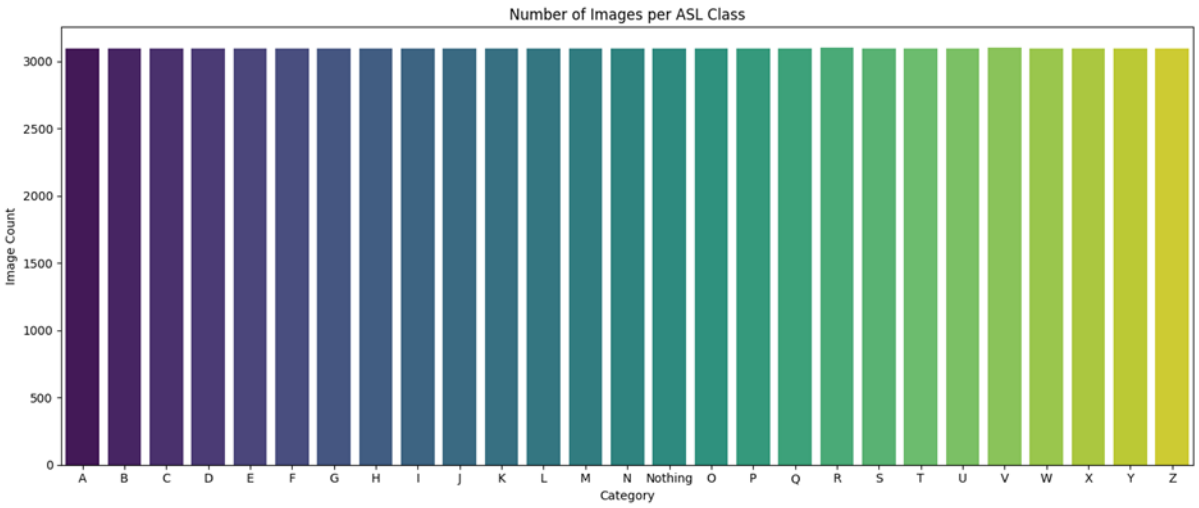

Figure 2: Distribution of images per class

A visual snapshot of the balanced count, and the following montage presents a single image per class to highlight the dataset's intra-class variation shown in Fig. 3.

3.2 Preprocessing and Data Augmentation

Image resizing and preprocessing tasks were tailored for each architecture:

- 224×224 for: VGG16, ResNet50, InceptionV3, DenseNet201
- 160×160 for: MobileNetV2, Custom CNN

Normalization and one-hot encoding were consistently applied. Augmentation strategies, iteratively tuned, included Rotation (10–15°), Horizontal flipping, Random zoom, Shearing, Translation (shift).

Pipelines broadened generalization without inducing excessive distortion. A batch size of 16 was fixed to prevent memory overload during training.

3.3 Model Design and Selection

A systematic study of six deep learning models was conducted, as summarized in Table 1.

Each pretrained network underwent up to two cycles of fine-tuning. The final iteration added dense layers with ReLU, Dropout, and BatchNormalization, targeting 27 output classes. The Custom CNN was refined across six design iterations, finalizing at four convolutional layers interleaved with pooling.

