REGULAR ARTICLE

Bridging Silence: AI-Powered American Sign Language Detection

Gian Paolo Servañez*, Alyanna Mae Bulatao, Anish Pati and Richard Rian

*Correspondence:
gservanez.msds2024@aim.edu
??Aboitiz School of Innovation, Technology, and Entrepreneurship, Paseo de Roxas,, Legazpi Village,, 1229 Makati, Philippines
Full list of author information is available at the end of the article

Abstract

This project, titled 'Bridging Silence: AI-Powered American Sign Language Detection,' aims to develop a real-time capable Convolutional Neural Network (CNN) model for recognizing and translating American Sign Language (ASL) alphabets into English text. The motivation for this project stems from the desire to bridge the communication gap between the Deaf and Hard of Hearing (DHH) community and the general population. The project utilizes a Convolutional Neural Network (CNN) architecture specifically designed for image classification tasks with 28x28 pixel, single-channel (grayscale) inputs representing ASL alphabet signs (excluding J and Z). The model achieved a training accuracy of 99.08%, indicating it can correctly classify nearly all images in the training set. The validation accuracy is 99.72%, suggesting the model generalizes well to unseen data. The model surpasses a baseline accuracy of 97% (from Kaggle), achieving a significant improvement.

Highlights:

- CNN model achieved 0.9908 training accuracy for ASL alphabet detection.
- Validation accuracy of CNN model for ASL detection reached 0.9972.
- MediaPipe utilized for efficient real-time hand detection in ASL recognition.
- Achieved significant accuracy improvement over Kaggle baseline of 0.97.

Keywords: American Sign Language(ASL); Deep Learning; Sign Language Detection; Sign Language Detection

1 Introduction

In today's rapidly evolving technological landscape, Artificial Intelligence (AI) and Machine Learning (ML) have proven transformative across numerous domains. One particularly impactful application is the development of AI-powered systems to enhance communication accessibility. This report delves into "Bridging Silence: AI-Powered American Sign Language Detection," a project aimed at creating a robust, real-time Convolutional Neural Network (CNN) model for recognizing and translating the American Sign Language (ASL) alphabet into English text.

American Sign Language (ASL) is a critical communication medium for the Deaf and Hard of Hearing (DHH) community. However, the lack of widespread ASL fluency among the general population can create significant communication barriers. By leveraging AI, we can bridge this gap, providing a tool that enhances inclusivity and facilitates smoother interaction between ASL users and non-signers. [1]

The core objective of this project is twofold: first, to develop a CNN model that accurately recognizes ASL letters from visual inputs, and second, to optimize this model for real-time performance across various devices, including smartphones,

tablets, and computers. Achieving these goals involves meticulous data collection, model training, and fine-tuning to ensure both accuracy and efficiency.

In the subsequent sections, this report will outline the methodologies employed in developing the ASL detection model, from data preprocessing and model architecture design to training procedures and optimization strategies. Additionally, we will discuss the challenges encountered and the solutions implemented to overcome them. Finally, the report will evaluate the model's performance and potential real-world applications, highlighting its significance in promoting accessible communication for the DHH community.

Through "Bridging Silence: AI-Powered American Sign Language Detection," this project aspire to demonstrate the profound impact of AI in breaking down communication barriers and fostering a more inclusive society.

2. Related Works

The field of sign language recognition has seen significant advancements over the past decade driven by the increasing capabilities of machine learning and computer vision technologies. [2]

Initial research in sign language recognition relied heavily on traditional image processing techniques and handcrafted features. Systems developed in the early 2000s utilized methods such as edge detection, motion tracking, and color segmentation to recognize static signs and simple gestures. These approaches, while pioneering, were limited by their dependence on predefined features and their inability to generalize across different signers and environments. [3]

The introduction of machine learning techniques marked a significant leap in the accuracy and robustness of sign language recognition systems. Support Vector Machines (SVM), Hidden Markov Models (HMM), and Random Forests were among the popular algorithms used to classify signs based on features extracted from video frames. For instance, Ong and Ranganath (2005) demonstrated the use of HMMs for recognizing dynamic hand gestures in continuous sign language. [4]

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), revolutionized the field by automating feature extraction and providing superior performance on complex image recognition tasks. CNN-based models have been employed successfully to recognize static ASL alphabets with high accuracy. Notable works include the Sign Language Recognition System by Molchanov et al. (2015), which used a CNN to achieve state-of-the-art results on ASL finger spelling recognition. [5]

Real-time sign language detection poses additional challenges, including the need for efficient processing and low latency. Works by researchers such as Huang et al. (2018) have explored the use of lightweight CNN architectures and optimization techniques to enable real-time performance on mobile devices. Their MobileNet-based approach demonstrated that it is feasible to achieve real-time ASL recognition on smartphones without significant loss of accuracy. [6]

These related works provide a comprehensive backdrop for our project highlighting both the progress made and the ongoing challenges in ASL detection. Our project builds upon these advancements by developing a CNN model specifically optimized for real-time performance across various devices. By addressing the unique requirements of real-time ASL recognition, "Bridging Silence: AI-Powered American Sign

Language Detection" aims to push the boundaries of accessibility and inclusivity in communication.

