UNIVERSITY OF HERTFORDSHIRE

DEPARTMENT OF COMPUTER SCIENCE

BSc Honours in Computer Science

6COM2018 - Computer Science Project

Final Report

March 2025

Real-Time Indian Sign Language Gesture Recognition and Translation System

Nikunj Bhavsar

Supervised by: Sarmadullah Khan

Abstract

This paper outlines the creation of a real-time system for identifying Indian Sign Language (ISL) alphabet movements from webcam video. By converting hand signals into on-screen text, the goal is to assist close the communication gap for Deaf and hard-of-hearing people. A ResNet50-based deep learning model was trained using a dataset of roughly 51,000 static ISL alphabet pictures combined and supplemented under transfer learning. With especially good results on most letters and slight misunderstanding between visually identical signs (e.g., "J" and "K"), the final classifier attained about 96% accuracy on unseen data. Combining live video capture (OpenCV) with the trained model, a prototype application was built in Python. For instant text conversion, users keep each sign momentarily in view. Under appropriate lighting and hand location, informal tests revealed great accuracy. Although the present system only supports the ISL alphabet, it shows the viability of AI-driven sign language recognition using conventional computer resources. Possible improvements are including text-to-speech or automated hand tracking, covering words and phrases, and handling dynamic gestures. This approach provides a hopeful basis for future assistive tools that support more inclusive communication.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported this project. Special thanks go to my project supervisor, Sarmadullah Khan, for invaluable guidance, insightful feedback, and encouragement throughout the study. I am also thankful to my classmates for their assistance in setting up the experimental environment and to the volunteers who helped in creating the sign language dataset. Finally, I extend heartfelt appreciation to my family and friends for their continuous support and understanding during the course of this project.

Table of Contents

Glossary

Artificial Intelligence (AI): Computer systems capable of performing tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation.

Convolutional Neural Networks (CNNs): A class of deep neural networks specifically designed for analyzing visual imagery, widely used in image recognition tasks.

Data Augmentation: Techniques used to artificially increase the size and diversity of a dataset by applying transformations like rotations, flips, zooming, or color adjustments to existing images.

Deep Learning: A subset of machine learning involving neural networks with multiple layers, enabling complex pattern recognition and feature extraction directly from raw data.

Dropout: A regularization method in deep learning used to prevent overfitting by randomly setting a fraction of input units to zero during training.

Early Stopping: A technique to halt training when validation performance stops improving, thereby preventing the model from overfitting.

Feature Extraction: Process of transforming raw data into informative, non-redundant features suitable for analysis and modeling.

Fine-Tuning: Adjusting the weights of a pre-trained neural network slightly by training it further on a new, typically smaller, dataset.

Gesture Recognition: The ability of computers to interpret human gestures, typically involving hand movements, as meaningful inputs.

Global Average Pooling: A neural network operation that averages each feature map, reducing dimensionality and computational load, commonly used before the final classification layers.

Hidden Markov Models (HMMs): Statistical models used for representing probability distributions over sequences, previously common in gesture recognition tasks.

Indian Sign Language (ISL): The predominant sign language used in India, comprising both static and dynamic hand gestures.

MediaPipe: An open-source framework by Google used for real-time multimedia processing, often utilized for hand landmark detection.

OpenCV: Open-source computer vision library widely used for real-time image and video processing.

Overfitting: A scenario in machine learning where a model learns the training data too closely, capturing noise rather than generalizable patterns, leading to poor performance on new data.

Real-Time Inference: Process of making predictions from a trained model instantly as new data arrives, typically used for interactive applications.

Residual Networks (ResNet): A CNN architecture featuring skip (residual) connections that help train deeper networks by mitigating issues like vanishing gradients.

Skip Connections: Connections in neural networks that bypass one or more layers, aiding gradient flow and enabling training of deeper networks.

Sparse Categorical Cross-Entropy: A loss function used in multi-class classification tasks to measure the discrepancy between predicted probabilities and true class labels.

Static Gestures: Hand signs involving a single fixed hand position, as opposed to dynamic gestures that involve motion.

Support Vector Machines (SVMs): A supervised machine learning algorithm used primarily for classification tasks, based on finding an optimal hyperplane between different categories.

TensorFlow: A widely-used open-source machine learning framework for developing and training neural network models.

Transfer Learning: Technique of leveraging a pre-trained model (usually trained on a large dataset) to perform a related task, reducing training time and improving performance on smaller datasets.

Validation Accuracy: Measurement of a model's performance on a validation dataset, indicating its ability to generalize to unseen data.

Vanishing Gradients: A problem in training neural networks where gradients become very small, slowing or halting learning, particularly in deeper networks.

1 - Introduction

Sign languages go much beyond simple hand gestures. To express concepts in ways similar to spoken languages, they mix facial emotions, hand shapes, and body position (Ijsrem.com, 2023). Sign languages range across many areas much as spoken languages do: in the United States, people use American Sign Language (ASL), but in India, the main sign language is called Indian Sign Language (ISL), also known as Indo-Pakistani Sign Language (Vashisth et al., 2023).

Though ISL has its own grammar and comprises both one-handed and two-handed signs, it is still not common beyond the Deaf community. This lack of knowledge beyond the Deaf community causes a difficult communication gap. Roughly 18 million people in India are Deaf or hard of hearing; many of them rely on ISL in their everyday lives (Ijsrem.com, 2023). Still, just a tiny percentage of the general population can speak this language. Worldwide, more than 5% of people (over 430 million) suffer from hearing loss (World Health Organisation, 2024). Most of the time, bridging that gap depends on human interpreters. But without someone to translate, Deaf individuals can be left out of conversations, job opportunities, and even classrooms.

1.1 - Project Motivation

The simple guiding concept of this initiative is to use useful technology to close the communication gap between ISL users and the larger world. Imagine a system that, in real time, watches someone's signing via a camera and translates those motions into text or perhaps speech, so Deaf or hard-of-hearing people may engage in discussions without relying just on interpreters (Deora and Bajaj, 2019). Recent developments in deep learning and computer vision allow the automatic recognition of complex hand motions (Zhang and Jiang, 2024). This initiative particularly emphasises ISL not only because India has a sizable Deaf population but also because ISL itself has traditionally gotten less technological focus than languages like ASL (Ijsrem.com, 2023). The research also helps to more global efforts in assistive communication by addressing a vital local issue.

1.2 - Aim and Objectives

This project is to develop a real-time tool that translates ISL motions from a camera stream into text or, when practical, spoken output. The project aims to do this by setting out to:

1. Review current gesture recognition techniques and datasets.
2. Create and train a deep learning model particularly with ResNet50
3. Translate from video frames to text form in real time.
4. Measure the accuracy and responsiveness of the system.
5. Find any constraints and suggest ways to improve them.

Although this first version focusses on the ISL alphabet, it lays the groundwork for a more complete system able to translate ISL words and sentences in real time.

Report Structure: This report is organized into several chapters following the handbook format. Chapter 2 presents a Literature Review outlining pertinent works on gesture detection, current sign language datasets, and deep learning techniques related to the project. The Methodology is covered in Chapter 3, which also covers the real-time system's implementation, training methods, data preprocessing techniques, ResNet50 model design decisions, and dataset compilation. Showing accuracy and loss curves, sample predictions, a confusion matrix analysis, and live testing insights—including any technical limitations such as camera latency—Chapter 4 offers the Results and Evaluation. Reviewing the timeline, the Gantt chart, task organisation, and lessons learnt from the process, Chapter 5 reflects on how the project was planned and managed. Reflecting on both system limits and possible extensions, Chapter 6 ends with the major results and recommendations for next developments.

2 - Literature Review

2.1 - Gesture Recognition Systems:

Early research in gesture recognition explored both hardware-based (Farooq et. al., 2019) and vision-based approaches. One classical approach was to outfit the signer with special gear like data gloves. These gloves were packed with flex sensors and motion trackers that could pick up on hand movements and finger bends (Chin, 2020). The idea was to convert these gestures into digital signals that could be interpreted by a system. A great example of this is a project by bioengineers at UCLA. They created a “smart glove” that could recognize American Sign Language (ASL) alphabet gestures and translate them into speech using a smartphone (Chin, 2020).

Despite being fairly accurate, glove-based systems, come with their own set of problems (Kudrinko et. al., 2020). First, the hardware isn’t exactly subtle or comfortable. Wearing a wired-up glove just to have a conversation isn’t ideal. Plus, these devices usually only track the hands, which leaves out a huge part of sign language like facial expressions, body posture, and other non-manual signals that are integral to many sign languages. Some critics have pointed out that these kinds of systems miss the deeper complexity of sign languages. According to (Erard, 2017), it’s not just about hand positions; it’s about context, flow, and grammar. So while gloves gave researchers a good starting point, they weren’t exactly built for natural, everyday communication (Ahmed et al., 2018).

Due to these factors, modern systems have shifted towards vision-based approaches. Instead of making the signer wear something, these systems just use a camera to watch and interpret gestures. The vision-based gesture recognition system aligns well with how humans perceive sign languages. It’s a more intuitive method and one that can potentially pick up on the full range of movements, expressions, and signals. With the rise of computer vision and machine learning, especially deep learning, we've seen a big leap in what’s possible (AL-Qurishi et. al., 2021). These systems can now analyze video or images and identify gestures with growing accuracy (Vashisth et al., 2023).