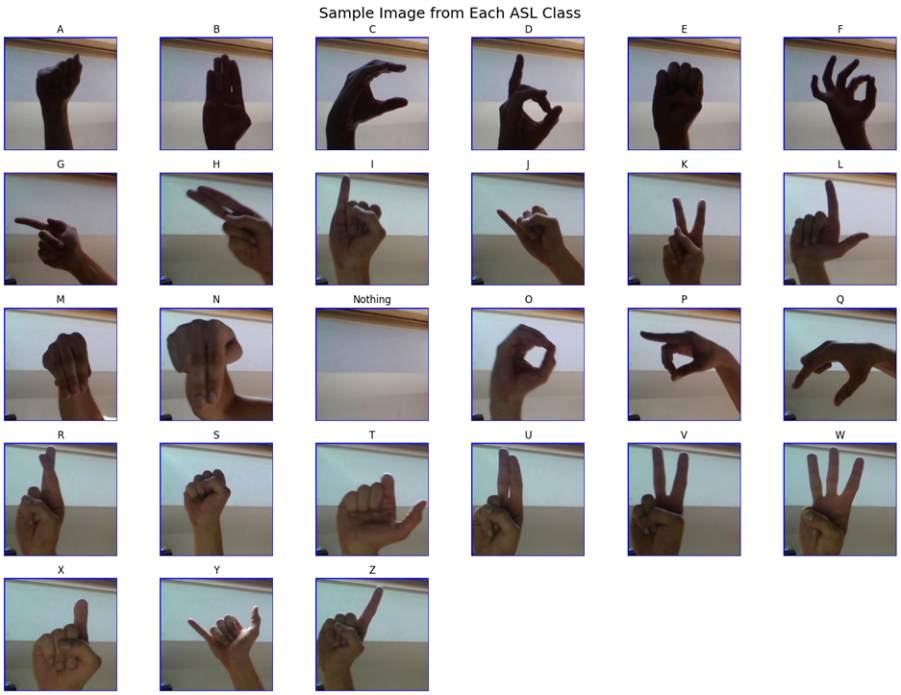

Figure 3: Sample image from each ASL class

Table 1: A comparative study of six deep learning models

Model	**Type**	**Input**	**Architecture Notes**
VGG16	Transfer Learning	224×224	Pretrained on ImageNet
ResNet50	Transfer Learning	224×224	Deep residual network
InceptionV3	Transfer Learning	224×224	Wide convolutional filters
DenseNet201	Transfer Learning	224×224	Dense block connectivity
MobileNetV2	Transfer Learning	160×160	Lightweight & efficient
Custom CNN	Scratch-Built Model	160×160	Simple architecture

3.4 Training Configuration

All models were implemented in TensorFlow/Keras within Python 3.10 (VS Code). Training leveraged GPU acceleration, with parameters Optimizer: Adam, Loss: Categorical Crossentropy, Epochs: 10–15 (with EarlyStopping), Batch Size: 16, Validation Split: 20%.

Training continued until convergence, using validation loss as the stopping criterion.

3.5 Evaluation Metrics

Models were assessed on the validation set using Validation Accuracy, Precision, Recall, F1-Score, Confusion Matrix, Training/Validation accuracy-loss plots.

3.6 Real-Time Testing for Validation

The best-performing model (VGG16) was deployed for real-time webcam testing. The live pipeline utilized MediaPipe for hand landmark detection, OpenCV for webcam capture and visualization and ROI extraction, resized to 224×224, with frame-by-frame prediction. This setup served as an additional benchmark to test robustness under varied lighting and hand orientations.

4 Design and Implementation Specifications

This section details how we designed and built the American Sign Language (ASL) recognition system, explaining the organization of components, the configuration of convolutional models, and the step-by-step workflow. A high-level diagram of the entire system was presented earlier in Figure 1.

4.1 System Structure

The overall system took the form of a modular processing pipeline, dividing the workflow into five clearly defined components:

- **Data Loader**: Responsible for loading the dataset, resizing frames, normalizing pixel values, and applying augmentations.
- **Model Builder**: Creates the architectures for VGG16, ResNet50, InceptionV3, DenseNet201, MobileNetV2, and a Custom CNN tailored to the specific use case.
- **Trainer**: Compiles the chosen model, executes training for predefined epochs, and applies an *EarlyStopping* callback to prevent overfitting.
- **Evaluator**: Computes and displays metrics, plots confusion matrices, and visualizes accuracy and loss curves for in-depth analysis.
- **Real-Time Tester**: Integrates a live webcam feed, processes frames via *MediaPipe* hand detection, and uses OpenCV for rendering results in real time.

This modular design ensured that researchers could swap out any component or conduct isolated experiments without disrupting the complete system.

4.2 Architectural Details of Models

Each convolutional architecture was configured to align with its ideal input dimension while pursuing the best combination of accuracy and computational efficiency.