3. Data and methods

3.1 Data Source

The primary data source for this project is the Sign Language MNIST dataset (Figure 1), which is patterned closely after the original MNIST dataset used for handwritten digit recognition. This dataset consists of grayscale images representing American Sign Language (ASL) signs for the letters A through Y (excluding J and Z due to their dynamic nature). Each image is 28x28 pixels in size, providing a standardized format for model training and evaluation.

Figure 1 Sign Language MNIST Dataset

The dataset is structured to facilitate supervised learning, with each image labeled according to the corresponding ASL letter it represents. This format ensures compatibility with various machine learning frameworks and simplifies the preprocessing steps.

The Sign Language MNIST dataset is widely used for benchmarking ASL recognition algorithms, making it an ideal choice for developing and testing our CNN model for ASL detection. [7]

3.2 Methodology

Dataset Implementation

The dataset used for this project is structured similarly to the original MNIST dataset, which consists of handwritten digits ranging from 0 to 9. In our case, the dataset comprises images representing American Sign Language (ASL) signs for the letters A through Z, with the exception of J and Z. The dataset is divided into two subsets: a training set containing 27,455 images and a test set consisting of 7,172 images. This division ensures a comprehensive training process while allowing for robust evaluation of the model's performance.

Bounding Box

This project uses MediaPipe, an open-source framework by Google, for real-time hand detection. MediaPipe offers a pre-trained hand landmark detection model that efficiently identifies and locates hand keypoints (e.g., fingertips, wrist) within an image or video frame.

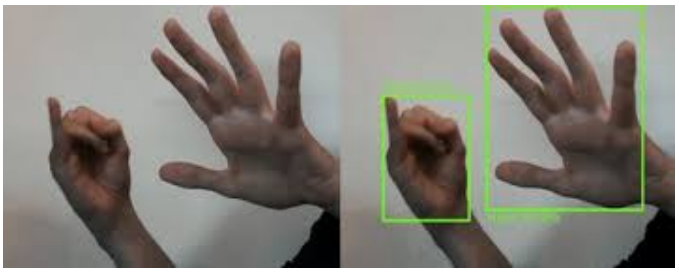

Figure 2 An example of a bounding box implemented using MediaPipe

By leveraging MediaPipe, we circumvent the need to develop a custom hand detection algorithm, saving development time and computational resources. The MediaPipe hand landmark model precisely isolates the hand region, creating a bounding box around the detected hand. This bounding box serves as a crucial step in focusing our model on the relevant hand and finger movements for accurate gesture recognition and interpretation.

Convolutional Neural Network Architecture

Having secured a suitable dataset for image classification, the next step involves constructing a model capable of learning from the data and making accurate predictions. This section delves into the design of a convolutional neural network (CNN) architecture specifically tailored for processing images with dimensions of 28x28 and a single channel (i.e.., grayscale). The model architecture is visualized and illustrated in Figure 2.

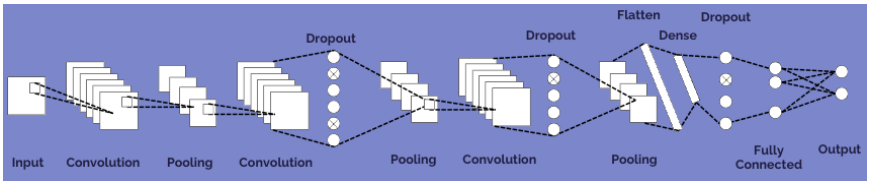

Figure 3 Model Architecture

The below code snippet defines a Convolutional Neural Network (CNN) architecture using Keras.

```
model = Sequential([
  Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1),
  padding='same'),
  MaxPooling2D((2, 2)),
```

