**A PROJECT REPORT**

**on**

**“Real-Time ASL Hand Sign Recognition Using Convolutional Neural Networks”**

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfillment of the Requirement for the Award of**

**BACHELOR'S DEGREE IN**

**COMPUTER SCIENCE & ENGINEERING**

**BY**

**Irina Mehra**  2205129

**Snehal Singh**  2205164

**Sarwagya** 22051788

**Sparsh** 22052421

**Sai Pratikshya Sahoo** 22053533

**Shivam Kumar Singh** 22053546

**UNDER THE GUIDANCE OF**

**Dr. Sujata Swain**



**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**April 2025**

**A PROJECT REPORT**

**on**

**“Real-Time ASL Hand Sign Recognition Using Convolutional Neural Networks”**

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfillment of the Requirement for the Award of**

**BACHELOR'S DEGREE IN**

**COMPUTER SCIENCE & ENGINEERING**

BY

**Irina Mehra**  2205129

**Snehal Singh**  2205164

**Sarwagya** 22051788

**Sparsh** 22052421

**Sai Pratikshya Sahoo** 22053533

**Shivam Kumar Singh** 22053546

**UNDER THE GUIDANCE OF**

**Dr. Sujata Swain**



**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA -751024**

**April 2025**

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024



CERTIFICATE

This is certify that the project entitled

“Real-Time ASL Hand Sign Recognition Using Convolutional Neural Networks”

submitted by

|  |  |
| --- | --- |
| Irina Mehra | 2205129 |
| Snehal Singh | 2205164 |
| Sarwagya | 22051788 |
| Sparsh | 22052421 |
| Sai Pratikshya Sahoo | 22053533 |
| Shivam Kumar Singh | 22053546 |

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Sci-ence & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2024-2025, under our guidance.

Date: 08 / 04 / 2025

( Dr. Sujata Swain)

Project Guide

**Acknowledgements**

We are profoundly grateful to **Dr. Sujata Swain** of **Affiliation** for her expert guidance and continuous encouragement throughout to see that this project meets its target since its commencement to its completion.

Irina Mehra

Snehal Singh

Sarwagya

Sparsh

Sai Pratikshya Sahoo

Shivam Kumar Singh

**ABSTRACT**

A significant advancement in the deaf-mute community's communication, sign language recognition has long been a vital area of study. Creating an interface with high recognition that is independent of the user should be the goal of a real-time hand gesture recognition system. Nowadays, convolutional neural networks (CNNs) show remarkable recognition rates in image classification tasks.

This article introduces a Convolutional Neural Network (CNN)-based method for detecting hand signs in American Sign Language (ASL). ASL is widely used by the hearing-impaired community, and automatic detection of sign language can improve communication accessibility. Twenty percent of the dataset is set aside for testing, while the remaining 80 percent is used to train our model. In terms of hand gesture recognition, experiments utilising deep learning techniques in the suggested system yield encouraging results. With the suggested model, testing accuracy has been attained at about 96.678%.

**Keywords:** Sign Language Recognition, Convolutional Neural Network (CNN), Hand Gesture Recognition, American Sign Language (ASL), Deep Learning, Image Classification.

Contents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Introduction | | | 1 |
| 2 | Basic Concepts/ Literature Review | | | 2 |
|  | 2.1 | American Sign Language (ASL) Basics | | 2 |
|  | 2.2 | Convolutional Neural Networks (CNNs) for Image Recognition | | 2 |
|  | 2.3 | Existing ASL Recognition Systems | | 2 |
| 3 | Problem Statement / Requirement Specifications | | | 3 |
|  | 3.1 | Project Planning | | 3 |
|  | 3.2 | problem Statement & Requirement | | 3 |
|  | 3.3 | System Design | | 3 |
|  |  | 3.3.1 | Data Selection & Preparation | 3 |
|  |  | 3.3.2 | CNN Model Architecture | 3 |
| 4 | Implementation | | | 4 |
|  | 4.1 | Methodology | | 4 |
|  | 4.2 | Training and Validation | | 4 |
|  | 4.3 | Result and Evaluation | | 4 |
|  |  | 4.3.1 Evaluation Metrics | | 4 |
|  |  | 4.3.2 Accuracy curve and loss curve | | 4 |
|  |  | 4.3.3 Real time sign language detection | | 4 |
|  | 4.4 | Model Optimization and Deployment | | 4 |
| 5 | Standard Adopted | | | 5 |
|  | 5.1 | Design Standards | | 5 |
|  | 5.2 | Coding Standards | | 5 |
|  | 5.3 | Evaluation Standards | | 5 |
| 6 | Conclusion and Future Scope | | | 6 |
|  | 6.1 | Conclusion | | 6 |
|  | 6.2 | Future Scope | | 6 |
| References | | | | 7 |
| Individual Contribution | | | | 8 |
| Plagiarism Report | | | | 9 |

Chapter 1

Introduction

Communication is a vital part of human interaction, and for individuals with hearing and speech impairments, American Sign Language (ASL) serves as a primary mode of expression. However, a significant barrier exists between ASL users and those who do not understand sign language, making daily communication challenging and limiting accessibility. To bridge this gap, there is a growing need for automated ASL recognition systems that leverage advancements in artificial intelligence (AI) and deep learning.

Earlier ASL recognition methods primarily relied on sensor-based technologies, such as gloves and motion sensors, to track hand movements. While effective, these solutions tend to be costly, intrusive, and less adaptable to real-world scenarios. The emergence of computer vision and deep learning has introduced a more efficient alternative by utilizing Convolutional Neural Networks (CNNs) for image-based ASL recognition. However, existing systems often struggle with low accuracy, difficulties in recognizing intricate hand gestures, and challenges in real-time application.

To overcome these issues, this project proposes an ASL hand sign detection system based on a CNN model integrated with ResNet50, a deep learning architecture renowned for its high accuracy in image recognition tasks. ResNet50 employs residual learning, which mitigates the vanishing gradient problem, enabling the training of deeper neural networks. By combining CNNs with ResNet50, this project aims to enhance the accuracy, efficiency, and real-time usability of ASL recognition systems.