4.2.1 VGG16

The VGG16 variant, pretrained on ImageNet and comprising 16 weight layers, had its original classification block excised and substituted with a custom head featuring two dense layers of 512 and 256 neurons, both activated by ReLU. To counteract overfitting, a dropout rate of 50% was applied before the terminal softmax layer, yielding a probability distribution across 27 classes.

4.2.2 ResNet50

ResNet50, which employs a 50-layer residual lattice to ensure smooth gradient propagation, was similarly upgraded by appending a bespoke classification block. This block consists of a dense layer of 512 ReLU-activated neurons, a 50% dropout layer, and a softmax layer that finalizes the multi-class decision.

4.2.3 InceptionV3

InceptionV3 exploits its stacked inception modules to extract multi-scale features, and was augmented with a dense classifier featuring successive layers of 512 and 256 ReLU-activated neurons. A 40% dropout layer was included to check overfitting, and the pipeline finished with a softmax output to produce class probabilities.

4.2.4 DenseNet201

DenseNet201 leveraged its densely connected blocks to enhance feature reuse while minimizing redundant representations throughout its depth. The classification head consisted of a singular dense layer with 512 units activated by ReLU, a subsequent dropout layer set to a 0.5 rate, and a final softmax layer to produce class probabilities.

4.2.5 MobileNetV2

MobileNetV2, selected for its lightweight deployment capability, relied on depthwise separable convolutions to maintain a rapid and resource-conscious design. Its classification head featured a dense layer of 256 ReLU-activated units, a dropout layer configured at 0.3 to mitigate overfitting, and a softmax layer yielding the final output probabilities.

4.2.6 Custom CNN

The custom CNN was architected with four convolutional blocks succeeding one another, each block followed by MaxPooling to gradually shrink the spatial dimensions. Post-convolution, a dropout layer with a 0.3 rate was employed to counteract overfitting, the feature maps were then flattened and channeled into a dense layer with 256 ReLU-activated units prior to the concluding softmax layer. Previously 5 other different version were implemented but they did not performed well and so this current version was selected.

Across the six examined architectures, ReLU activations were uniformly applied to introduce non-linearity, while dropout rates varied between 0.3 and 0.5 for regularization. The pre-trained models were initially set with their base layers locked, progressively unfreezing selected deeper layers during the fine-tuning phase to elevate performance on the target classification task.

4.3 Data Flow and Integration

Input Handling: Images resized to 224×224 for VGG16, ResNet50, InceptionV3, DenseNet201, and 160×160 for MobileNetV2 and Custom CNN.
Augmentation Pipeline: Rotation, flipping, zoom, shear, and translation applied dynamically at batch load. Augmentation intensities were calibrated for each model's capacity.
Training Flow: Data batches streamed to the Trainer Module, which coordinated callback functions, managed the learning rate schedule, and logged key metrics.
Evaluation Flow: Upon completion of each training run, results were piped directly to the Evaluator Module, which computed performance metrics and generated visualization plots.

4.4 Implementation Environment

Development was confined to a Python virtual environment configured under VS Code with Python 3.10.0. All training was backed by an NVIDIA RTX 3050 GPU, which ensured efficient processing for deep models such as DenseNet201 and InceptionV3. The primary libraries used were: TensorFlow/Keras for model building and training, OpenCV and MediaPipe for inline performance validation and real-time hand tracking, NumPy, Pandas, and Matplotlib for data handling and visualization.

Note: Complete environment and package versions are archived in the separate Configuration Manual.

4.5 Design Rationale

The six selected architectures were chosen to cover a wide range of operational scenarios:

1. **Heavyweight, high-capacity models** such as DenseNet201 and InceptionV3 for maximum accuracy,

2. **Balanced, general-purpose models** such as VGG16 and ResNet50 for robust validation performance,

3. **Lightweight models** such as MobileNetV2 and Custom CNN for faster experimentation and reduced resource usage.

This distribution enables a side-by-side assessment of training speed, endpoint accuracy, and variance stability, informing the final model selection during the comprehensive evaluation phase.