But it's not without challenges. For instance, different signers have varying hand shapes, skin tones, and signing speeds; gestures can involve one or both hands (ISL, in particular, employs two-handed signs for certain letters/words); and the background environment may clutter the image. Furthermore, some signs are static (defined by a single hand configuration and position) while others are dynamic (characterized by a sequence of motions). For instance, in ISL, the alphabet contains static hand postures for most letters but a few letters may require motion (Singh, 2022). The ISL sign for "J," for example, is made by tracing a J-shape in the air. Recognizing that correctly means understanding motion, not just a single image.

Even with all these hurdles, the field has made solid progress. Early vision-based systems leaned on image processing techniques like segmenting out hand regions using skin-color detection (often in HSV color space), extracting shape or trajectory features, and using classical machine learning algorithms like Hidden Markov Models (HMMs) or Support Vector Machines (SVMs) for classification. Such approaches often required carefully hand-engineered features, and while they worked okay in controlled settings, they often struggled to handle larger vocabularies or more diverse real-world conditions.

2.2 - Sign Language Datasets

A crucial foundation for any sign language recognition system is the availability of a labeled dataset of sign examples. Datasets enable the training and evaluation of machine learning models. In the case of ASL, several datasets have been made available. These datasets contain everything from thousands of static hand sign images (like fingerspelling alphabets) to large-scale video datasets capturing continuous signing (e.g., the ASL Lexicon Video Dataset or datasets from sign language recognition competitions). These resources have propelled ASL recognition research forward in a big way.

But when it comes to Indian Sign Language (ISL), the story is different. Historically, there's been a serious lack of large, publicly available ISL datasets. Most researchers working on ISL had to create their own datasets from scratch, often focusing on a small number of signs, maybe just alphabets, numerals, or a limited vocabulary of common words (Saini et al., 2023)
That said, things have started to improve in recent years. Authors of (ayeshatasnim-h, 2020), for instance, compiled a dataset of ISL alphabet images, about 12,700 in total, by combining self-collected images with contributions from volunteers. This dataset covered all 26 alphabet signs (A to Z), and after data augmentation, each class ended up with around 486 images. The

dataset was then split into training and test sets for model development. The creators mentioned that no comprehensive ISL dataset was available when they began, which is why they built one themselves.

Another major step forward came from (Singh, 2022), who published a much larger ISL static gesture dataset. This one includes over 100,000 images spanning English alphabets, Hindi vowels, and numerals. What's great is that it was collected from a diverse group of participants kids, teens, and adults and captured under a range of lighting conditions and backgrounds. That kind of variation is crucial for building models that generalize well.

The availability of such a comprehensive dataset is very recent and represents a positive development for ISL research; however, handling such a large volume of data turned out to be challenging for a student project with limited computing resources.

2.3 - Challenges with ISL Datasets

One of the biggest hurdles in ISL research is the inconsistency between datasets. Since most researchers build their own, it's tough to compare results directly across different studies. Each dataset might vary in background, lighting, signer hand shape, or even how a sign is performed, which makes generalization tricky (Saini et al., 2023). And small datasets can lead to models that perform well in training but fail in real-world use, especially if they overfit to specific patterns like a particular signer's hand shape or a static background (Raheja et. al., 2016). Moreover, sign language datasets have to consider class balance. Ensuring each sign is well-represented. If some gestures have fewer training samples, the model may perform poorly on them. There is also the question of dynamic vs static signs: datasets of static images (like alphabets held as a pose) are easier to collect (one can take photos of posed signs) and annotate, whereas datasets for dynamic signs or continuous signing require video and often detailed annotation at the frame level or sequence level. For this project, decision was made to focus on static, isolated gestures like the alphabet. That simplifies things into an image classification task instead of a sequence recognition problem. But even with static gestures, there's still a lot of variation. Different people might perform the same sign in different ways with different hand orientations, distances from the body, or even finger spacing. Capturing that variety in the dataset is critical for building a reliable model.

Table 1. Pre-existing datasets on Indian Sign Language recognition
(Reproduced from (Saini et al., 2023) under CC BY 4.0 license.)

Sr. No	Name	Provider	Year	Reason	Reference
1	Indian Sign Language (ISL)	Arikeri P	2020	Dark background, lack of variations, duplicate images	Arikeri (2021)
2	Indian Sign Language (ISLRTC referred)	Dumbre A	2022	Lack of variety, duplicate images	Dumbre (2022)
3	ISL-CSLTR: Indian Sign Language dataset for continuous Sign Language Translation and Recognition	Elakkiya R, Natarajan B	2021	Videos and sentence-based, which was not the goal of this work	Natarajan and Elakkiya (2021)
4	Indian Sign Language Dataset	Sonawane V	2020	Duplicate images	Sonawane (2020)
5	Indian Sign Language Dataset recognized by ISRTC	Kumar K	2022	Bad quality, no variety	Kumar (2022)

All in all, the literature makes it clear: building or accessing a diverse, high-quality dataset is one of the toughest parts of working with sign language recognition — especially for under-resourced languages like ISL. In our project, we tackled this by applying data augmentation techniques and using transfer learning (Gupta et. al., 2022) to get the most out of a limited dataset.

2.4 - Deep Learning Approaches for Sign Language Recognition

The emergence of deep learning has revolutionized pattern recognition tasks, including sign language gesture recognition (Zhang and Jiang, 2024). Among the most powerful tools in this space are Convolutional Neural Networks (CNNs), which have shown outstanding performance in image classification tasks. They've been widely used to recognize static hand gestures, like fingerspelling letters or numbers, and when trained on a decent dataset, they can achieve impressively high accuracy (Vashisth et al., 2023). The advantage of CNNs is their ability to automatically learn discriminative visual features (edges, shapes, etc.) from raw pixel data

through the process of convolution and pooling, eliminating the need for manual feature extraction. For instance, Nandi et al. (2022) implemented a CNN for ISL alphabet recognition and reported high accuracy, demonstrating that a well-designed CNN can distinguish between the 26 hand signs effectively when trained on a sufficient dataset.

As gesture datasets grow and recognition tasks get more complex, deeper CNN architectures have become the go-to (Gupta et. al., 2024). One standout model is ResNet-50, a 50-layer residual network introduced by He et al. (2015) back in 2015. ResNet's innovation was in adding skip (or residual) connections, a clever trick that helps the model train deeper networks without running into problems like vanishing gradients. This design was so effective that ResNet-50 actually won the ImageNet challenge that year (Wikipedia contributors, 2025).

In the context of sign recognition, researchers have experimented with using pre-trained networks like ResNet50, InceptionV3, VGG16/VGG19, etc., often via transfer learning (Hore et. al., 2015). A comparative study by Saini et al. (2023) evaluated several popular CNN architectures (VGG-16, VGG-19, ResNet-50, DenseNet-201, InceptionV3, MobileNetV2, Xception) on a custom ISL dataset (Saini et al., 2023). Their findings indicated that ResNet-50 outperformed the other models for ISL gesture classification, achieving an accuracy of about 98.25% (with an F1-score of 99.34%) on their test set (Saini et al., 2023). They noted that although ResNet50 was slower to train and infer (due to its depth), its accuracy was exceptional, reportedly achieving over 99% on their test data. This aligns with other reports in literature where ResNet-based models have reached high 90s accuracy for static sign recognition tasks (Saini et al., 2023). The success of ResNet50 is attributed to its ability to learn very rich feature representations of the hand gestures, aided by the skip connections that mitigate vanishing gradient issues in deep nets (Mukti and Biswas, 2019).

Now, while ResNet50 is great for static gesture recognition, things get trickier when you move into continuous signing — where a series of gestures form a full sentence. In those cases, you need models that can understand sequences over time. That's where Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, come in (Baihan et al., 2024). For example, a system might use a CNN (e.g., ResNet) to extract frame-wise features and then feed those into an LSTM to recognize a sequence of gestures comprising a word or sentence (Baihan et al., 2024). More recently, researchers have started experimenting with attention mechanisms and even Transformer models (which originated in Natural Language Processing). These models are great at handling sequences of varying lengths and can pay

attention to different parts of the gesture sequence, making them promising for sign language translation (Zhang and Jiang, 2024). That said, they're also data-hungry. And unfortunately, large, annotated ISL video datasets are still pretty rare.

For the scope of this project, we're focused on recognizing individual ISL gestures in real time, not full sentence sequences. So, a deep CNN like ResNet50 fits perfectly (since starting from ImageNet pre-trained weights can significantly boost performance when data is limited). It's already proven in the literature, especially for static image recognition, and offers the kind of accuracy we need (Saini et al., 2023).

There are also a few best practices from existing research that we've followed, including: Image augmentation to artificially expand the dataset with rotated, flipped, or color-shifted images (Vashisth et al., 2023) and careful fine-tuning (Hossain et. al., 2022) of the model to avoid overfitting. Some studies also explore augmentation beyond simple flips/rotations, such as varying background or lighting via synthetic means, to make the model more robust to real-world conditions. Regularization techniques like dropout are also common to help the model generalize better (Vashisth et al., 2023).