This report offers an in-depth exploration of the project, covering fundamental concepts, literature review, and existing ASL recognition approaches. It further defines the problem statement and system requirements, followed by the model's implementation and training details. The model's performance is assessed using evaluation metrics such as accuracy, precision, recall, and F1-score. The report concludes by summarizing key findings, suggesting future improvements, and discussing potential real-world applications.

Through deep learning advancements, this project seeks to make ASL recognition more precise, accessible, and scalable, ultimately facilitating better communication for individuals with hearing and speech impairment

Chapter 2

Basic Concepts/ Literature Review

**2.1 American Sign Language (ASL) Basics**

What is American Sign Language?

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing and is used by some hearing people as well.

The importance of sign language

Sign language plays a central role in preserving and celebrating Deaf culture. Sign language is an essential aspect of Deaf culture, enabling the preservation and celebration of heritage, while creating inclusivity and understanding in society. By recognising the importance of sign language, both the hearing and Deaf communities can build a more inclusive world where Deaf individuals can fully participate and contribute without discrimination or prejudice.

Whether it’s in education, professional settings or day-to-day life, the role of sign language is crucial in ensuring the empowerment and visibility for members of the Deaf community. So, let’s continue to appreciate and support sign language as a vibrant and powerful means of communication, celebrating the richness of Deaf culture.

Challenges in Automated Recognition of ASL:

* **Complexity of Gestures**: ASL consists of intricate hand movements, finger positions, and facial expressions, making recognition difficult.
* **Contextual Understanding**: Many signs change meaning based on facial expressions and sentence structure, requiring advanced AI models to interpret them correctly.
* **Variability in Signing Styles**: Different signers may have unique styles, speeds, and hand sizes, making standardization challenging.
* **Real-time Processing**: High-speed and accurate recognition is needed for effective real-time translation, requiring robust hardware and software solutions.
* **Limited Datasets**: ASL datasets for training machine learning models are relatively scarce compared to spoken language corpora.

**2.2 Convolutional Neural Networks (CNNs) for Image Recognition**

Convolutional Neural Networks (CNNs) are a specialized class of neural networks designed to process grid-like data, such as images. They are particularly well-suited for image recognition and processing tasks. They are inspired by the visual processing mechanisms in the human brain, CNNs excel at capturing hierarchical patterns and spatial dependencies within images.

**CNN Architecture:**

* **Convolutional Layers:** These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.
* **Pooling Layers** (e.g., Max Pooling): Reduce the spatial dimensions of feature maps while preserving important information. It helps in reducing computation and preventing overfitting.
* **Activation Functions:** They introduce non-linearity to the model, allowing it to learn more complex relationships in the data.
* **Fully Connected (Dense) Layers:** These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.
* **Output Layer:** Provides the final classification result, such as recognizing a specific ASL sign.



CNNs are ideal for ASL recognition because:

* Feature Extraction: They automatically learn spatial features from hand gestures, facial expressions, and motion.
* Translation and Rotation Invariance: CNNs recognize signs even with slight variations in hand positions and orientations.
* Hierarchical Learning: Captures low-level (edges) and high-level (gesture patterns) features progressively.
* Scalability: CNNs can be trained on large datasets to improve accuracy in recognizing diverse ASL signs.

**2.3 Existing ASL Recognition Systems**

Early attempts at American Sign Language (ASL) recognition relied primarily on specialized gloves or other hand-tracking devices equipped with sensors to measure finger positions and hand movements, which provided a means of capturing sign language's complex gestures but were limited by bulkiness, high cost, and unnatural signing motions, preventing widespread adoption.

How Did Early Glove-Based Systems Work?

Sensor Integration: These gloves have flex sensors, which vary resistance with bending. By positioning these sensors along the fingers and palm, the glove could detect the degree of finger flexion at each joint and provide information about hand posture.

Data Acquisition: When a signer donned the glove and made a sign, sensor readings were recorded and sent to a computer system.

The system extracts relevant elements from sensor data, such as finger angles and palm position, and uses machine learning techniques to identify them as known indications.

Limitations of Early Glove-Based Approaches:

Cumbersome Equipment:

The large gloves may inhibit normal hand movement, making signing appear unnatural and uncomfortable for users.

High-quality sensor gloves can be costly and cumbersome to use due to frequent calibration needed to maintain accuracy.

Limited Sign Recognition: These systems relied heavily on hand posture data, making it difficult to recognize complicated signs with subtle motions or facial expressions.

Wearing a customized glove limited the signer's natural expression.

Current Trend: Vision-Based Approaches: As computer vision and machine learning have advanced, researchers are turning to vision-based ASL identification systems that employ cameras to directly record hand gestures, providing a more organic and approachable method:

Hand Tracking and Detection:

Algorithms use contour analysis and other methods to locate the hand region in video frames.

Feature Extraction: After locating the hand, the video sequence is examined for information such as palm orientation, joint angles, fingertip positions, and hand motion.

Machine Learning Classification: To categorize the hand movements into the appropriate signals, these extracted features are placed into machine learning models (such as Convolutional Neural Networks).

Principal Benefits of Vision-Based Methods:

Natural Signing: Without the usage of protective gloves, signers can use their natural hand motions.

Accessibility: Most gadgets have cameras, which makes it easier for consumers to access.

Increased Accuracy: A greater variety of ASL signs can be recognized with high accuracy using more sophisticated deep learning models.

Problems with Vision-Based Methods:

Hand Occlusion Handling: Accurate recognition may be difficult when a hand is partially obscured by clothing, for example.

Lighting Variations: Hand detection and feature extraction may be impacted by variations in lighting.

Real-time Processing: Computationally efficient techniques are needed to achieve high accuracy in real-time.

Chapter 3

Problem Statement / Requirement Specifications

**PROBLEM STATEMENT**

The deaf-dumb community's main social problems are communication barriers between the hearing majority and the deaf-dumb minority, which prohibit them from obtaining basic and needed life services .

**Statement and Requirements**

**3.1 Project Planning**

The project planning for our ASL detection system, designed for high accuracy using a CNN model, follows a structured 10-week (approximately) timeline. We've segmented the development into four distinct phases:

* Literature review and dataset preparation,
* Focused model development and training utilizing Google Colab's GPU resources
* Testing and performance evaluation Cross-validation
* Documentation and presentation preparation.