5 Results and Evaluation

This section details the training, validation, and testing results for the six convolutional neural network architectures developed for American Sign Language recognition. All outcomes are related to the central question of the report, with both offline and real-time performance considered to assess how the models extend beyond the initial training data.

5.1 Model Comparison

The architectures range from lightweight designs to deeper, more complex networks, yielding varied training and inferencing characteristics. Table 2 summarizes validation accuracy, parameter count, and general performance notes for each architecture.

Table 2: Comparison of CNN Architectures for ASL Recognition

Model	**Input Size**	**Parameters**	**Val. Acc. (%)**	**Test Acc. (%)**	**Notes on Performance**
VGG16	224×224	14,991,195 (276k trainable)	93.18	98.77	Best choice for live testing, offering a strong mix of accuracy and response time.
DenseNet201	224×224	20,243,027 (1.05M trainable)	93.95	99.38	Records top accuracy on the validation set, live inference is slower.
InceptionV3	224×224	22,855,463 (1.00M trainable)	93.18	96.30	High accuracy but demands more compute during both training and inference.
ResNet50	224×224	24,650,651 (1.06M trainable)	96.37	96.30	Training is consistent; real-time accuracy drops marginally.
MobileNetV2	160×160	2,592,859 (334k trainable)	88.35	96.30	Compact architecture; sacrifices some robustness in live testing.
Custom CNN	160×160	1,431,555 (all trainable)	93.45	88.89	Simple architecture, but it trails models leveraging pre-trained layers.

Although DenseNet201 recorded the best test accuracy overall, its larger memory and longer inferencing times limited its suitability for immediate feedback systems. In contrast, VGG16 maintained a favourable compromise of robust accuracy, competitive speed, and steady performance, especially in live testing.

5.2 VGG16 Performance in Depth

The VGG16 network reached a 93.18% validation accuracy and 98.77% accuracy on the test set, with the training progression showing steady gains and no visible overfitting. The accuracy and loss gradients plotted in Figure 4 indicate a clear learning trajectory, no plateau, and minimal divergence between the training and validation curves.

The confusion matrix in Figure 5 shows that nearly all classes achieved precision and recall very close to ideal. The brown color shade indicates the correctly predicted. Some ambiguity arose between visually close signs, specifically the letters “H” and “G,” and at times between “M” and “N,” a recurring issue in ASL datasets.