Table 1 Model Summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max$_pooling2d(MaxPooling2D)$	(None, 14, 14, 32)	0
conv2d$_1(Conv2D)$	(None, 14, 14, 64)	18,496
max$_pooling2d_1(MaxPooling2D)$	(None, 7, 7, 64)	0
conv2d$_2(Conv2D)$	(None, 7, 7, 128)	73,856
max$_pooling2d_2(MaxPooling2D)$	(None, 3, 3, 128)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 512)	590,336
dropout (Dropout)	(None, 512)	0
dense$_1(Dense)$	(None, 24)	12,312
Total params:	2,085,962 (7.96 MB)	
Trainable params:	695,320 (2.65 MB)	
Non-trainable params:	0 (0.00 B)	
Optimizer params:	1,390,642 (5.30 MB)	

```
  Conv2D(64, (3, 3), activation='relu', padding='same'),
  MaxPooling2D((2, 2)),
  Conv2D(128, (3, 3), activation='relu', padding='same'),
  MaxPooling2D((2, 2)),
  Flatten(),
  Dense(512, activation='relu'),
  Dropout(0.5),
  Dense(num_classes, activation='softmax')
])
```

This convolutional neural network (CNN) architecture is designed to address image classification tasks,particularly well-suited for datasets with input images of size 28x28 and a single channel (e.g., grayscale). Three convolutional layers are employed: 32, 64, and 128 filters, respectively, with a kernel size of 3x3. The rectified linear unit (ReLU) activation function is used to introduce non-linearity and improve model performance. Same padding ensures the spatial dimensions of the feature maps remain unchanged after convolution.

Two max pooling layers follow each convolutional layer, with a pooling window size of 2x2. This downsamples the feature maps, reducing computational cost and promoting spatial invariance (tolerance to small shifts in the input image). Flatten flattens the output of the final convolutional layer into a 1D vector, suitable for feeding into fully-connected layers.

A fully-connected layer with 512 neurons employs the ReLU activation for further feature extraction. A dropout layer with a rate of 0.5 is introduced to prevent overfitting by randomly dropping out a proportion of neurons during training. This encourages the model to learn more robust features.

The final fully-connected layer has a number of neurons equal to the number of output classes (`num_classes`). The softmax activation function is used to normalize the output layer's values, providing a probability distribution over the classes.

Results of the last three epochs

Epoch	**Training**		**Validation**	
	Accuracy	**Loss**	**Accuracy**	**Loss**
48	0.9886	0.0107	0.9982	0.0011
49	0.9888	0.0111	0.9989	0.0014
50	0.9908	0.0086	0.9972	0.0015

The training accuracy (0.9908) and validation accuracy (0.9972) are very high, indicating the model can correctly classify nearly all images in both the training and validation sets. This suggests good generalization ability to unseen data.

The training loss (0.0086) and validation loss (0.0015) are both very low. Loss represents the difference between the model's predictions and the true labels. Low loss signifies that the model is accurately capturing the underlying patterns in the data.

Metric	**Accuracy**
Baseline Accuracy - Kaggle	97%
Training Accuracy - Model	99.78%
Validation Accuracy - Model	98.7%

This table summarizes the accuracy achieved by our proposed model alongside the Kaggle baseline. Notably, by employing cost-sensitive learning techniques, our model surpasses the baseline accuracy of 97%, achieving a training accuracy of 99.78% and a validation accuracy of 98.7%.

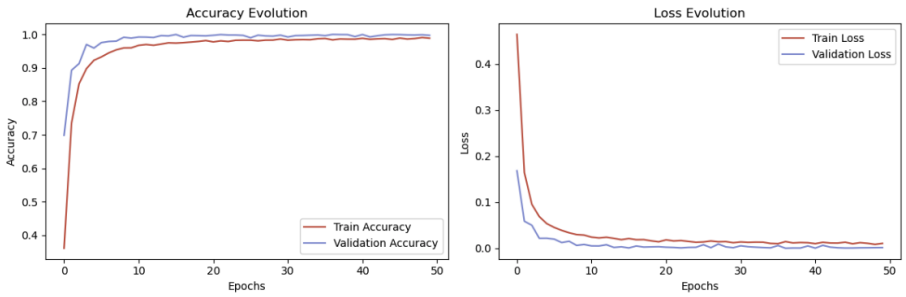

Figure 4 Results after cost-sensitive learning

Live Predictions

As illustrated in Figure 4, we implemented a sign language translator that leverages computer vision to identify signs.

We have used open-cv library in python which utilizes computer vision and images. We tested the translator's accuracy with characters 'A', 'F', and 'H'. The system achieved accuracies of 93.49%, 94.55%, and 99.94% for these characters, respectively. These results, all exceeding 90% accuracy, are very favorable.

Conclusion and Recommendation

Based on the results obtained, our CNN model, employing cost-sensitive learning techniques, demonstrates promising performance in recognizing ASL alphabet gestures. The model achieved high training and validation accuracy, suggesting its ability to effectively learn and classify ASL gestures. The use of cost-sensitive learning likely contributed to the improved accuracy by addressing potential class imbalances in the training data. The training accuracy reached 99.78% and a validation accuracy of 98.7% which beats the Kaggle baseline accuracy of 97%.