Resource allocation includes leveraging Python with different types of libraries :

* TensorFlow/Keras for implementation
* OpenCV-contrib-python
* Split-folders

**3.2 Project Analysis**

The project analysis for our ASL detection model commenced with a meticulous examination of the defined accuracy and real-time processing requirements. We analyzed the complexity of ASL gesture recognition, considering the nuanced hand movements and better performance .We carefully looked at the data we had to train our model. If it showed different kinds of signs and if it had any problems that could make the model learn things wrong and provide error. We found that Google Colab’s (GPU) only had 15GB of memory, which was a big problem. This meant we had to be very careful with the size of the pictures we used, so we wouldn't run out of memory. To solve this, we used black and white pictures instead of color. This helped us use less memory and made the model run faster, without losing too much important information, which made the system easy to understand .

Potential hazards were noted, including difficulties in attaining real-time performance on target hardware or model overfitting as a result of few dataset variances. Mitigation solutions were created, such as model improvement to increase inference speed and data augmentation techniques to increase dataset diversity. Lastly, we examined how changes in lighting and background noise can affect the accuracy of the model and provided instructions for improving the model's resistance to these environmental influences.

**3.3 System Design**

We describe the systematic approach taken to design a CNN model for high-accuracy ASL gesture detection. This includes a thorough justification of our architectural choices, dataset handling procedures, training and testing methodologies,and real time execution, all aimed at minimizing errors and improving recognition precision.

**3.3.1 Design Constraints**

Software constraints :

* **Programming Languages:** Python 3.9 (Libraries: TensorFlow, Keras, OpenCV and etc ).
* **Operating System:** Window 11 .
* **Development Environment:** Google Colab and Visual Studio code. .

Hardware constraints :

* **GPU/CPU:** Model training was conducted on a Lenovo IdeaPad Flex 5 laptop equipped with an Intel Core i7 processor, 16GB RAM, and an integrated Intel Iris Xe GPU. Inference was tested on both CPU and GPU to enhance performance.
* **RAM and Storage:** Machine is attached with 8GB RAM and 512 GB storage space .
* **Camera/Sensor :** Code using frame size of (128,128, 1) and pixel of 200x200.

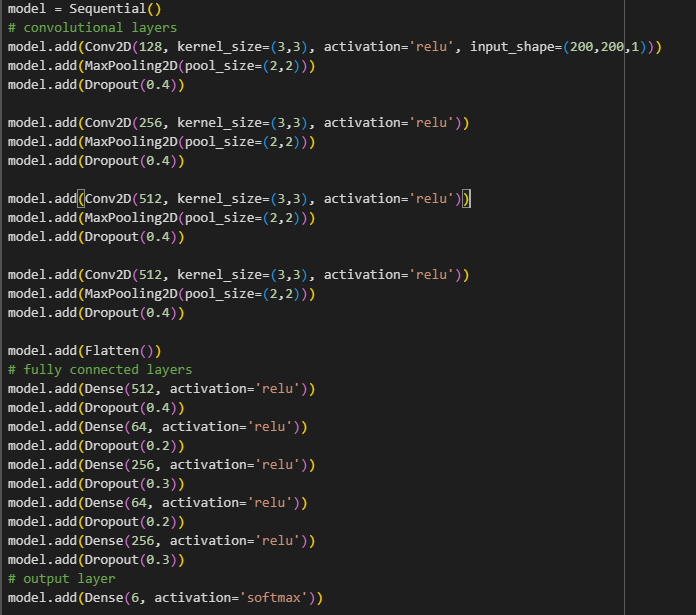
Environment setup:

* **Lighting Conditions:** Data was collected under bright lighting to minimize variations ,confusion and to have a better look .
* **Background:** Clear white background for enhancement of data set .
* **Distance and Angle:** Specific distance between machine’s camera and human hand for better resolution and system focus .
* **Data Collection Environment:** Data collection was conducted in a controlled laboratory setting.

**3.3.2 System Architecture OR Block Diagram**

The model architecture was built using the Keras framework. The validation dataset was used to obtain optimal settings for the CNN model. The input shape of the images is given as (128,128, 1), indicating that the images have a resolution of 200x200 pixels and contain one color channel (Gray) to represent. Grey images were used to provide simple information to help the model easily.

**Sequential Code for CNN Model Architecture-:**



**Code Explanation:**

* **Input Shape:** Input Shape is (128,128,1) as it is best fit for the human hand and made easy to understand for a system . It is grayscale to reduce color mismatch issues .
* **Kernel Size:** A 3x3 kernel allows the CNN to capture and examines a small, focused region of pixels at a time.
* **Activation Function:**  It solves the vanishing gradients issue.
* **Pooling Layers:** It is used in CNN model to reduce the spatial dimensions of the input feature maps while retaining the most important information .
* **Dropout:** Helps to prevent overfitting .

**Hardware Design**

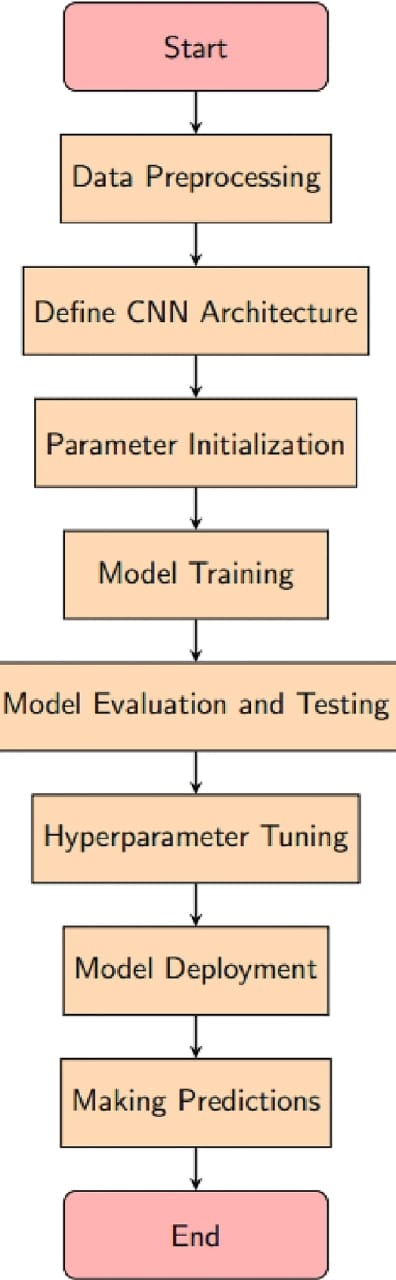


Fig.1 Block diagram representing the system design .