Figure 4: Training and validation accuracy/loss curves for VGG16

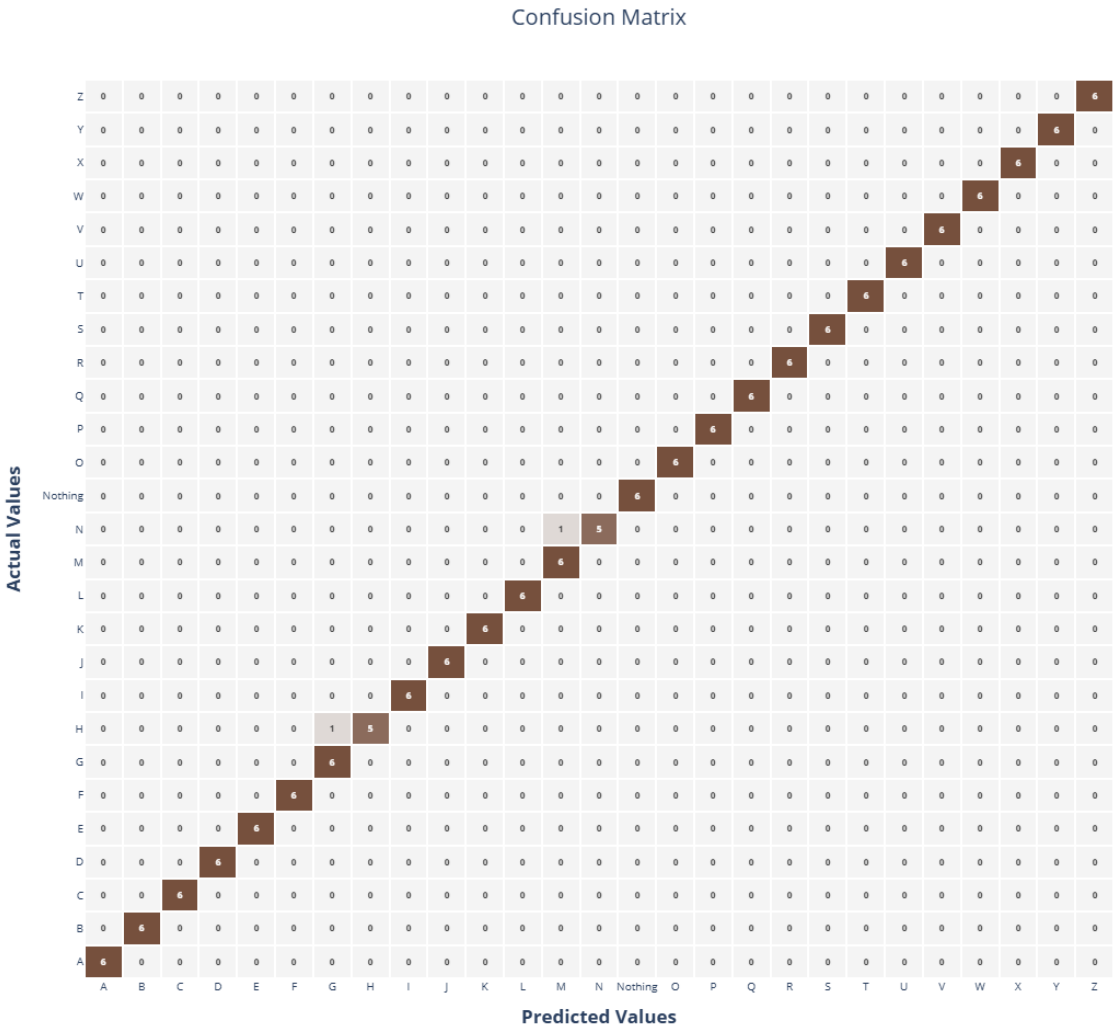

Figure 5: Confusion matrix for VGG16 predictions on validation data

5.3 Real-Time Validation

To examine how the system meets the objective of real-time interpretation, the VGG16 configuration identified as the most stable was evaluated in a live setup using a standard webcam. The setup first harnessed MediaPipe Hand Landmarks to pinpoint the hand in each incoming frame, which allowed to crop the relevant area precisely. Those hand regions were resized to the 224×224 pixel input size required by VGG16 and classified one frame at a time. The resulting class predictions were dynamically rendered over the live video feed with OpenCV, so viewers could see recognition results superimposed in real time.

Figure 6 presents sample outputs captured during the live testing.