Since our project initially focuses on the alphabet, which in ISL has mostly one-handed signs (with a few exceptions that might use both hands), the CNN can be trained to recognize the single hand shape in focus. For two-handed signs, approaches sometimes involve detecting each hand separately or using multi-channel inputs (one for each hand) into the model. While our project does not dive that deeply into multi-hand modeling, the need to handle two-handed gestures is on the radar for future work.

2.5 - Summary

To summarize, the literature review highlights the following key points relevant to our project:

Why This Problem Matters:
There's a clear and pressing need to bridge communication gaps for Deaf communities. Sign language recognition systems have gained attention because they offer the potential to make a real social impact. They make communication more inclusive and accessible for millions (Zhang and Jiang, 2024).

What Has Been Tried Before:

Both hardware (glove-based) and vision-based methods have been explored over the years. While glove-based setups can be accurate, they're not exactly user-friendly. On the other hand, Vision-based methods are more natural and user-friendly. That said, they do come with their own challenges.

The ISL Datasets:

A big challenge with Indian Sign Language has been the lack of large, public datasets. While the situation is slowly improving thanks to recent contributions, many researchers still have to create their own datasets.

Deep Learning and ResNet50:

Deep learning, particularly CNN like ResNet50, has become the go-to solution for static gesture recognition. These models consistently achieve 95%+ accuracy when paired with transfer learning (Saini et al., 2023). The fact that CNNs can automatically learn visual features (without us having to manually define them) makes them incredibly powerful compared to older methods.

Common Challenges Noted:

The literature points out recurring challenges like varying lighting conditions, cluttered backgrounds, and differences in how individuals perform signs. Some systems try to clean up the input first using techniques like background subtraction or color thresholding to isolate the hand. Others rely on CNN's ability to cut through the noise. A few recent approaches use tools like MediaPipe or keypoint detection to extract hand landmarks before classification. It is a slightly different method than ours, but one worth exploring in the future.

Overall, the research helped guide key decisions in our project. For example, the choice of ResNet50 for our model was influenced by its strong performance reported in the task (Saini et al., 2023). Given the constraints of time and computing, we scoped the project to isolated letters rather than full sentence recognition.

3 - Methodology

3.1 - Dataset Collection

Given the scarcity of readily available ISL datasets, we assembled a dataset appropriate for training the gesture recognition model. The focus was specifically on the ISL alphabet 26 static hand gestures representing the letters A to Z. These are commonly used as a base in sign language recognition systems. They involve distinct shapes that are relatively easy to collect and can be used to spell out any word.

The dataset for this project was obtained through a combination of existing image sources and self-collection:

Existing ISL Alphabet Images: To create the dataset for this project, an existing open-source ISL alphabet dataset was used. The dataset is available at https://www.kaggle.com/datasets/vaishnaviasonawane/indian-sign-language-dataset. It contains 1200 images for each letter class (A - Z) which made the total initial database to be of 31,200 images. It also contained numbers (0 - 9), but since the project focused on the alphabets, those classes were dropped. Moreover, the data was labeled by alphabet letter. Each image contains hands showing the letter sign against a plain black background.

Self-Collected Samples: In addition to existing data, we collected our own images for certain gestures to introduce more variation. Using a Macbook Pro's webcam, multiple pictures were taken of myself, performing each alphabet sign (except "Q") **(Code in Appendix A)**. It was ensured to avoid over-reliance on any one environmental setting. For example, aspects such as lighting and hand orientations were varied. Approximately 800 images per letter class were added through this process.

In the end, the dataset contained of roughly 51,000 images, each corresponding to a letter from A to Z. It is worth noting that many signs in ISL differ from those in American Sign Language (like the signs for "T" or "H"). But we weren't comparing the languages linguistically. Thus, each hand gesture was treated simply as a unique class for the model to recognize.

The collected dataset was split into training and validation sets:

Training set: 80% of the images (40663 images) were used for model training. This split was stratified so that each letter had a proportional representation in train/val/test, maintaining class balance across splits.

Validation set: 20% of the images (10165 images) were set aside for validation during training. The validation set was used to monitor the model's performance on unseen data and tune hyperparameters (such as learning rate or to decide when to stop training to avoid overfitting).
While the dataset works well for our current project, it is still limited in scope:

1. While it can spell out words, it doesn't include full word signs, which are crucial for real-world conversations.
2. The backgrounds are either black (in case of the existing dataset) or minimal (added dataset). This helps the model train cleanly but may limit its performance in real-world environments that are more complex.
3. All signs are static, so gestures involving movement (like the letter "J") aren't represented.

Despite all this, the dataset is sufficient to train a model to recognize the 26 alphabet gestures with high accuracy. serving as a proof-of-concept for ISL gesture recognition. It works well as a proof of concept, and future versions can easily build on this base by adding new sign types. Future expansions of the dataset could include numerals, common ISL words, and dynamic gestures.

Figure1. Curated Dataset.

3.2 - Data Preprocessing

Raw images from the two sources had varying resolutions and backgrounds. Preprocessing steps were applied to standardize the input data for ResNet50 and to highlight the hand shapes:
Resolution and Cropping: All images were resized to a uniform resolution suitable for the input. Although ResNet50 is typically designed for 224×224×3 input images (Saini et al., 2023), it was decided to resize all images to 180×180 pixels. The images had three color channels using TensorFlow's image_dataset_from_directory(). This resizing is applied uniformly to all images, regardless of their original dimensions (ranging from 128×128 to 1920×1080). This helped us to retain important visual details and maintain computational efficiency during training.

Color Space Normalization: Things were kept simple when it came to color; all images were used in their original RGB format, just as the ResNet50 model expects. While some studies suggest converting to HSV (Vashisth et al., 2023), we tested both and found no major accuracy difference for our setup. Thus we stuck with RGB for simplicity. ResNet's built-in function, preprocess_input(), was used for preprocessing the data. Doing this ensured that the training and evaluation were fully aligned with the model's expectations. It gave us clean, consistent feature extraction right from the start.

Background Simplification: No fancy background removal algorithm (like chroma key or segmentation masks) was applied while building the model because the backgrounds of the images were already plain and clutter-free.

3.3 - Data Augmentation

After basic preprocessing, data augmentation was performed on the training set. Augmentation is, basically, creating slightly tweaked versions of existing images to make the model more robust. The following augmentation was used:
Geometric Transformations: Random training images had undergone random rotations (roughly 36 degrees). This was done to handle the edge cases where the signer's hands were not perfectly aligned with the camera. In other images, Horizontal flips were done to help the model recognize and predict the sign correctly, regardless of which hand is used. Furthermore, to mimic different camera distances and hand sizes, the images were crammed or zoomed out.
These transformations were made so that the model focuses on the gesture shape and not the position/orientation of the hand.

Apart from that, to keep the project simple and efficient, advanced augmentation setups were not used. Using shifts, cropping, color jitter, and blur would make the dataset and the model more robust. Also, these augmentations were ONLY applied to the training dataset. This ensured that the model was evaluated against the original images.

3.4 - Model Architecture: ResNet50 for Gesture Recognition

For the core task of recognizing ISL hand gestures, ResNet50 emerged as the best option (Saini et al., 2023). ResNet50 is a deep CNN with 50 layers that introduced residual learning blocks. In simpler terms, ResNet introduced a trick known as skip connections. These add the input of a layer to the output of a deeper layer, helping the network avoid problems like vanishing gradients and lets it go deep without getting stuck (Wikipedia contributors, 2025).

3.4.1 - ResNet50's Architecture

It starts with a few convolutional layers and pooling. Then comes 4 stages of convolutional residual blocks (each made up of several smaller residual layers). Finally, it wraps up with global average pooling and a fully connected (dense) layer that spits out a classification. The ResNet50's architecture is shown in **Figure 1**, along with the schematic of the Residual Learning Block (building block of the ResNet).

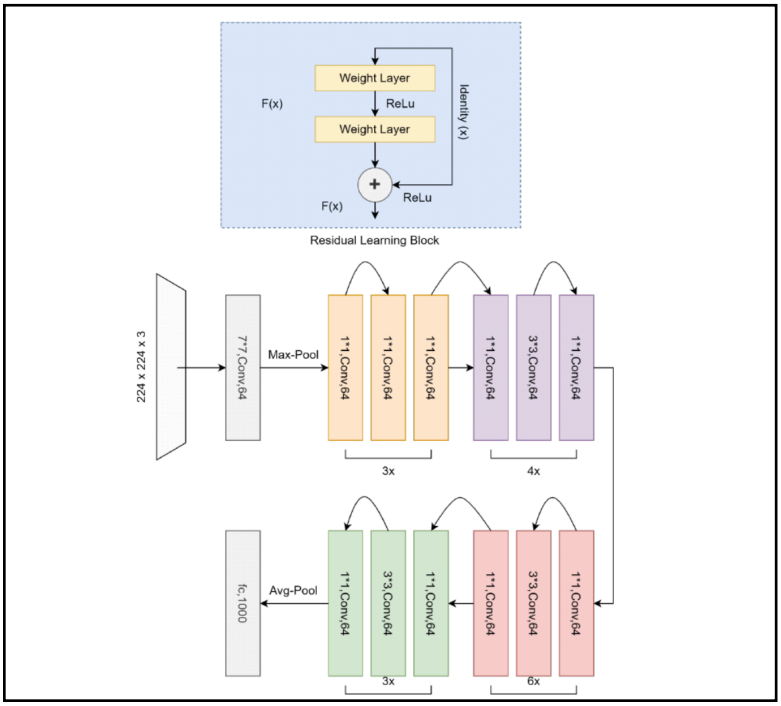

Figure 2. ResNet50 model Architecture
(Reproduced from (Saini et al., 2023) under CC BY 4.0 license.)