Chapter 4

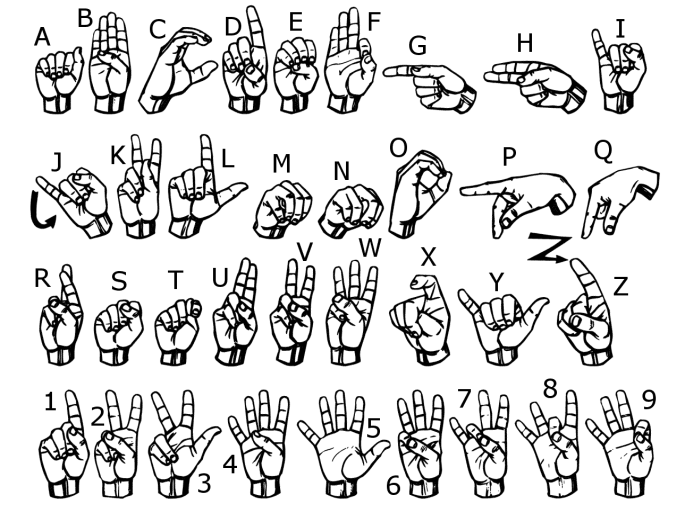
Implementation

**4.1 Methodology**

Our approach involves dataset preprocessing, model training, and evaluation. The ASL dataset is resized, converted to grayscale, and normalized before being split into 80% training and 20% validation. We use Python, Keras, TensorFlow, and NumPy to develop a CNN model with Conv2D, MaxPooling, and Dropout layers, and also implement ResNet50 for comparison. The model is trained for 100 epochs, and performance is analyzed using accuracy, precision, recall, F1-score, and confusion matrices. The CNN model achieves 94% accuracy, while ResNet50 underperforms at 66%, indicating areas for improvement.

**4.1.1 Dataset**

Our dataset consists of 1987 labeled images categorized into thirty-six different classes ( A to Z & 0 to 9). Each image is assigned a label corresponding to its class facilitating supervised learning.



American Sign Language [ASL benchmark dataset]

The dataset consists of grayscale images of ASL hand signs, resize to 128x128 pixels, representing different sign language gestures. It is divided into training (80%) and validation (20%) sets for model training and evaluation.

**4.1.2 Programming Language & Libraries Used:**

Language: Python is the primary language for model development.

Libraries:

Keras: Model building (Sequential), layers (Dense, Conv2D, Dropout, etc.), and one-hot encoding (to\_categorical).

TensorFlow: Image preprocessing (ImageDataGenerator).

NumPy: Numerical computations.

OS: File and directory handling.

Matplotlib & Seaborn: Data visualization.

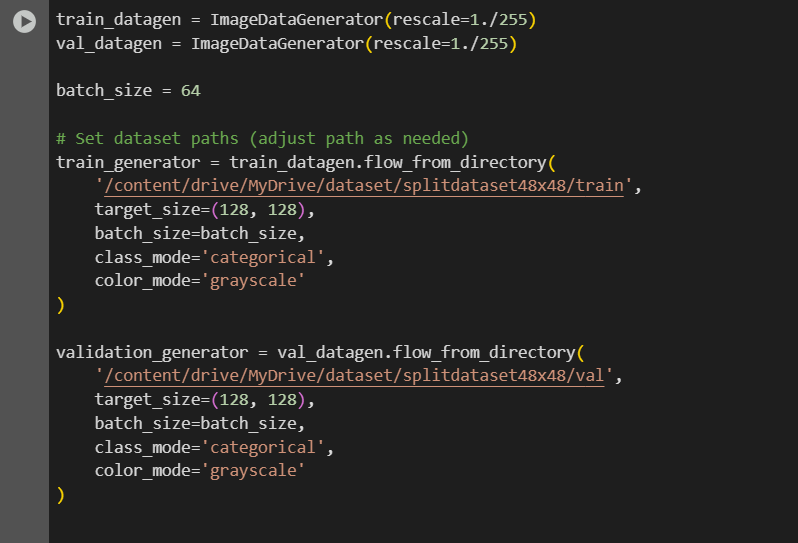
Scikit-learn: Model evaluation (classification\_report, confusion\_matrix).

**4.1.3 Preprocessing**

The code uses ImageDataGenerator to preprocess ASL hand sign images, normalising pixel values (0-1), scaling them (128x128), and converting them to grayscale. It reads training and validation data from specified directories in batches of 64. The dataset has been formatted for multi-class classification using a CNN model's categorical mode.

### **Preprocessing Steps for Real-Time Sign Language Detection:**

1. **Resizing**: Resizing the incoming video frame to the desired input shape (128x128).
2. **Grayscale Conversion**: Convert the frame to grayscale to simplify the input.
3. **Normalization**: Normalize the pixel values to the range [0, 1].
4. **Prediction**: Use the pre-trained model to predict the sign language gesture from the preprocessed frame.



**OUTPUT**

A - F contains 746 images

G - L contains 606 images

M - R contains 625 images

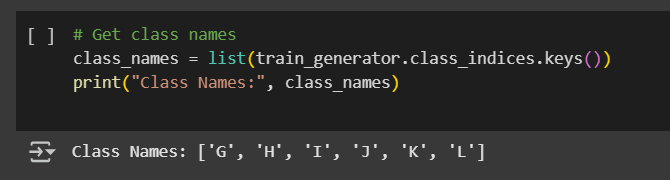
S - X contains 764 images

Y - Z & 0 - 3 contains 606 images

4 - 9 contains 606 images

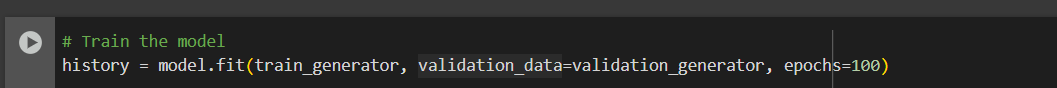
Number of images in the training set: 3150

Number of images in the validation set: 803

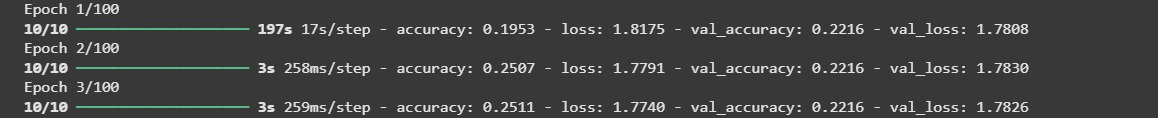


The code extracts and outputs the class labels (ASL hand signs) from the training dataset.

**4.2 Training and Validation**



* history = Stores training details (loss, accuracy) for analysis.
* model.fit(...) Trains the CNN using the dataset.
* train\_generator Supplies preprocessed training images in batches.
* validation\_data=validation\_generator Evaluates model performance to detect overfitting.
* epochs=100 Trains for 100 cycles; too many may cause overfitting.

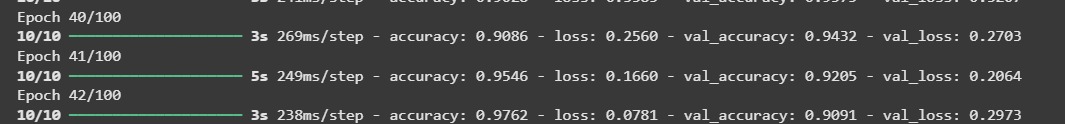


**Train Accuracy:** Improved from **19.53% →25.11%** (indicating slight learning).

**Train Loss:** Decreased (**1.8175 → 1.7740**), suggesting gradual optimization.

**Validation Accuracy:** Stagnant at **22.16%**, implying minimal generalization improvement.

**Validation Loss:** Slight decrease (**1.7808 → 1.7826**), but still high, indicating slow progress.



**Train Accuracy:** Improved from **90.86% → 97.62%** (model learning well).

**Train Loss:** Decreased significantly (**0.2560 → 0.0781**), indicating better optimization.

**Validation Accuracy:** Fluctuating (**94.32% → 90.91%**), suggesting **overfitting**.

**Validation Loss:** Increased in Epoch 42 (**0.2703 → 0.2973**), confirming overfitting risk.

**4.3. Results and Discussion**

In this subsection, the output of the experiment or study in terms of some graphs, plots must be presented. Also, if some implementation is done then its screenshots can be presented here, so as to showcase the proof of the output.**Evaluation Metrics**

**4.3.1 Evaluation Metrics**

We evaluate our model using both performance metrics and a confusion matrix. We use following

metrics for performance evaluation:

**Accuracy** : The percentage of correct predictions.

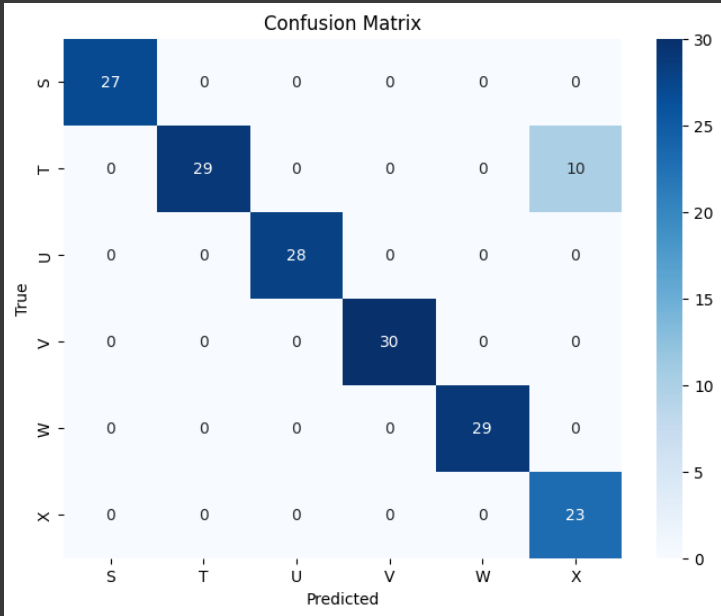
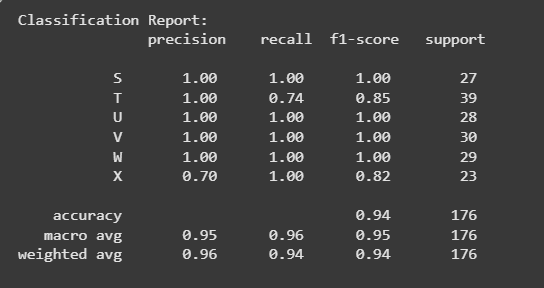
**Precision & Recall :** Measure how well each class is predicted.

**F1-score:**Harmonic mean of precision and recall to balance false positives and false negatives.

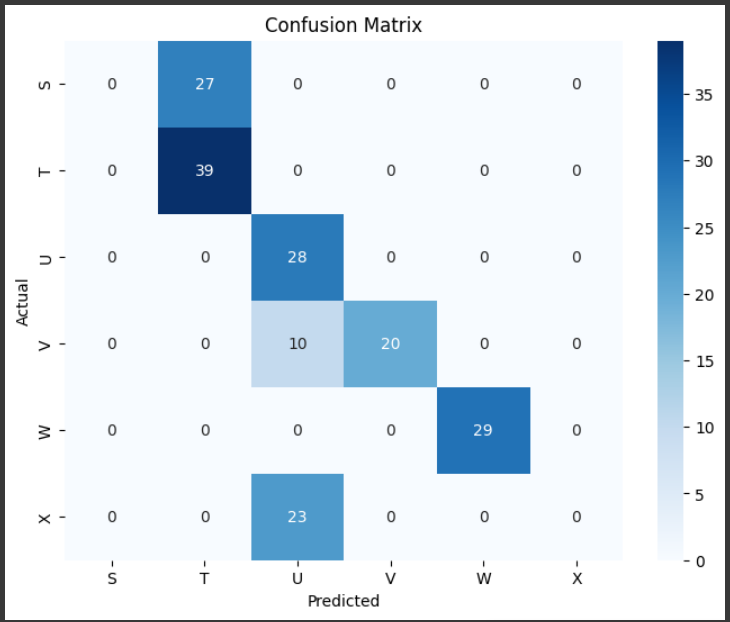
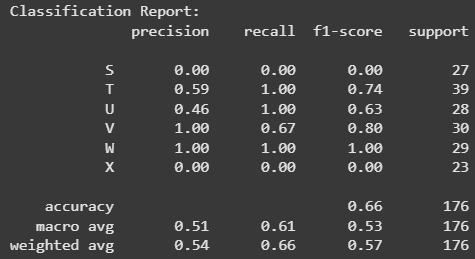
**Confusion Matrix :** A confusion matrix shows the count of true positives, false positives, true negatives, and false negatives, helping to evaluate a model's classification performance and error patterns.