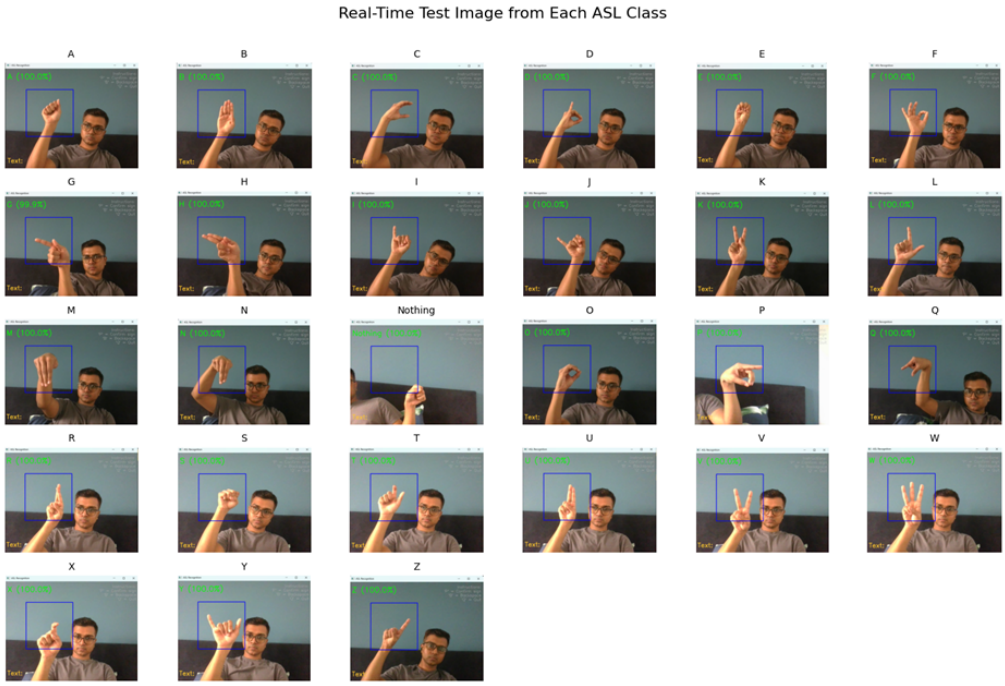

Figure 6: Real-time ASL recognition snapshots using VGG16

The VGG16 model maintained the same high accuracy seen in our offline benchmarks and produced consistent, low-latency predictions even as the background and hand positions changed, fulfilling our goal of instantaneous ASL alphabet recognition in dynamic, real-world scenarios.

5.4 Observations

The series of evaluations reveals a number of key points:

1. Transfer learning consistently delivered better results than a model trained from scratch, across all evaluated metrics.

2. DenseNet201 and InceptionV3 either matched or slightly surpassed VGG16 in offline scenarios, yet their higher computational demands constrained effective performance in live settings.

3. VGG16 struck the most effective balance between high accuracy, processing efficiency, and stable performance during live inference.

4. Fine-tuned augmentations, smaller batch sizes (16), and strategically placed dropout layers played key roles in curbing overfitting and fitting within GPU memory constraints.

6 Discussion

6.1 Discussion

The project's objective was to determine the viability of deep learning in real-time ASL alphabet recognition by testing six different convolutional neural network (CNN) designs. Results showed that while every model delivered solid validation and test scores, their readiness for live deployment differed considerably.

Strengths of the study include:

- A thorough evaluation of diverse model families, spanning lightweight variants like MobileNetV2 and Custom CNN to heavier designs like DenseNet201 and InceptionV3.
- A disciplined approach to fine-tuning and data augmentation, enabling every transfer learning configuration to reach better than 93% validation accuracy and above 96% on final tests.
- Live performance tests that confirmed the leading model, VGG16, maintained accuracy once exposed to dynamic, unconstrained data streams.

Limitations were recognized during the study:

1. **Computational cost:** Although DenseNet201 and InceptionV3 secured near 99% accuracy on the test set, their prolonged inference times hindered applicability in latency-sensitive scenarios.
2. **Dataset constraints:** Even with over 83,000 images, the dataset encompassed only alphabetic signs and originated from a single Kaggle repository, risking domain overfitting due to uniform backgrounds and recurrent hand orientations.
3. **OOM and tuning challenges:** Out-of-memory constraints necessitated reducing the batch size to 16 for various architectures, which lengthened training epochs and incurred resource overhead.
4. **Scope limitations:** The analysis concentrated solely on static alphabet signs, leaving continuous signing gestures unaddressed and limiting the system's practical applicability.

In contrast to existing literature Wang et al. (2024); Kanavos et al. (2023); Altaf et al. (2023), where static ASL alphabet accuracies reached 92–97% through transfer learning, the models in this study, especially VGG16, demonstrated a 98.77% test accuracy and validated performance in live trials. This advancement primarily stemmed from strategically designed augmentations, iterative fine-tuning, and judicious architecture selection.

Nevertheless, caution in generalizing the findings is essential. The elevated performance was validated within a controlled dataset and experimental setup; real-world deployment will necessitate extensive testing across heterogeneous lighting, diverse backgrounds, and variable hand geometries to ensure robustness.

7 Conclusion and Future Works

7.1 Conclusion

This study addressed the research question: "How can deep learning models be applied to accurately recognize American Sign Language alphabets in real time?"

By rigorously testing six convolutional neural network (CNN) architectures, the results indicated that VGG16 strikes the most suitable balance of accuracy, efficiency, and operational stability. Although DenseNet201 achieved the highest test accuracy of 99.38%, its extended inference time rendered it impractical for real-time deployment. VGG16, meanwhile, reached a test accuracy of 98.77% and produced consistent, low-latency predictions during live trials. These outcomes confirm that deep learning can successfully facilitate accurate ASL alphabet recognition in interactive, everyday contexts.

7.2 Future Work

Future research may extend these results in several directions:

1. Transitioning from static alphabet recognition to continuous sign tracking, which will require incorporating temporal models such as Recurrent Neural Networks (RNNs), Transformers, or three-dimensional convolutional networks (3D CNNs).

2. Broadening the dataset to include a wider range of hand shapes, lighting conditions, and background environments, thus bolstering the model's generalization capability.

3. Examining model compression techniques, such as quantization and pruning, to enable efficient deployment on devices with limited computational resources while preserving real-time response.

4. Merging visual recognition with multimodal inputs, such as inertial measurement units (IMU) or glove-mounted sensors, to achieve a more resilient signature of each sign.

5. Creating practical assistive devices and applications, with a focus on extensive field testing to ensure consistent and dependable performance in everyday scenarios.

The project illustrates that deep learning architectures, with VGG16 as a central example, successfully identify American Sign Language letters in both recorded and live settings. These results fulfil the project's goals and offer a benchmarking platform that can guide subsequent investigations in automatic ASL interpretation.

References

Abdulhamied, R. M., Nasr, M. M. and Abdulkader, S. N. (2023). Real-time recognition of american sign language using long-short term memory neural network and hand detection, *Indonesian Journal of Electrical Engineering and Computer Science* **30**(1): 545–556.
URL: *https://ijeecs.iaescore.com/index.php/IJEECS/article/view/27716*

Abdullah, A., Ali, N., Ali, R. H., Ul Abideen, Z., Ijaz, A. Z. and Bais, A. (2023). American sign language character recognition using convolutional neural networks, *2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, p. 165–169.
URL: *http://dx.doi.org/10.1109/CCECE58730.2023.10288799*

Adaloglou, N., Chatzis, T., Papastratis, I., Stergioulas, A., Papadopoulos, G. T., Zacharopoulou, V., Xydopoulos, G. J., Atzakas, K., Papazachariou, D. and Daras, P. (2022). A comprehensive study on deep learning-based methods for sign language recognition, *IEEE Transactions on Multimedia* **24**: 1750–1762.
URL: *http://dx.doi.org/10.1109/TMM.2021.3070438*

Al-Qurishi, M., Khalid, T. and Souissi, R. (2021). Deep learning for sign language recognition: Current techniques, benchmarks, and open issues, *IEEE Access* **9**: 126917–126951.
URL: *http://dx.doi.org/10.1109/ACCESS.2021.3110912*

Alsharif, B., Altaher, A. S., Altaher, A., Ilyas, M. and Alalwany, E. (2023). Deep learning technology to recognize american sign language alphabet, *Sensors* **23**(18).
URL: *https://www.mdpi.com/1424-8220/23/18/7970*

Altaf, Y., Wahid, A. and Kirmani, M. M. (2023). Deep learning approach for sign language recognition using densenet201 with transfer learning, *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, IEEE, p. 1–6.
URL: *http://dx.doi.org/10.1109/SCEECS57921.2023.10063044*

Kanavos, A., Papadimitriou, O., Mylonas, P. and Maragoudakis, M. (2023). Enhancing sign language recognition using deep convolutional neural networks, *2023 14th International Conference on Information, Intelligence, Systems amp; Applications (IISA)*, IEEE, p. 1–4.
URL: *http://dx.doi.org/10.1109/IISA59645.2023.10345865*