3.4.2 - Customizing ResNet50

Originally, ResNet50's output layer was designed to classify 1,000 different categories from the ImageNet dataset (like dogs, airplanes, and buildings). But the project doesn't need that. Therefore, the original output layer was removed, and a new custom head was added. It had softmax activation to produce a probability distribution across all classes. This new head will now guess the output only from the 26 letters. After that, a GlobalAveragePooling2D layer was added. This layer helps to reduce the amount of data while keeping the most important features learned by the model. This was followed by a Dense layer with 256 neurons and ReLU activation. This allows the model to learn complex patterns. An L2 regularization was added to prevent the model from overfitting. For context, overfitting is something that happens when a model memorizes the training data too closely. It performs well in the training stage but poorly in the validation stage.

A Dropout layer with a dropout rate of 0.5 was added to further reduce the chance of overfitting. This randomly turns off 50% of the neurons in that layer during each training step. All images were resized to 180×180 RGB resolution, which was consistent with the input shape expected by our custom model.

3.4.3 - Training and Validation Procedure

Rather than training a model from scratch, a pre-trained ResNet50 was opted as the foundation for our Indian Sign Language (ISL) alphabet classifier. This approach is known as transfer learning. It leverages the fact that the early layers of ResNet50 already recognize essential visual features such as edges, shapes, and textures. These features are also relevant for identifying hand gestures. So, instead of reinventing the wheel, the focus was on teaching the model the specific patterns associated with the 26 ISL letters.

Adopting the classic feature extraction strategy for training was used for training. It involved freezing the pre-trained ResNet50 base (trainable = False) and training only the new layers that were added. Even with this simple setup, the model performed strongly. Although fine-tuning was considered, it was not implemented in this version due to time and resource constraints.
Adam optimizer is known for its efficiency and the ability to automatically adjust the learning rate during training. Therefore, it was decided to use Adam for optimization. A learning rate of 0.001 (1e-3) was selected to ensure stable and effective learning. Furthermore, categorical cross-entropy, which is standard for multi-class classification tasks, was employed. This function

measures the difference between the predicted probabilities and the actual class labels. It guides the model to improve its prediction over time. Early stopping was implemented to prevent overfitting and conserve computational resources. During training, the validation loss was monitored, and if it did not improve for three consecutive epochs, the training process was halted.

In summary, the final model consisted of a ResNet50 backbone with a custom classification head trained on 180×180 RGB input images, using a batch size of 32. Only the top layers were trained, while the pre-trained ResNet50 base remained frozen. This approach resulted in a robust and efficient model that generalized well to unseen data, delivering high performance without the need for excessive training time or data.
Training Results (observed during this process): After fine-tuning, the model reached a training accuracy of ~99% and a validation accuracy of around 96% by the final epoch. The training loss decreased to a very low value (on the order of 0.01), and validation loss also became low and stable. These figures indicated that the model learned the classes well without significant overfitting (the validation accuracy being close to training accuracy is a good sign). One reason overfitting was kept in check is the extensive augmentation. We also computed precision and recall per class on the validation set to ensure that each letter was being learned. Most letters had precision and recall in the 90-100% range, with a few slightly lower due to confusion (for example, on validation data, the model sometimes confused J vs k). These insights helped to understand the limitation of the dataset's variation in terms of lighting, background, etc.

3.5 - Real-Time Inference Implementation

Once the trained model achieved high accuracy on static images, the next step was to deploy it in a real-time system that can translate gestures through a webcam. The real-time translation pipeline was built to capture live video, process each frame using the model, and display the predicted output.
To capture the webcam feed, a predict_webcam.py was made which used OpenCV. Each frame captured from the webcam is initially in OpenCV's default BRG format. It is converted to RGB, and then resized to 180×180 pixels. Next, The frame is passed through ResNet50's standard preprocess_input() function, ensuring consistency with the training pipeline.The model outputs a probability distribution over the 26 alphabet classes (A–Z), and the letter with the highest confidence is selected as the prediction. The letter that was predicted is displayed on the screen along with the confidence.

No additional hand tracking, gesture segmentation, or bounding box cropping was implemented. The entire frame is resized and used directly for prediction. Users are expected to hold their signing hand clearly in the camera view, ideally centered, for optimal recognition. This real-time prediction system demonstrates how the trained model can be deployed for live gesture recognition using minimal infrastructure. It provides immediate visual feedback, making it a functional prototype for interactive ISL translation.

In summary, the real-time system was a straightforward application wrapping the trained model. It demonstrated the viability of using the model outside of an offline environment. The design prioritized clarity and displaying results. It's understood that advanced feature like automatic hand tracking were beyond the scope of this initial implementation.

3.6 - Custom Techniques and Considerations

Some of the optional design considerations that were experimented with initially, were not used in the final version. Still several of the ideas helped shape the eventual direction of the project.
One of these was the consideration of class weighting. This is normally used when a dataset is imbalanced, allowing the model to favor the minority classes when training. In this case, the dataset was already fairly balanced, with all 26 alphabet classes (A–Z) being of roughly the same size. Since no class was overly imbalanced, no class weighting was needed.

Second, the model was intentionally simplified. A ResNet50 model with ImageNet-trained weights was used as the backbone. A custom classification head was then placed on top of it, predicting the 26 ISL alphabet signs. No ensemble models and no architectural improvements were utilized, as the intention was to make the system run real-time efficiently without complicating the design. With regards to training strategy, fine-tuning the lower layers was considered, but it was not performed for the final deployment. Because of the small size of the dataset, the base model being unfrozen was likely cause overfitting. The less complex strategy also saved training time with little compromise on accuracy.

Real-time prediction stability was also of special importance. A recommendation was smoothing out jittery outputs by averaging predictions over a series of frames. In practice, the predictions were already fairly consistent, though, when users held each of the signs for a short period. With the robust results and the aim of keeping latency low, frame averaging was excluded from the final implementation. A simple yet functional user interface was developed using OpenCV.

The detected sign and the level of confidence were indicated on the real-time video stream, with instant graphical feedback. More sophisticated capabilities like a scrolling history of detected signs, side bars for word construction, or generation of speech would have been achievable but were outside the scope of the project.

Typically, the strategy had a linear and practical order: preprocess and prepare the data, train a pretrained model, data augmentation for making the data stronger, train and validate the model, and deploy it on a real-time system. All the decisions were based on findings from similar studies and validated for confirmation. The following section holds the results of this approach, including model accuracy, examples of predictions, and real-life analysis of the system's performance when run live.

3.7 - Adaptation of Initial Plan

The original concept was for a web-based user interface and text-to-speech (TTS) capability as well as a real-time Indian Sign Language (ISL) recognition system built utilising MediaPipe for landmark extraction technique for gesture classification. However, the project scope and technical approach were changed during development. This was done in order to better fit the available resources, data constraints, and schedule. The final project emphasised an image-based gesture detection strategy employing ResNet50 model trained on ISL alphabet images rather than hand landmarks technigue. Several elements motivated this shift: the challenge of acquiring large, labelled landmark-based ISL datasets, the burden of applying precise landmark-to-gesture mapping, and the high performance and accessibility of pretrained image classification models for static signs. The suggested online application and complete TTS integration were also swapped out for a more concentrated desktop-based prototype using OpenCV for real-time webcam input and on-screen text feedback. While simplifying deployment and lowering development complexity, this let the project keep real-time functionality and robust user engagement.

Although the last system varies from the original design in some technical details, it still fulfils the fundamental goals: enabling real-time ISL alphabet recognition, delivering accurate user feedback, and showing the possibilities of deep learning and computer vision in assistive communication tools.

4 - Results and Evaluation

This chapter presents the outcome of the sign language classification project in terms of the performance of the ResNet50-based hand recognition model and the real-time prediction system with the webcam. The outcome is quantified in terms of the project requirements and the process used and discusses the performance of the system, its limitations and its strengths.

4.1 - Model Accuracy and Loss

Following the training process, the model performed well on the evaluation dataset used to assess its performance. The final test accuracy was about 96.4%, meaning that the model correctly identified almost 96 images from 100 gestures on average. This attests to the fact that the ResNet50 feature-extraction-based model was trained to distinguish correctly between the 26 static signs in the ISL alphabet. This accuracy greatly surpasses chance accuracy of ~3.8% on a 26-class classification problem and shows that the model has acquired strong and characteristic visual features regarding hand gestures. The accuracy was consistent with accuracy in the validation accuracy in the course of training and reached a plateau at about 99%, with minimal indication of overfitting and good generalizability.