Below are the confusion matrices and performance metrics for CNN Model and ResNet50 Model:

**Custom CNN Metrices:**

 ****

**ResNet50 Metrices:**

**Model Performance Analysis**

**CNN Model**

94% accuracy, 96% precision, 94% recall, and 94% F1-score for the CNN model

Observations from the Confusion Matrix: The model does well on the majority of classes, particularly S, U, V, and W, but it has trouble with T (lower recall) and X (lower precision).

In conclusion, the CNN model performs well generally, while it has trouble differentiating between classes like T and X. Although its high accuracy and F1-scores indicate that it is a dependable model, the misclassification of related classes may be addressed with additional model enhancements or tuning.

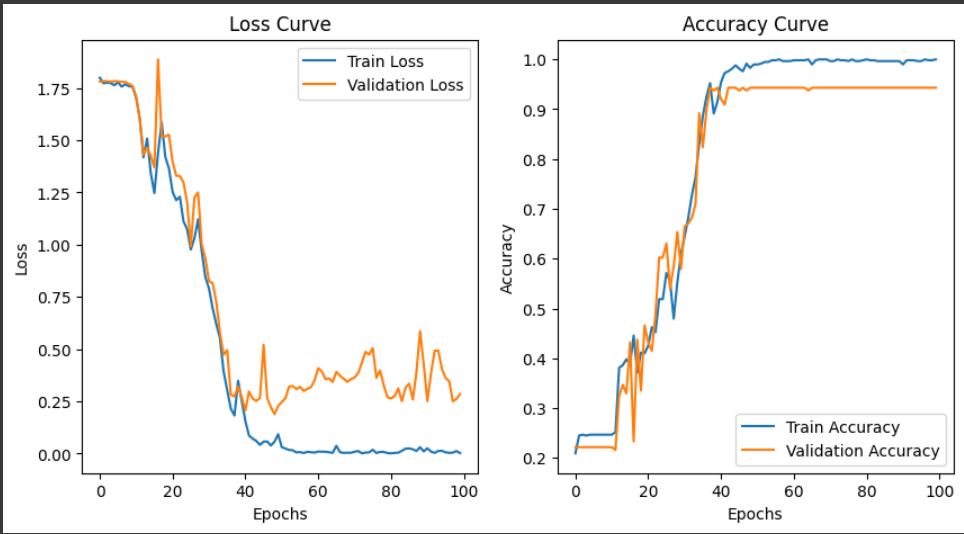
**ResNet50 Model**

F1-score: 57%, Accuracy: 66%, Precision: 54%, and Recall: 66%

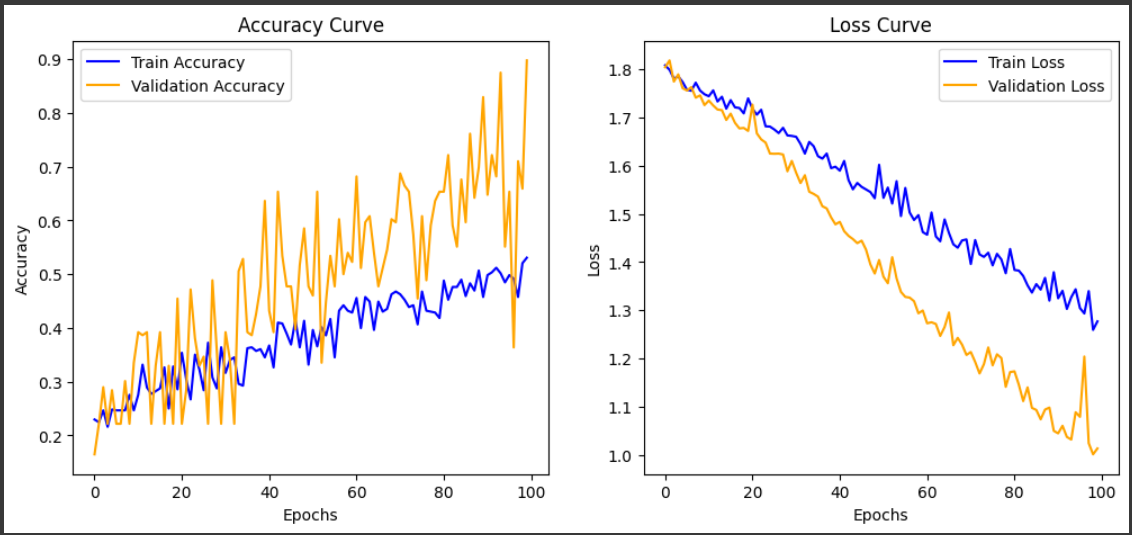
Confusion Matrix Observations: The ResNet50 model performs better on T and W but very poorly on S and X, with precision and recall close to zero.

In conclusion, the ResNet50 model performs substantially worse overall and has severe difficulties with certain classes, particularly S and X. Its accuracy and F1-scores show that it requires improvement and isn't working correctly.

**4.3.2 Accuracy Curve and Loss Curve  
  
1. CNN MODEL**

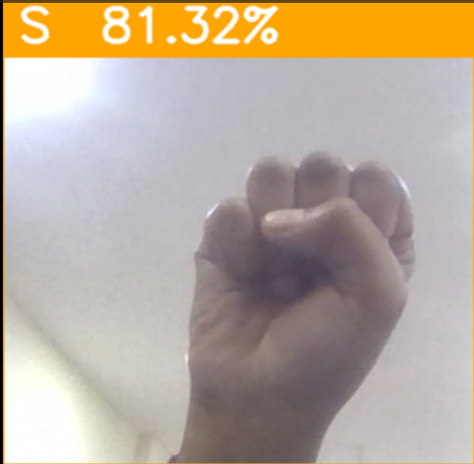
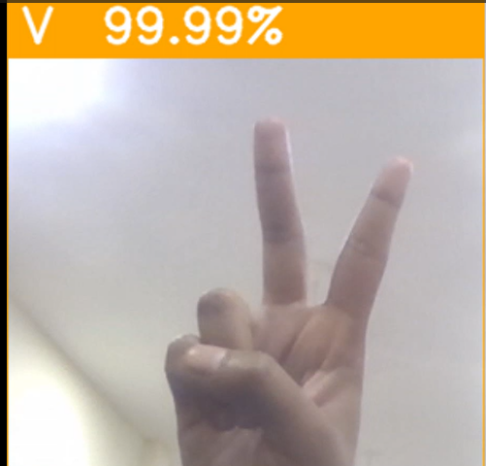


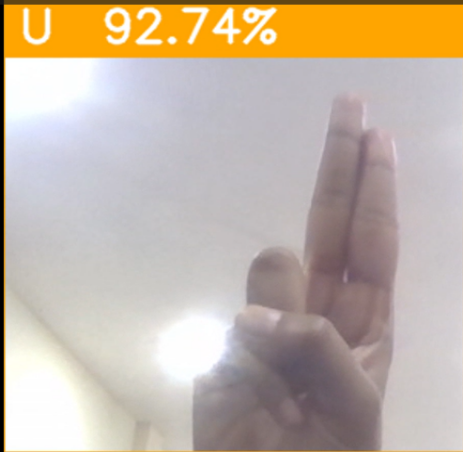
**2. RESNET-50**



|  |  |  |
| --- | --- | --- |
| **Aspect** | **CNN Model** | **ResNet-50 Model** |
| **Loss Trend** | Training loss reaches near zero, but validation loss fluctuates | Both losses decrease, with validation loss reducing faster |
| **Accuracy Trend** | Training accuracy reaches ~100%, validation accuracy is stable | Training accuracy is lower, validation accuracy fluctuates |
| **Overfitting** | Some overfitting due to validation loss instability | Less overfitting but high validation accuracy fluctuation |
| **Generalization** | Performs well but may require regularization | Generalizes better but struggles with stable accuracy |

**4.3.3 Real Time Hand Size Detection**

* *

* *

**4.4 Quality Assurance**

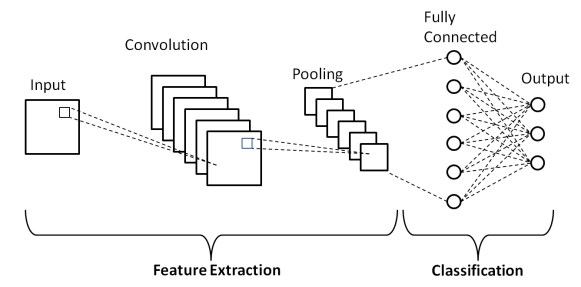
Quality Assurance (QA) is essential in evaluating the accuracy, efficiency, and reliability of real-time ASL hand sign recognition using CNN and ResNet50 models. In real-world scenarios, a systematic quality assurance strategy guarantees that the system provides accurate and consistent sign language detection. To ensure optimal performance across a variety of ASL gestures, the models are evaluated using accuracy, precision, recall, and F1-score. The custom CNN is contrasted with a pre-trained deep learning model called ResNet50 in order to examine trade-offs between classification accuracy, speed, and feature extraction capacity. Additionally, a variety of methods, such as confusion matrix analysis, unit testing, and integration testing, are used to verify the dataset quality, real-time processing speed, and model robustness. Finding the best model for real-time deployment is the aim in order to minimise misclassification and maximise usability in assistive apps

Chapter 5

Standards Adopted

**5.1 Design Standards**

The standards we adhered to during the project's design phase are the main topic of this section. This mostly relates to how we managed our data and created our CNN model for an ASL detection project. We aim to provide a comprehensive set of design standards, drawing inspiration from engineering disciplines like those that use IEEE and ISO standards.



Dataset Standard

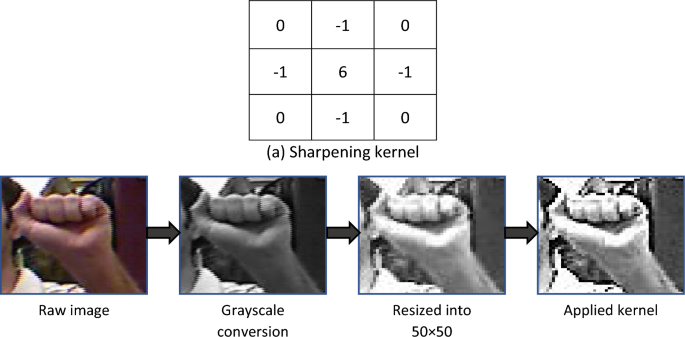
We utilized our personal dataset as our primary data source. This dataset comprises a collection of images representing the 26 letters of the American Sign Language alphabet and 10 numeric digits . The dataset is structured into 36 folders, one for each letter and digit.

Data Acquisition and Preparation:

* Personal dataset was created by capturing the photos .
* Each image in the dataset is a 200x200 pixel GREY image, depicting a single hand sign against a relatively uniform background.

Image Preprocessing:

* Resizing: No resizing was necessary as the images were already at the desired resolution of 200x200 pixels.
* Grayscale Conversion: The images were kept in GRAYSCALE format.



**5.2 Coding Standards**

Programming Language: The project was implemented using Python 3.9 .

Libraries: The following key libraries were used:

* TensorFlow 2.6: For building and training the CNN model.
* Keras: A high-level API for building neural networks.
* Matplotlib: For data visualization.
* NumPy: For numerical computations.
* OpenCV-contrib-python: For computer vision, image processing, and machine learning.
* Split-folders: For splitting dataset into 2 ratios ( training and validation) .

Code Style Guidelines : We adhered to the PEP 8 style guide for Python code.

* Write as few lines as possible.
* Use appropriate naming conventions .
* Segment blocks of code in the same section into paragraphs .
* Use indentation to mark the beginning and end of control structures. Clearly specify the code between them .
* Don’t use lengthy functions. Ideally, a single function should carry out a single task

**5.3 Testing Standards**

Data Splitting: The dataset was divided into training and validation with a ratio of 80:20 . This split ensures that the model is trained on a substantial portion of the data, validated on a separate set for hyperparameter tuning, and evaluated on unseen data to assess its generalization performance. We used stratified sampling to maintain the class distribution across the splits.

Evaluation Metrics: The model's performance was evaluated using the following metrics:

* Accuracy: To measure the overall correctness of the model's predictions.

accuracy = (TP + TN) / (TP + TN + FP + FN)

* Precision, Recall, and F1-score: To assess the model's performance for each individual sign class, particularly useful for identifying potential class imbalance issues. We calculated both micro and macro averages for these metrics.

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1-score = 2 \* (Precision \* Recall) / (Precision + Recall)

* Confusion Matrix: To provide a detailed breakdown of the model's predictions, showing which signs are most often confused with each other. We normalized the confusion matrix to show percentages.

Validation Strategy: We used k-fold cross-validation with k=5. This technique provides a more robust estimate of the model's performance by training and evaluating it on multiple subsets of the data. We ensured that the folds were stratified to maintain class balance in each fold.

Class Imbalance: The dataset was analyzed for class imbalance. If significant imbalance was detected, techniques such as stratified sampling during data splitting and class weighting during training were considered to mitigate its impact. We used the class weights parameter in the training process to give more weight to minority classes

Chapter 6

Conclusion and Future Scope

**6.1 Conclusion**

In this study, we explored a machine learning-based approach to recognizing American Sign Language (ASL). We highlighted the crucial role that sign language recognition systems play in improving communication for the deaf and hard-of-hearing communities. Our model demonstrated a high level of accuracy in identifying ASL gestures and proved capable of real-time recognition. However, incorporating more diverse data from different environments could further enhance its reliability and effectiveness. Currently, the system is limited to recognizing only static ASL alphabet gestures, but future improvements could expand its capabilities to include dynamic signs and more complex gestures.

**6.2 Future Scope**

To enhance communication even further, future research should focus on expanding the system’s ability to recognize dynamic sign language motions. Currently, the model has been specifically trained and tested on the ASL alphabet, allowing it to interpret individual letters. However, sign languages vary in vocabulary and structure across different countries and regions. To make the system more versatile and widely applicable, additional research and refinements are needed to evaluate its performance on other sign languages and ensure its adaptability for a broader audience.

**References**

[1] **MDPI**, "Sensors | Free Full-Text | Title of the Paper," Sensors, vol. 22, no. 3, p. 706, 2022. [Online]. Available: [https://doi.org/10.3390/s22030706](https://doi.org/10.3390/s22030706" \t "_new)

[2] **A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem**, "Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model," Applied Sciences, vol. 11, no. 9, p. 4164, 2021. [Online]. Available: [https://doi.org/10.3390/app11094164](https://doi.org/10.3390/app11094164" \t "_new)

[3] **R. K. Pathan, M. Biswas, S. Yasmin, M. U. Khandaker, M. Salman, and A. A. F. Youssef**, "RETRACTED ARTICLE: Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network," Scientific Reports, vol. 13, Art. no. 16975, 2023. [Online]. Available: [https://doi.org/10.1038/s41598-023-43852-x](https://doi.org/10.1038/s41598-023-43852-x" \t "_new)

[4] **D. Bhavana, K. K. Kumar, M. B. Chandra, P. V. S. K. Bhargav, D. J. Sanjana, and G. M. Gopi**, "Hand Sign Recognition using CNN," International Journal of Performability Engineering, vol. 17, no. 3, pp. 314-321, 2021. [Online]. Available: [https://doi.org/10.23940/ijpe.21.03.p7.314321](https://doi.org/10.23940/ijpe.21.03.p7.314321" \t "_new)

[5] **Sign Solutions**, "The importance of sign language," Sign Solutions, [Online]. Available: [https://www.signsolutions.uk.com/why-are-there-different-types-of-sign-language/](https://www.signsolutions.uk.com/why-are-there-different-types-of-sign-language/" \t "_new).

[6] **GeeksforGeeks**, "Convolutional Neural Network (CNN) in Machine Learning," GeeksforGeeks, [Online]. Available: [https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/](https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/" \t "_new). [Accessed: 10-Oct-2024].

[7] **H. Pradhan, S. Kumar, A. S. R. Tiwari, and A. Yadav**, "Development of a Sign Language Recognition System Using Machine Learning," ResearchGate, 2023. [Online]. Available: [https://www.researchgate.net/publication/373385057](https://www.researchgate.net/publication/373385057" \t "_new)

[8] **A. A. Alsabbagh and M. M. Al-Jarrah**, "Real-Time Sign Language Fingerspelling Recognition using Convolutional Neural Network," The International Arab Journal of Information Technology, vol. 19, no. 4, pp. 593–600, 2022. [Online]. Available: [https://iajit.org/upload/files/Real-Time-Sign-Language-Fingerspelling-Recognition-using-Convolutional-Neural-Network.pdf](https://iajit.org/upload/files/Real-Time-Sign-Language-Fingerspelling-Recognition-using-Convolutional-Neural-Network.pdf" \t "_new)

[9] **M. Solis**, "ReLU activation function: What is it & how does it work?" Built In, 25-Aug-2023. [Online]. Available: [https://builtin.com/machine-learning/relu-activation-function](https://builtin.com/machine-learning/relu-activation-function" \t "_new)

**Appendix**

The complete project, including the dataset, is available at the following link:

“ <https://github.com/Sparsshhh/Asl-Hand-Sign-Recognition> “