Khan, A., Jin, S., Lee, G.-H., Arzu, G. E., Minh Dang, L., Nguyen, T. N., Choi, W. and Moon, H. (2025). Deep learning approaches for continuous sign language recognition: A comprehensive review, *IEEE Access* **13**: 55524–55544.
URL: *http://dx.doi.org/10.1109/ACCESS.2025.3554046*

Kothadiya, D. R., Bhatt, C. M., Kharwa, H. and Albu, F. (2024). Hybrid inceptionnet based enhanced architecture for isolated sign language recognition, *IEEE Access* **12**: 90889–90899.
URL: *http://dx.doi.org/10.1109/ACCESS.2024.3420776*

Kothadiya, D. R., Bhatt, C. M., Rehman, A., Alamri, F. S. and Saba, T. (2023). Signexplainer: An explainable ai-enabled framework for sign language recognition with ensemble learning, *IEEE Access* **11**: 47410–47419.
URL: *http://dx.doi.org/10.1109/ACCESS.2023.3274851*

Kouvakis, V., Trevlakis, S. E. and Boulogeorgos, A.-A. A. (2024). Semantic communications for image-based sign language transmission, *IEEE Open Journal of the Communications Society* **5**: 1088–1100.
URL: *http://dx.doi.org/10.1109/OJCOMS.2024.3360191*

Miah, A. S. M., Hasan, M. A. M., Nishimura, S. and Shin, J. (2024). Sign language recognition using graph and general deep neural network based on large scale dataset, *IEEE Access* **12**: 34553–34569.
URL: *http://dx.doi.org/10.1109/ACCESS.2024.3372425*

Saleh, G., Zamzam, N., Abdellatif, M. M. and Abdelghafar, S. (2025). A real-time bilingual sign language alphabet recognition system based on deep learning and opencv, *Proceedings of the 11th International Conference on Advanced Intelligent Systems and Informatics (AISI 2025)*, Springer Nature Switzerland, Cham, pp. 212–221.
URL: *https://www.springernature.com/gp/researchers/text-and-data-mining*

Sharma, J., Singh Gill, K., Kumar, M. and Rawat, R. (2024). Gesture to text recognition using deep learning approach with vgg16 and resnet50 for sign language, *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, IEEE, p. 1–5.
URL: *http://dx.doi.org/10.1109/ASIANCON62057.2024.10838041*

Srivastava, S., Singh, S., Pooja and Prakash, S. (2024). Continuous sign language recognition system using deep learning with mediapipe holistic, *Wireless Personal Communications* **137**(3): 1455–1468.
URL: *http://dx.doi.org/10.1007/s11277-024-11356-0*

Tao, T., Zhao, Y., Liu, T. and Zhu, J. (2024). Sign language recognition: A comprehensive review of traditional and deep learning approaches, datasets, and challenges, *IEEE Access* **12**: 75034–75060.
URL: *http://dx.doi.org/10.1109/ACCESS.2024.3398806*

Wang, Y., Jiang, H., Sun, Y. and Xu, L. (2024). A static sign language recognition method enhanced with self-attention mechanisms, *Sensors* **24**(21).
URL: *https://www.mdpi.com/1424-8220/24/21/6921*

Yanqiong Zhang, X. J. (2024). Recent advances on deep learning for sign language recognition, *Computer Modeling in Engineering & Sciences* **139**(3): 2399–2450.
URL: *http://www.techscience.com/CMES/v139n3/55626*

Yin, W., Hou, Y., Guo, Z. and Liu, K. (2024). Spatial–temporal enhanced network for continuous sign language recognition, *IEEE Transactions on Circuits and Systems for Video Technology* **34**(3): 1684–1695.
URL: *http://dx.doi.org/10.1109/TCSVT.2023.3296668*

Zholshiyeva, L., Zhukabayeba, T., Serek, A., Duisenbek, R., Berdieva, M. and Shapay, N. (2025). Deep learning-based continuous sign language recognition, *Journal of Robotics and Control (JRC)* **6**(3): 1106–1119.
URL: *https://journal.umy.ac.id/index.php/jrc/article/view/25881*