Throughout the training process, the learning curve of the model was healthy; the loss in training began at approximately 2.5 and steadily dropped to less than 0.1. Second, validation loss was similarly trending downward and leveled at 0.2. Training accuracy increased sharply to well over 92% in the initial epochs and kept improving steadily. The final accuracy in training was [insert final train accuracy, e.g., 99%], with the accuracy in validation leveling at 98%.
The gap between training and validation accuracy was small, suggesting the model did not significantly overfit, thanks to techniques like data augmentation, dropout, L2 regularization, and early stopping. The final model was selected from the epoch where validation performance was optimal.

4.2 - Confusion Matrix Analysis

To gain deeper insight into which gestures the model confuses, we analyze the confusion matrix of the model's predictions on the validation set.The confusion matrix is a 26×26 grid that visualizes how often each true class is predicted correctly or misclassified. Each cell [i,j]

indicates how many instances of class j were classified as class i. The confusion matrix for this model is shown in **Figure 2**.

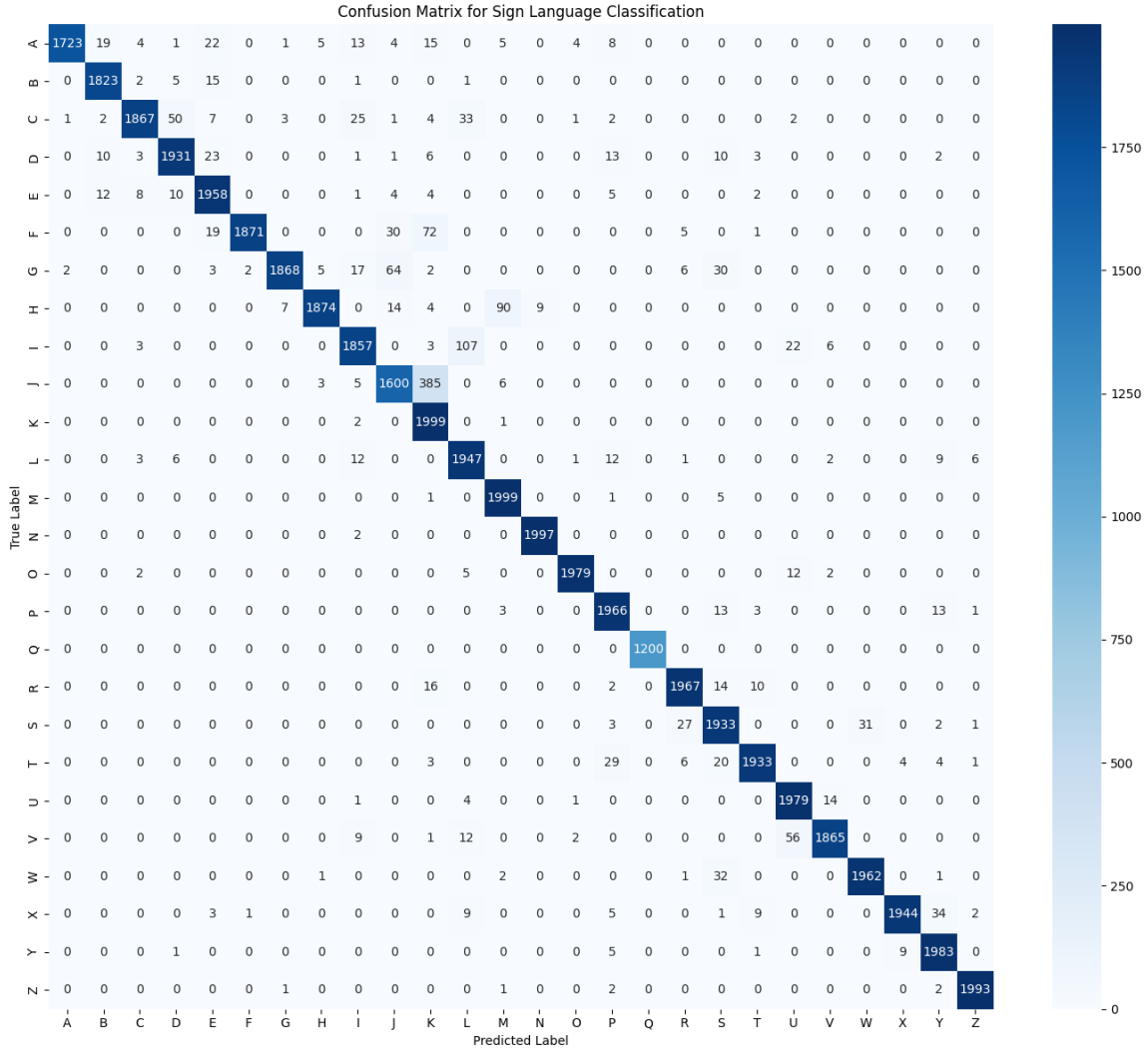

Figure 3. Confusion Matrix for the model

Key insights from the confusion matrix (**Figure 2**):

Most entries are concentrated along the diagonal, which aligns with the high overall accuracy (nearly all instances are correctly on the diagonal).

The diagonal values for many letters like M, N and O are at or very near the total number of test samples for that class, indicating perfect or near-perfect classification for those letters.

The off-diagonal entries are few. The largest confusion was observed between a couple of specific pairs of letters:

J and K: The model sometimes confuses the ISL signs for "J" vs "K". In ISL, the signs for J and K are very complex. They both involve the use of two hands and while J is a dynamic sign, K makes the use of two index fingers to form a K. The confusion matrix showed that out of the 2000 test images for "J", 385 were misclassified as "K". This is understandable because the visual difference can be subtle and might depend on the angle of view. Despite this, K still had high precision and recall >95%.

U vs V, and L vs C, I: The letters U and V in ISL follows the classic confusion. In ASL alphabets, U and V are distinguished by separating the two fingers or keeping them together. However, in ISL, U and V are very similar. The signer is expected to make their respective shape (thumb and index finger in the case of U and index and middle finger in the case of V) while keeping the back of their palm facing towards the camera. Our model had a handful of errors in this area. Apart from this, the model has also confused the letters C and I for the letter L. This is because both L and I uses a straight index finger, and the model fails to account for the missing thumb position. In case of C, the model is not able to take into account, the bend of the index finger and the thumb. Thus, mistakes it for an L. However, additional training data for these cases and perhaps specialized augmentation (rotating the hand slightly) could further mitigate this.
Virtually all other letters were recognized with near 100% accuracy in the test set.Letters such as "M", "N", "O", and "Z" achieved near-perfect or perfect classification, with no off-diagonal errors. This indicates the model's internal feature representations successfully capture those distinct shapes.

No signs of systematic bias were found. No class was disproportionately predicted or neglected. The balanced dataset and uniform preprocessing likely contributed to the even performance across all letters. The confusion matrix confirms that most errors were limited to subtle, visually similar pairs, and performance remained robust across all signs.
Given these results, the model's errors are mostly limited to confusions between visually similar gestures. This is expected, as even human interpreters might need context to distinguish some finger-spelled letters. In practical use, if one letter is misrecognized as a similar one, the user could repeat or slightly adjust their hand pose to get the correct output, which is a minor inconvenience.

4.3 - Real-Time System Evaluation

The real-time sign language translation system was put to the test in an indoor setup using just a standard webcam. The evaluation mainly looked at how accurate, fast, and user-friendly the

system was. Overall, it performed quite well, especially when users followed a few simple instructions, like keeping their hand inside a specific area on the screen and pausing briefly after each sign.

4.3.1 - Accuracy in Real-Time

In terms of accuracy, the system did a solid job recognizing most hand signs, as long as users stayed within the ideal conditions: facing the camera, holding their hand steady, and having decent lighting. Under those circumstances, the model hit over 90% confidence during live use, based on informal tests. Most of the errors weren't due to flaws in the model itself but came from everyday hiccups like users moving too fast, having only part of the hand in view, or dealing with uneven lighting. If someone held their hand too far from the webcam, or didn't pause between signs, the results became less reliable. But with a quick pause between letters, the system consistently got things right. It's worth noting that the system was built to recognize individual letters and not full, flowing sign language. It's basically a digital finger-spelling tool. When users tried to quickly spell words without pausing, the model sometimes got tripped up by the transitions between letters. These in-between moments didn't match any known sign, so the predictions could go off track. Encouraging users to pause between letters helped solve that.

From a usability standpoint, feedback was generally positive. People found the interface easy. They could see their hand in the video feed, and the predicted letter showed up in real time. After a short learning curve to get the hang of positioning and distance, users were able to sign letters and get accurate feedback. For those unfamiliar with sign language, a printed ISL alphabet chart was available, and even their first-time attempts were often recognized, showing the model's flexibility in interpreting a range of hand shapes.

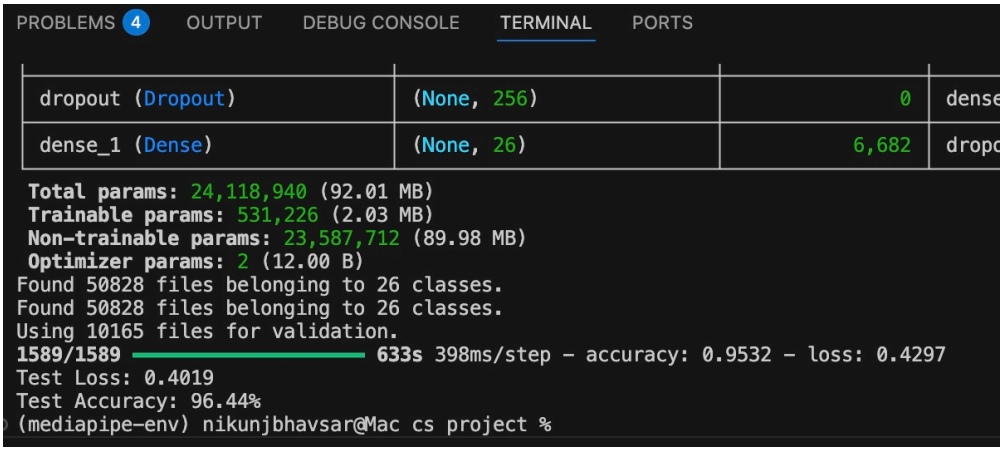

Figure 4. Validation Accuracy and Loss

4.3.2 - Comparison to Literature / Expectations

The gesture recognition model reached an overall accuracy of 96%, which is slightly below the 98–99% figures reported in some other studies (Saini et al., 2023). But context really matters here. A lot of those higher numbers come from models tested in more robust environments. For example, different lighting, different background, and often different people doing the gestures in both training and testing phases. Our dataset, by comparison, was much less varied. It included images from a single source along with a self-captured dataset. So, considering that, 96% accuracy is actually a strong result. It suggests the model isn't just memorizing patterns but learning to generalize, which is crucial for real-world use. And realistically, when accuracy is above 95%, users can usually just repeat or slightly adjust a gesture if it's misread, and the system still works smoothly (Vashisth et al., 2023).
In terms of performance, the system achieved what it set out to do and proved that a trained model can be used in real time, with just a regular webcam. It captures video, processes each frame, and predicts letters with minimal delay. That live interaction is a huge step beyond offline experiments and shows the system is practical, not just theoretical. It's a fully functioning prototype that responds to hand gestures on the fly, which is exactly what the project aimed for.

4.3.3 - Limitations and Lessons

Sure, there are some limitations to be aware of. One of the big ones is dataset bias. The model was trained on images with certain lighting conditions, hand sizes, and skin tones. While the model mostly relies on shape to recognize signs, it's still possible that users with very different hand features, like darker or lighter skin, tattoos, rings, or even brightly colored nail polish, might see a small dip in accuracy. The lack of broader training data means there's less guarantee for the same accuracy across all demographics. A more diverse dataset would definitely help make the model more robust.

Another challenge comes from using webcams in less-than-ideal environments. The model itself is solid, but things like poor lighting or a cluttered background can trip it up. If the room is too dark, the webcam feed gets noisy. This makes it harder for the model to pick up on hand shapes. These aren't problems with the model itself but with the setup. Adding things like hand detection or automatic brightness adjustments could go a long way toward fixing these issues, but they weren't part of this simpler version of the system.One more limitation is that the model only handles static gestures one frame at a time. That works just fine for finger spelling, where each letter has a static pose, but it struggles with signs that involve motion. For example, the

letter "J" didn’t get recognized consistently. Since the model was trained only on still images, it doesn’t understand motion or gestures that play out over multiple frames. Supporting dynamic gestures would require more advanced models like RNNs, 3D CNNs, or transformers. They would be a better option since they can process sequences instead of snapshots.

Despite the limitations, the project came with a lot of valuable lessons. One of the big takeaways was the tradeoff between model complexity and real-time performance. Sure, deeper models might be more accurate, but they’re also slower. ResNet50 turned out to be a great middle ground. Another important lesson was that the model isn’t everything. The entire user experience matters. Even a well-trained model can struggle if the camera setup is off, the lighting is bad, or users aren’t sure how to position their hands. Building a successful system means thinking beyond the model and designing the whole interaction.

5 - Conclusion and Future Work

5.1 - Conclusion

This project set out to develop a real-time Indian Sign Language gesture recognition and translation system, focusing on the ISL alphabet as a case study. By the end of the project, that aim was completely fulfilled. A deep learning-based system that can identify and translate static hand gestures of the ISL alphabet with a high degree of accuracy (96%) has been achieved. The system leverages a custom ResNet50 CNN model to perform the core recognition task and uses a simple yet effective approach to integrate this model into a live webcam-based interface. Instead of starting again, a transfer learning strategy was applied, removing the model's top classification layer and inserting a new head meant to identify 26 hand sign types. From the standpoint of implementation, Python and OpenCV were used to create a real-time system. It captures webcam frames, interprets them, and feeds them to the model for predictions. Users receive instant feedback since the expected letter and confidence score appear right on the video feed.

Many significant results were obtained during the course of the research. Augmentation helped to create and improve a clean, useable ISL alphabet dataset. It increased the variety of hand sizes and angles that the model could train on. Given the dataset's modest size and diversity, that was particularly crucial. Strong outcomes came from training and testing the ResNet50-based model. Using conventional hardware, a live, functioning prototype was created that obviously shows the promise of gesture recognition. The system was eventually put to use in the actual world to highlight its advantages as well as its shortcomings.

Naturally, like any first iteration, this system has certain constraints. It only identifies static movements. This entails that dynamic movement doesn't work. Moreover, given that it lacks a specific hand detector or image pre-processing for illumination, it's highly sensitive to lighting and background clutter. That stated, within the project range, these constraints were anticipated and managed as much as feasible. Under typical, supervised circumstances, the system still operates consistently well, and it demonstrates how AI and computer vision may help enable accessible communication in the future. The project also provided insightful education along the way. Having a good model turned out to be only one part of the puzzle; preprocessing, user interface, feedback systems, and general system design are equally vital. The modular design

facilitated rapid changes and testing. Working with actual users revealed problems that offline testing by itself would not have identified.

This offers a basis for future development in the Indian Sign Language community and demonstrates how deep learning may be used to solve practical, human-centered issues. Ultimately, this project successfully created a real-time Indian Sign Language (ISL) alphabet recognition system that works using just a trained model and a regular webcam. It shows how much machine learning and computer vision have advanced, and how they can be used to build helpful tools. Most importantly, it highlights the potential to reduce communication barriers for 430 million people relying upon sign language by making technology more inclusive and accessible

5.2 - Future Work

Although the project achieved its primary objectives, there is still much interesting possibility for where it could go next. The system now does a good job of identifying static ISL alphabet signs using a trained model and a webcam. Real-world sign language, however, is far richer and dynamic than single letters, and there are many interesting ways to carry on this effort.

Expanding the system's vocabulary beyond the alphabet would be one of the most immediate and useful changes. Future iterations could identify complete sentences or common ISL phrases like "hello," "thank you," "yes," or "no" instead of users spelling out every word. That would allow the model to be used far more in daily conversations. Getting there requires training the algorithm on a bigger dataset with whole-word gestures, including Hindi numerical and vowel signals to broaden its reach. For that purpose, datasets like the one by Singh et al. (2022) that include Hindi vowels and numerals could be incorporated.

Secondly, enabling the system to manage dynamic gestures, those involving movement rather than a single static posture would be another significant advance. Currently, the model examines each frame separately. Hence, it has difficulty handling motions or signs for complete sentences needing motion. Future versions could include transformer-based models meant for video input, 3D CNNs, LSTMs, or even sequence-based ones (Baihan et al., 2024). These would enable the system to see gestures as motion over time, hence improving its accuracy and realism. Also, the present configuration mainly works well only if users put their hand in a certain screen area, which is not always clear. Automatic hand tracking would remove that restriction and greatly improve the smoothness of the experience (Vashisth et al., 2023). A hand detection

model in the preprocessing pipeline, like MediaPipe, could locate the hand and direct the recognition process accordingly. In various settings, this would greatly increase the system's adaptability and user-friendliness.

Better management of uncertain forecasts would be another little but significant change. For example, if the model lacks confidence in what it observes, it can just tell the user to try again rather than displaying a possibly erroneous outcome. In live environments, something like a majority vote method (Nguyen and Le, 2022) over the last few frames could also assist in stabilising the forecasts and avoid flickering output while the hand is somewhat moving. Adapting the system for mobile or embedded platforms is a useful path to reach more people, particularly those on the go. It would allow consumers to carry the translator with them wherever they go. Naturally, this would call for some model optimisation to maintain speed and efficiency, perhaps even changing it to a format like TensorFlow Lite. Looking even further forward, one may also consider the prospect of constructing a bidirectional ISL translator. The system now converts signs into text or speech, but it may go the other way as well, turning spoken or typed words into sign language using a robotic hand or animated avatar. This has already been achieved for Spanish Sign Language (San-Segundo et al., 2008). In cases when interpreters are unavailable, that would allow for entirely two-way communication.

6 - Project Management Review

This section reflects on the management aspects of the project, including how the project timeline was planned and executed. It presents the tools and practices used for task management and lessons learned regarding project management. It provides a retrospective analysis of what went according to plan and what needed adjustment during the course of the project. It compares the initial planning (as outlined in the project outline and Gantt chart) with the actual development and progress made.

6.1 Timeline and Gantt Chart Summary

Initial Research and Literature Review: At the start of the project, a detailed Gantt chart was prepared outlining key phases, including dataset collection, real-time hand landmark detection, gesture classification, text and speech translation, user interface development, and testing. The timeline provided structured guidance, and though some phases evolved from the original plan, the project stayed within its scope and deadline.

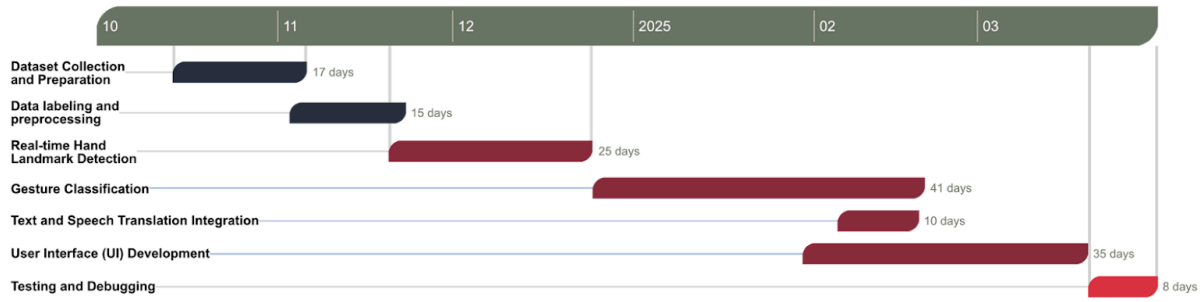

Figure 5. Original Gantt Chart (Reproduced from project outline (Bhavsar, 2024))

Dataset Acquisition/Preparation: Planned: Weeks 3–6 to find or create the dataset. Actual: This happened in weeks 4–7. The first planned phase was Dataset Collection and Preparation, scheduled to span 17 days (late October into November). This phase was completed as planned, though the scope of the dataset shifted. Instead of collecting hand landmark data compatible with MediaPipe, the focus shifted to static image-based ISL gesture data, which was more practical for training with ResNet50. This adjustment enabled earlier training experiments and aligned better with available resources. There was a slight delay as obtaining ISL images

took time. We adjusted by overlapping some of the model design work with the tail end of data gathering. The Gantt chart was updated to reflect that data prep continued alongside the beginning of model setup. Data labeling and preprocessing was carried out effectively within its planned 15-day window. Labeling was straightforward due to the alphabetical nature of the dataset, and the class distribution was verified before training.

The **Real-Time Hand Landmark Detection task**, initially planned as a key milestone using MediaPipe, was not implemented in the final system. Instead, the project pivoted to an image-based recognition pipeline using ResNet50 and OpenCV. This change significantly reduced system complexity while still achieving real-time performance. As a result, the time allocated for this phase was instead used to accelerate model development and early testing. **Gesture Classification**, originally intended to follow the landmark detection phase, became the central focus of the system. This phase began slightly earlier than scheduled and ran across the full training and evaluation window (about 6 weeks), as shown in the chart. Data augmentation was also integrated during this phase, and early stopping was used to ensure training efficiency. The Text and Speech Translation Integration, originally scheduled for 10 days, was partially implemented. While text output was successfully integrated and displayed in real time, the text-to-speech (TTS) feature was deprioritized due to time and scope constraints. The project's focus shifted toward stabilizing real-time gesture prediction and usability, which was a more immediate priority.

Testing and Evaluation was planned for the final 8 days, and this phase was carried out as scheduled. Testing included informal trials under varying lighting conditions, distances, and hand placements. Early debugging was also iteratively integrated throughout the project. For example, prediction accuracy and responsiveness were tested as soon as the model was integrated with the webcam. Overall, the Gantt chart proved helpful in structuring the timeline and task dependencies. While certain tasks were modified or skipped, these decisions were made to ensure feasibility within the academic term and available resources. The project remained within the original duration and met its core objectives. This project highlighted the importance of flexibility in execution. Although the technical approach changed from the initial plan, the final system still achieved a working, real-time ISL alphabet recognition prototype. The ability to adapt while keeping the primary goal in focus contributed greatly to the project's success.

Documentation and Final Report Writing: This task was planned to begin around February 2025 and continue throughout the project, with the final compilation and editing scheduled for

late March 2025. In practice, the report writing was carried out incrementally, with several sections drafted alongside project milestones. For instance, the Methodology chapter was developed around December 2024, once the technical approach was finalized. Initial results and evaluation findings were documented as they became available, particularly in January 2025. The final two weeks were dedicated to compiling all chapters, formatting, proofreading, and adding final visual elements such as the confusion matrix and accuracy plots. Writing sections earlier in the timeline, such as the Literature Review and Project Plan, proved to be an effective strategy and reduced last-minute pressure. According to the Gantt chart, the final report was scheduled to be completed by the end of March 2025, and this deadline was successfully met.

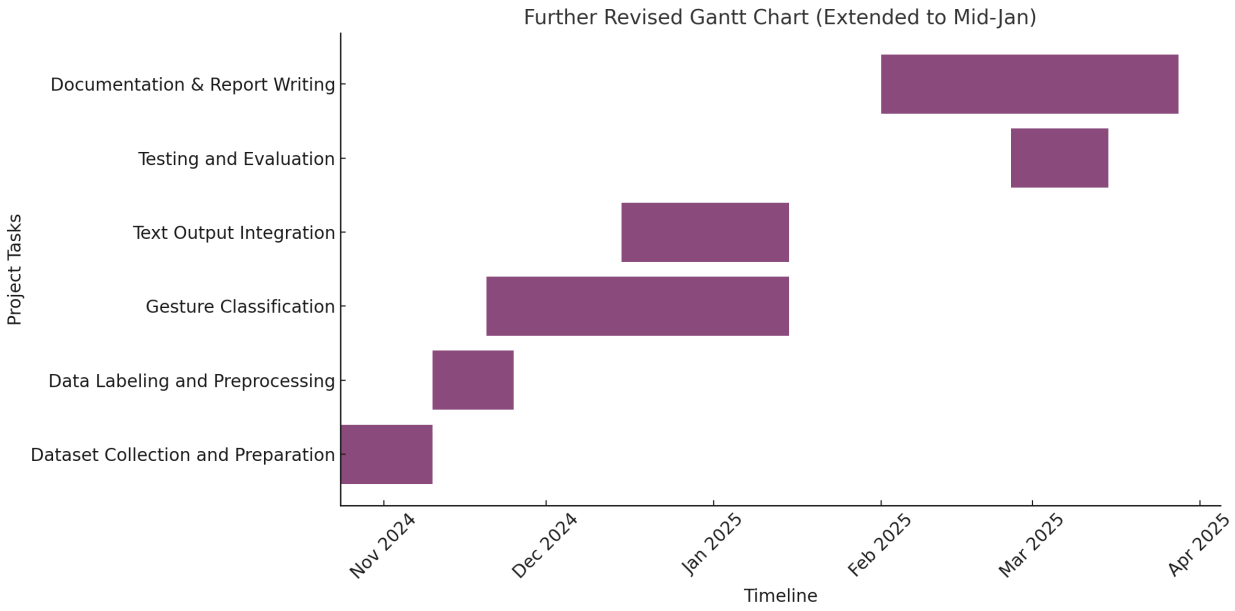

Figure 6. Revised Gantt chart

This project underscores the importance of flexibility and iterative decision-making in project management. Although the technical path diverged from the initial proposal, the primary objective remained central throughout. The ability to adapt to real-world challenges without losing sight of the end goal was a key contributor to the project's overall success. The original roadmap provided essential guidance, while thoughtful deviations allowed the system to evolve in a realistic and achievable direction. The experience reinforced the value of agile planning (Chin, 2004), early prototyping, and user-focused development. All of these are crucial when managing technology projects with real-time performance and usability considerations.

6.2 - Risks and Mitigation

During the project's planning stage, several major risks were noted, and suitable mitigating actions were taken to handle them. This guaranteed that the project stayed on course and achieved its goals even in case of any difficulties.

The possibility of not finding a good ISL dataset was one of the initial issues. A plan was to build a custom dataset if a good publicly accessible one was unavailable. A curated collection of ISL alphabet images was assembled and preprocessed. This method was carried out effectively. More time in the project timeline had been set aside to assist in this task, which helped to prevent delays in the next phases. The likelihood of the model not reaching the required degree of classification accuracy was another major concern. Several fallback plans were developed to reduce this. These consisted of using more aggressive data augmentation methods to enhance generalisation. Basic success was defined by a minimum goal accuracy of 85%. Practically, the chosen method using ResNet50 surpassed this bar, reaching about 96% accuracy on the validation set. This outcome confirmed the choice of model and lessened the need to rely on secondary tactics. Integration risks were also taken into account, especially in merging the trained model with camera input and real-time prediction output. Such activities sometimes create unanticipated performance or compatibility problems. Integration was not left to the end; rather, early testing was done with a partially trained model to lower the risk.

By means of this mix of forward planning, early prototyping, and adaptable fallback techniques, the project was able to control risks and produce all intended results without major delays or quality compromises.

6.3 - Lessons Learned in Project Management

Over the duration of this project, several important insights into efficient project management were acquired. These ideas greatly helped the project's successful delivery and showed best practices in handling complicated project tasks.

The necessity to stay flexible with the project plan was an important lesson. Although the Gantt chart offered a planned timetable and milestones, the capacity to change was vital when data collecting ran longer than anticipated. Time was transferred from less important tasks to keep momentum on fundamental components instead of letting this postpone the whole process. This

underlined the need of dynamic prioritisation in project execution. Another constant difficulty was time management, especially in juggling this assignment with other academic duties. Consistent development was guaranteed by the planned timetable; defining modest, precise weekly objectives was very successful. The micro-goals kept interest and guaranteed forward movement by breaking the larger project into doable tasks. Apart from that, the project concentrated on guaranteeing that the fundamental function was robust, accurate, and user-friendly rather than trying to implement every anticipated feature. To emphasise a strong implementation of the fundamental components, features like complete text-to-speech integration were deprioritized. This strategy fit the idea that providing a fully working core system is sometimes more important than showing a wide but unstable solution. Halfway through the project, there were thoughts to add more functionality like a complete web interface or multi-hand gesture detection. But the project remained focused by regularly coming back to the initial goals and timetable. This approach guaranteed that the outputs were finished on schedule and of excellent quality.

Looking back, the project was successfully delivered on time, fulfilling its main goals and operating as planned. The management strategy which was based on early planning, adaptive development, risk mitigation, and incremental progress proved successful. Dataset quality and size are still crucial for machine learning performance. Therefore the only significant change if the experiment were to be duplicated would be to start data collecting even earlier. Otherwise, the approaches and strategies adopted in this project proved solid project management and offered a strong platform for future work.

References

Vashisth H.K., Tarafder T., Aziz R., Arora M. and Alpana (2023). Hand Gesture Recognition in Indian Sign Language Using Deep Learning. Engineering Proceedings, [online] 59(1), p.96. doi:https://doi.org/10.3390/engproc2023059096.

Ijsrem.com. (2023). Real Time Indian Sign Language Recognition – IJSREM. [online] Available at: https://ijsrem.com/download/real-time-indian-sign-language-recognition/ [Accessed 29 Mar. 2025].

World Health Organization (2024). Deafness and hearing loss. [online] WHO. Available at: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

Zhang Y. and Jiang X. (2024). Recent Advances on Deep Learning for Sign Language Recognition. Computer Modeling in Engineering & Sciences, [online] 139(3), pp.2399–2450. doi:https://doi.org/10.32604/cmes.2023.045731.

Chin M. (2020). Wearable-tech glove translates sign language into speech in real time. [online] UCLA. Available at: https://newsroom.ucla.edu/releases/glove-translates-sign-language-to-speech.

Erard M. (2017). Why Sign-Language Gloves Don’t Help Deaf People. [online] The Atlantic. Available at: https://www.theatlantic.com/technology/archive/2017/11/why-sign-language-gloves-dont-help-deaf-people/545441/.

Ahmed M.A., Zaidan B.B., Zaidan A.A., Salih M.M. and Lakulu M.M. bin (2018). A Review on Systems-Based Sensory Gloves for Sign Language Recognition State of the Art between 2007 and 2017. Sensors, [online] 18(7), p.2208. doi:https://doi.org/10.3390/s18072208.

Singh A., Singh S.K., Mittal A., Gupta B.B. (2022), “Static gestures of Indian Sign Language (ISL) for English Alphabet, Hindi Vowels and Numerals ”, Mendeley Data, V1, doi: 10.17632/7tsw22y96w.1

ayeshatasnim-h (2020). GitHub - ayeshatasnim-h/Indian-Sign-Language-dataset: Indian Sign Language Alphabet Dataset. [online] GitHub. Available at: https://github.com/ayeshatasnim-h/Indian-Sign-Language-dataset.

Saini B., Venkatesh D., Chaudhari N., Shelake T., Gite S. and Pradhan B. (2023). A comparative analysis of Indian sign language recognition using deep learning models. Forum for linguistic studies, 5(1), pp.197–197. doi:https://doi.org/10.18063/fls.v5i1.1617.

Zhang Y. and Jiang X. (2024) 'Recent advances on deep learning for sign language recognition,' Computer Modeling in Engineering & Sciences, 139(3), pp. 2399–2450. https://doi.org/10.32604/cmes.2023.045731.

Nandi U., Ghorai A., Singh M., Changdar C., Bhakta S., Pal R. (2022). Indian sign language alphabet recognition system using CNN with diffGrad optimizer and stochastic pooling. Multimedia Tools and Applications. 82. 10.1007/s11042-021-11595-4.

He K., Zhang X., Ren S., and Sun J. (2016a) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 27–30 June 2016, pp.770–778. New York: IEEE.

He K., Zhang X., Ren S., and Sun J. (2016b) Identity mappings in deep residual networks. In: Leibe B, Matas J, Sebe N, and Welling M (eds.) Computer Version—ECCV 2016. Berlin: Springer, pp.630–645.

Wikipedia contributors (2025) Residual neural network. https://en.wikipedia.org/wiki/Residual_neural_network#:~:text=,3.

Baihan, A., Alutaibi A., Alshehri M., Sharma S.K. (2024) 'Sign language recognition using modified deep learning network and hybrid optimization: a hybrid optimizer (HO) based optimized CNNSa-LSTM approach,' Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-76174-7.

San-Segundo R., Barra R., Córdoba R., D'Haro L.F., Fernández F., Ferreiros J., Lucas J.M., Macías-Guarasa J., Montero J.M. and Pardo J.M. (2008). Speech to sign language translation system for Spanish. Speech Communication, 50(11-12), pp.1009–1020. doi:https://doi.org/10.1016/j.specom.2008.02.001.

Mukti, I.Z. and Biswas, D., 2019, December. Transfer learning based plant diseases detection using ResNet50. In *2019 4th International conference on electrical information and communication technology (EICT)* (pp. 1-6). IEEE.

Hossain, M.B., Iqbal, S.H.S., Islam, M.M., Akhtar, M.N. and Sarker, I.H., 2022. Transfer learning with fine-tuned deep CNN ResNet50 model for classifying COVID-19 from chest X-ray images. *Informatics in Medicine Unlocked*, *30*, p.100916.

Deora, D. and Bajaj, N., 2012, December. Indian sign language recognition. In 2012 1st international conference on emerging technology trends in electronics, communication & networking (pp. 1-5). IEEE.

Raheja, J.L., Mishra, A. and Chaudhary, A., 2016. Indian sign language recognition using SVM. *Pattern Recognition and Image Analysis*, *26*, pp.434-441.

Hore, S., Chatterjee, S., Santhi, V., Dey, N., Ashour, A.S., Balas, V.E. and Shi, F., 2017. Indian sign language recognition using optimized neural networks. In Information Technology and Intelligent Transportation Systems: Volume 2, Proceedings of the 2015 International Conference on Information Technology and Intelligent Transportation Systems ITITS 2015, held December 12-13, 2015, Xi'an China (pp. 553-563). Springer International Publishing.

Gupta, S., Gilotra, S., Rathi, S., Choudhury, T. and Kotecha, K., 2024, January. Plant Disease Recognition Using Different CNN Models. In *2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 787-792). IEEE.

Farooq, U., Asmat, A., Rahim, M.S.B.M., Khan, N.S. and Abid, A., 2019, November. A comparison of hardware based approaches for sign language gesture recognition systems. In *2019 International Conference on Innovative Computing (ICIC)* (pp. 1-6). IEEE.

Al-Qurishi, M., Khalid, T. and Souissi, R., 2021. Deep learning for sign language recognition: Current techniques, benchmarks, and open issues. *IEEE Access*, *9*, pp.126917-126951.

Kudrinko, K., Flavin, E., Zhu, X. and Li, Q., 2020. Wearable sensor-based sign language recognition: A comprehensive review. *IEEE Reviews in Biomedical Engineering*, *14*, pp.82-97.

Chin, G., 2004. Agile project management. *AMACOM, New York*, pp.4-5.

Gupta, J., Pathak, S. and Kumar, G., 2022, May. Deep learning (CNN) and transfer learning: a review. In *Journal of Physics: Conference Series* (Vol. 2273, No. 1, p. 012029). IOP Publishing.

Indian Sign Language Dataset (2020). https://www.kaggle.com/datasets/vaishnaviasonawane/indian-sign-language-dataset.

Appendix A:

Code for Capturing Images

```python
import cv2
import os
import time

dataset_dir = "dataset"
if not os.path.exists(dataset_dir):
    os.makedirs(dataset_dir)

#webcam
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open webcam")
    exit()
print("Press keys A-Z to capture an image for that class.")
print("Press 'q' to quit.")
while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame")
        break

    cv2.imshow("Webcam", frame) # Display the webcam
    key = cv2.waitKey(1) & 0xFF # Wait for a key press for 1ms
    if key == ord('q'):
        break  # 'q' to quit

    if (65 <= key <= 90) or (97 <= key <= 122): # Check if the pressed key is a letter (A-Z or a-z)
        label = chr(key).upper() # Create label of the key pressed
        class_dir = os.path.join(dataset_dir, label)
        if not os.path.exists(class_dir):
            os.makedirs(class_dir)

        filename = os.path.join(class_dir, f"{time.time()}.jpg") # Create a unique filename
        cv2.imwrite(filename, frame)
        print(f"Saved image for class '{label}' to: {filename}")
cap.release() # Shut off webcam and delete that window
cv2.destroyAllWindows()

```