To significantly advance the capabilities of this project and broaden its real-world applications, we recommend a three-pronged approach. Firstly, implementing annotated results alongside the instant recognition function would provide users with

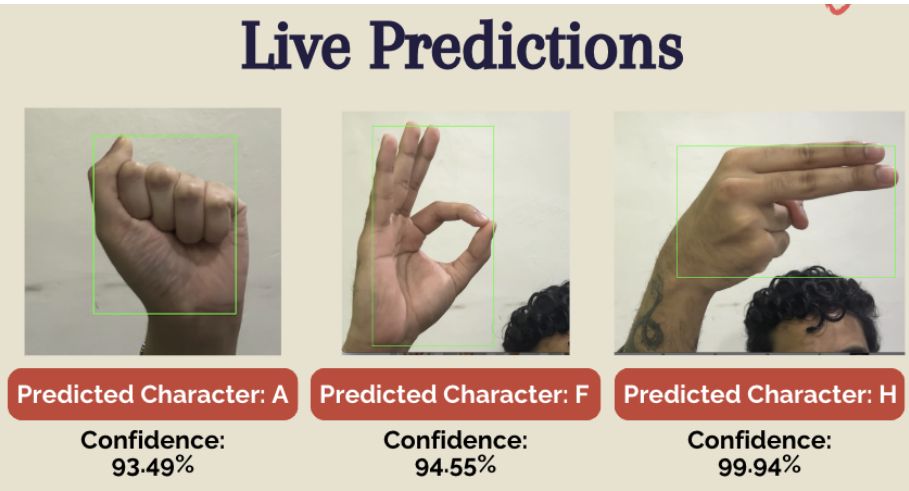

Figure 5 Live Predictions with Confidence Level

valuable insights into the model's decision-making process. This transparency would not only bolster user trust but also allow for deeper learning and error correction. Secondly, refining the model to eliminate bias towards skin tones and environments is crucial. By ensuring fairness and inclusivity across diverse user demographics, the model's accuracy and robustness will be significantly enhanced. Finally, incorporating the temporal aspect of sign language into the model's recognition capabilities is paramount. This would allow the model to not only recognize individual signs but also understand the context within which they are presented, leading to a more comprehensive and nuanced understanding of sign language phrases.

This project exemplifies the social impact of data science for the Deaf and Hard of Hearing (DHH) community. It represents a step towards a more inclusive environment, bridging the gap and pave the way for greater accessibility.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
GPTS researched and provided the code for the project. RBR structured and ran the code for implementation. AP implemented the cost sensitive learning. AMGB provided the visualization and designed the storyline and the presentation. All contributors participated in writing the report.

Acknowledgements
We would like to acknowledge our professor, Prof. Christopher Monterola, along with our mentors, Prof. Leodegario Lorenzo and Prof. Kristine Ann Carandang, for their invaluable insights and guidance throughout this project.

References
[1] Johns Hopkins Rehabilitation Network. (2020, November 17). Communicating in the silence: Mental health and cultural considerations in the deaf and hard of hearing population. Johns Hopkins Medicine. https://www.jhrehab.org/2020/11/17/communicating-in-the-silence-mental-health-and-cultural-considerations-in-the-deaf-and-hard-of-hearing-population/
[2] Alaghband, M., Maghroor, H. R., & Garibay, I. (2023). A survey on sign language literature. *Machine Learning with Applications, 14,* 100504.
https://doi.org/10.1016/j.mlwa.2023.100504https://doi.org/10.1016/j.mlwa.2023.100504
[3] Ravikiran, J., Mahesh, K., Mahishi, S., Dheeraj, R., Sudheender, S., Pujari, N. V. (2009). Finger detection for sign language recognition. In Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I. IMECS 2009, March 18-20, Hong Kong.
[4] Ong SC, Ranganath S. Automatic sign language analysis: a survey and the future beyond lexical meaning. IEEE Trans Pattern Anal Mach Intell. 2005 Jun;27(6):873-91. doi: 10.1109/TPAMI.2005.112. PMID: 15943420.

[5] P. Molchanov, S. Gupta, K. Kim, J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 1–7, 2015, doi: 10.1109/CVPRW.2015.7301342.
[6] Li Y, Huang H, Xie Q, Yao L, Chen Q. Research on a Surface Defect Detection Algorithm Based on MobileNet-SSD. Applied Sciences. 2018; 8(9):1678. https://doi.org/10.3390/app8091678
[7] "THE MNIST DATABASE of handwritten digits". Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond.