CSCI 6364 - MACHINE LEARNING FINAL PROJECT REPORT

ASL RECOGNITION

Team Trifecta:
Anbu Ezhilmathi Nambi - G33350186
Aparna Shankar - G49628466
Kishan Ramesh - G44387483

TABLE OF CONTENTS

1. Introduction

In today's society, effective communication is vital in fostering inclusivity and accessibility. However, traditional spoken language can present challenges for individuals with hearing or speech impairments. American Sign Language (ASL) offers a comprehensive alternative, utilizing hand signs, facial expressions, and body postures. ASL serves as the primary mode of communication for millions of deaf and hard-of-hearing individuals globally. Yet, barriers between ASL users and those unfamiliar with the language often impede accessibility and inclusivity across different contexts.

2. Motivation

The motivation behind this project stems from the pressing need to address the challenges faced by American Sign Language users due to limited availability and high costs associated with human interpreters. Currently, ASL users encounter significant barriers to real-time communication in various settings, including educational, professional, and social environments further impeding their ability to participate and engage fully with others. The progress in ML and CV technologies holds great promise for addressing ASL recognition and translation challenges, effectively breaking down any communication barriers. This project seeks to leverage ML to develop solutions for ASL recognition and translation.

3. Problem Statement

To develop an ASL recognition system using machine learning (ML) techniques that address communication breakdowns between ASL users and those unfamiliar with sign language, thus reducing frustration and isolation.

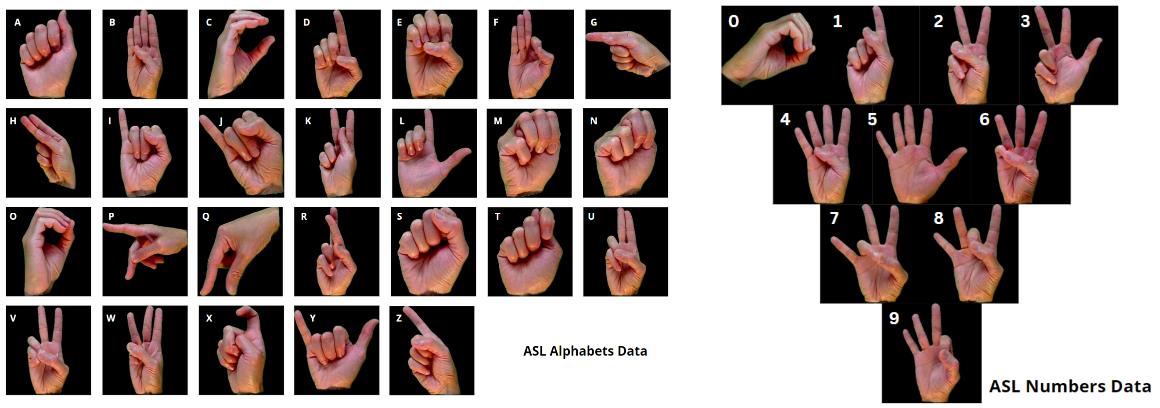
4. Objective

- The main objective of this project includes developing Machine Learning models that can accurately recognize and interpret ASL gestures.
- Specific objectives include achieving high accuracy, and real-time performance to improve accessibility, inclusivity, and communication for ASL users across diverse settings

.

5. About the dataset

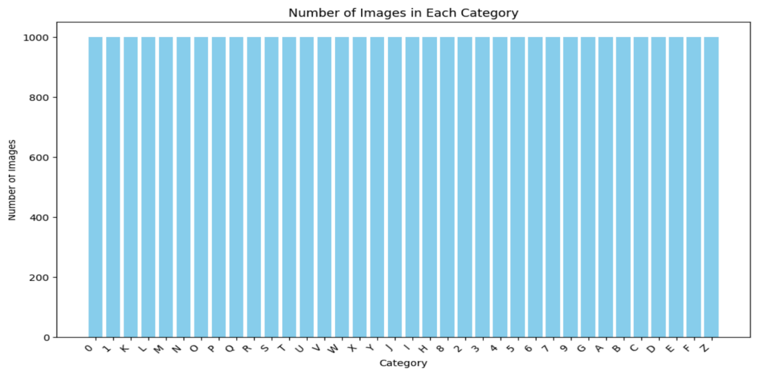
The dataset, compiled from Kaggle, contains image data depicting American Sign Language (ASL) gestures representing both the digits 0-9 and the letters A-Z. Each category encompasses approximately 1500 to 3000 images, resulting in a merged dataset with a rich and diverse collection of data for training and testing the ASL recognition model.



Sample images from the dataset

6. Understanding the data

The dataset initially comprised more than 50,000 images categorized into 43 classes. However, a few irrelevant classes excluding the alphabet and numbers were removed, resulting in a dataset with 36 classes. Following preprocessing, each class is resampled to include 1000 images, aiding in the model's training. The following graph shows the distribution of images after pre-processing them.



Class distribution of images

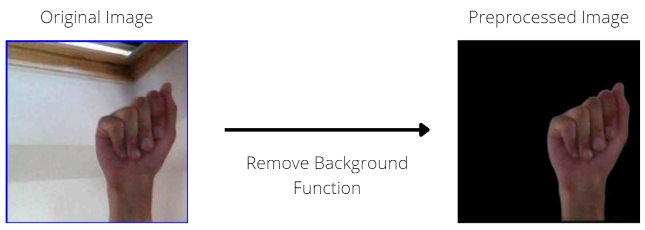
7. Data Pre-processing

7.1. Background Removal Function:

The background removal function is designed to automate the process of eliminating backgrounds from images, facilitating batch image processing efficiently. This aids in better segmenting the hand from the rest of the image, to interpret the hand gestures, as it provides a clear boundary between the hand and the background.

- The function works by systematically scanning a specified input directory to identify image files in JPEG, JPG, and PNG formats.
- Upon detecting these files, it employs a custom 'remove' function that meticulously extracts the backgrounds from each image.
- The processed images, now with backgrounds removed, are then saved to a designated output path.

This streamlined approach is particularly beneficial for tasks that require the removal of backgrounds on a large scale, optimizing both time and resource utilization in various applications.



Background removal of image

7.2. Data Augmentation:

The dataset has been strategically partitioned into training and validation sets to facilitate effective model training while enabling performance evaluation on a separate set, thus minimizing the risk of overfitting and enhancing the model's ability to generalize to new data. To further augment the dataset and reduce the likelihood of overfitting, data augmentation techniques are employed, creating various transformed versions of the original images. These transformations include:

- Rescaling pixel values from the range [0, 255] to [0, 1] to accelerate model convergence
- Applying random rotations up to 20 degrees to accommodate different image orientations
- Flipping images horizontally to ensure the model's robustness to orientation changes,
- Implementing a shear range of 0.2 to introduce slanting transformations that mimic different viewing angles.

These preprocessing steps are critical in preparing the dataset for more effective training and ultimately achieving a more generalized and robust model.

8. Model Implementation

8.1. VGG 16:

The model leverages a pre-trained VGG16 architecture, sourced from ImageNet, with its top layer omitted to serve primarily for feature extraction. To ensure that the intrinsic properties of VGG16 are maintained, the weights of the existing network are frozen, allowing the training process to concentrate on the layers that have been newly added. These enhancements include a SpatialDropout2D layer for regularization, which helps prevent overfitting. This is followed by a Flatten layer and two Dense layers, employing ReLU and softmax activation functions respectively, to facilitate the classification process. The construction of the final model is accomplished using Keras' Model class, which specifies both the input and output layers, thereby completing the setup for effective deep learning training tailored to specific classification tasks.

8.2. ResNet50:

The implementation utilizes the ResNet50 architecture, which is renowned for its deep network of convolutional, identity, and pooling layers that effectively extract features. The structure of ResNet50 is designed to alternate between convolutional blocks, which serve the purpose of down-sampling, and multiple identity blocks that transform features while maintaining the same dimensionality.

In detail, the architecture comprises one convolutional block followed by two identity blocks in Stage 2, one convolutional block and three identity blocks in Stage 3, one convolutional block paired with five identity blocks in Stage 4, and finally, one convolutional block and two identity blocks in Stage 5. To prepare the feature maps for the classification process, the architecture incorporates an average pooling layer followed by a flatten layer, ensuring that the spatial features are condensed into a format suitable for classification.

8.3. Custom CNN:

The sequential model is constructed with three convolutional layers, which are configured with decreasing kernel sizes of 5x5, 3x3, and 3x3, and increasing filter counts of 32, 64, and 64, respectively. This setup enhances the model's ability to capture detailed image features. Following each convolutional layer, a ReLU activation function introduces non-linearity, and a 2x2 max pooling operation is employed to reduce the spatial dimensions of the feature maps. To combat overfitting, a SpatialDropout2D layer with a dropout rate of 0.4 is applied after the pooling steps, effectively deactivating approximately 40% of the neurons at a given time. The model transitions from multidimensional feature maps to a 1D vector through a flatten layer, which feeds into a dense layer comprising 128 neurons activated by ReLU. The architecture culminates with a softmax layer that contains 36 neurons, tailored for the task of classification.

8.4. Voting Classifier of 3 Models:

The ensemble model employs a voting classifier mechanism that integrates predictions from three distinct architectures—Custom CNN, ResNet, and VGG16—to improve accuracy. This is achieved through an average voting approach, where the predicted probabilities for each class from each model are averaged. The class receiving the highest mean probability from the combined predictions of all models is chosen as the final output. This method of average voting leverages the unique strengths while offsetting the weaknesses of each model, thereby enhancing the reliability and robustness of the ensemble's overall predictions.

9. Generalization Techniques

9.1. Batch Normalization:

Implementing batch normalization in the project played a pivotal role in enhancing the performance and stability of the model. It standardizes activations within each mini-batch to adjust the batch mean and standard deviation, ensuring uniformity across the network's activations. The function incorporates scale (γ) and shift (β) parameters post-normalization, dynamically optimizing the distribution of activations throughout the training process. It also introduces a mild regularization effect, reducing overfitting risk by injecting noise. Additionally, batch normalization decreased the model's sensitivity to initial weight configurations, ensuring consistent training outcomes and enhancing model performance.

9.2. Spatial Dropouts:

Adaptation of dropout for CNNs deactivates entire feature maps instead of individual neurons, randomly zeroing out channels with a consistent percentage dropped across layers. Spatial dropouts limit their dependency on specific feature sets, encouraging robust and diverse feature representations, and enhances model resilience to input variations and noise. It significantly improves CNN generalization, preventing overfitting and promoting comprehensive use of learned features, especially beneficial for spatial data handling, as it is tailored for spatial data like images and videos. Hence, addition of dropout techniques played a pivotal role in achieving higher model accuracy and performance, enhancing the model's ability to generalize, in the ASL predictions.

10. Techniques to avoid Overfitting

Callbacks are instrumental in automating specific tasks at each training epoch, providing essential controls over the training process. In the implemented models, two key callbacks are employed:

CALLBACKS	FUNCTIONALITY	OVERFITTING MITIGATION
EarlyStopping	Monitors a designated metric, such as 'val_loss', during training. If there is no improvement in the monitored metric for a specified number of epochs, the training process is halted.	Particularly effective in mitigating overfitting as it can revert to the best-performing model weights on the validation set when the 'restore_best_weights' option is enabled. It ensures that the model does not deviate from the most effective solution during training.
ReduceLROn Plateau	Played a crucial role in fine-tuning the learning process by adjusting the learning rate when progress in a specified metric, like 'val_accuracy', stalls, determined by the patience parameter.	By reducing the learning rate, this callback helps prevent overfitting and facilitates more effective model convergence

11. Evaluation and Performance Metrics

Following the implementation of the model, it is imperative to assess its performance using metrics such as Accuracy, Loss, Precision, Recall, and F1 score. Subsequently, in the subsequent chapter, the performances of all three models were evaluated, and their outcomes are presented.

11.1. VGG 16

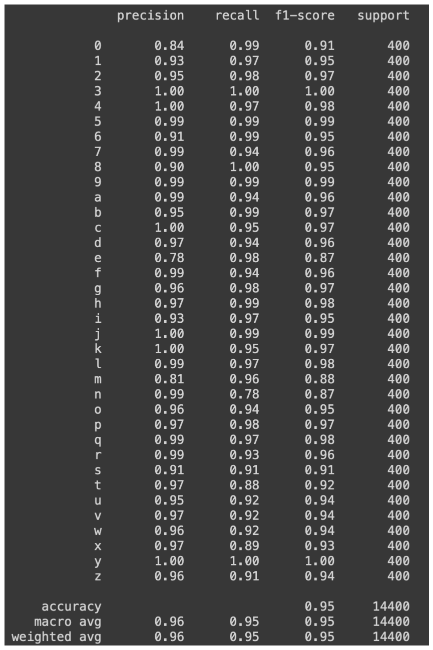
Data	Accuracy	Loss
Training data	99.15%	0.0320
Validation data	98.5%	0.0594
Test data	95.47%	0.1921

11.1.1. Train Accuracy and Loss: The VGG16 model demonstrates strong performance on the training dataset, achieving a training accuracy of 99.15% and exhibiting minimal training loss of 0.0320.

11.1.2. Validation Accuracy and Loss: It also excels on validation data, achieving an impressive validation accuracy of 98.5% and demonstrating a validation loss of 0.0594.

11.1.3. Test Accuracy and Loss: The VGG16 model also exhibits robust performance on the test dataset, achieving a testing accuracy of 95.47% and demonstrates a testing loss of 0.1921.

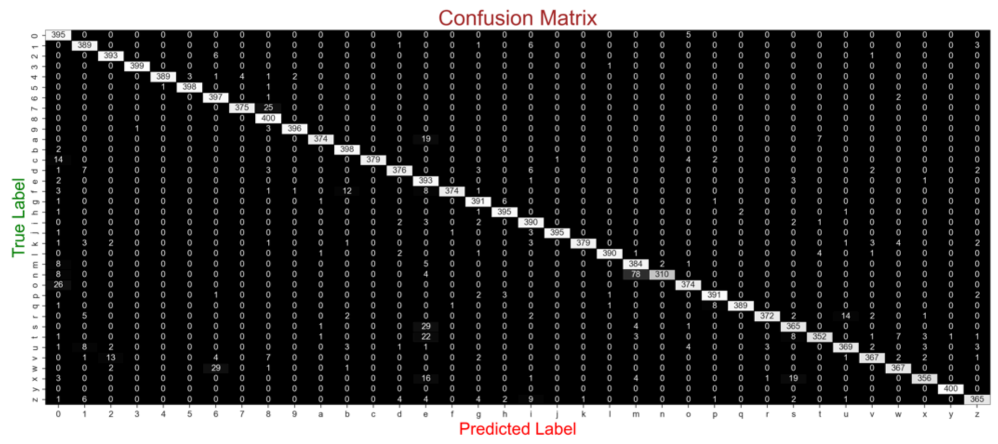
11.1.4. Classification Report



```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91       400
           1       0.93      0.97      0.95       400
           2       0.95      0.98      0.97       400
           3       1.00      1.00      1.00       400
           4       1.00      0.97      0.98       400
           5       0.99      0.99      0.99       400
           6       0.91      0.99      0.95       400
           7       0.99      0.94      0.96       400
           8       0.90      1.00      0.95       400
           9       0.99      0.99      0.99       400
           a       0.99      0.94      0.96       400
           b       0.95      0.99      0.97       400
           c       1.00      0.95      0.97       400
           d       0.97      0.94      0.96       400
           e       0.78      0.98      0.87       400
           f       0.99      0.94      0.96       400
           g       0.96      0.98      0.97       400
           h       0.97      0.99      0.98       400
           i       0.93      0.97      0.95       400
           j       1.00      0.99      0.99       400
           k       1.00      0.95      0.97       400
           l       0.99      0.97      0.98       400
           m       0.81      0.96      0.88       400
           n       0.99      0.78      0.87       400
           o       0.96      0.94      0.95       400
           p       0.97      0.98      0.97       400
           q       0.99      0.97      0.98       400
           r       0.99      0.93      0.96       400
           s       0.91      0.91      0.91       400
           t       0.97      0.88      0.92       400
           u       0.95      0.92      0.94       400
           v       0.97      0.92      0.94       400
           w       0.96      0.92      0.94       400
           x       0.97      0.89      0.93       400
           y       1.00      1.00      1.00       400
           z       0.96      0.91      0.94       400

    accuracy                           0.95     14400
   macro avg       0.96      0.95      0.95     14400
weighted avg       0.96      0.95      0.95     14400
```

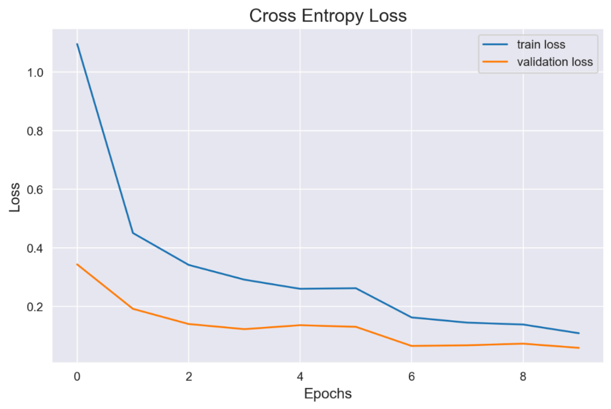
Upon examination of the classification results, it becomes evident that the model has attained elevated levels of precision, recall, and F1-score across most classes, indicating a high level of performance on this dataset. Consequently, the model exhibits the capability to accurately classify a significant portion of images while also minimizing the number of relevant images that are not detected.

11.1.5. Confusion Matrix



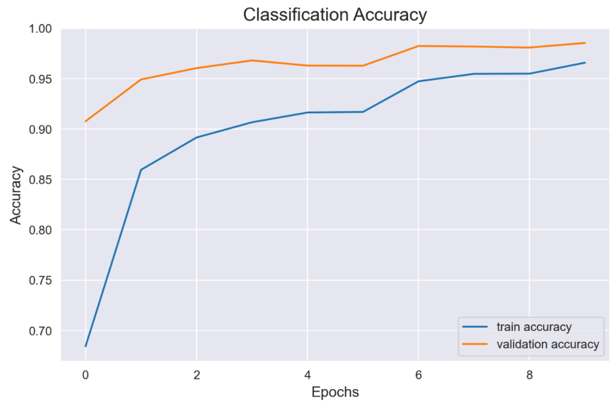
The confusion matrix suggests that the model has a good performance on this dataset, with high accuracy and a lower false negative rate than false positive rate.

11.1.6. Cross Entropy Loss



The cross-entropy loss of train and validation data is decreasing over time. This suggests that the model is learning to better classify the data.

11.1.7. Classification Accuracy



An upward trend in the training and validation accuracy suggests that the model's predictive accuracy is improving as training progresses.

11.2. ResNet50

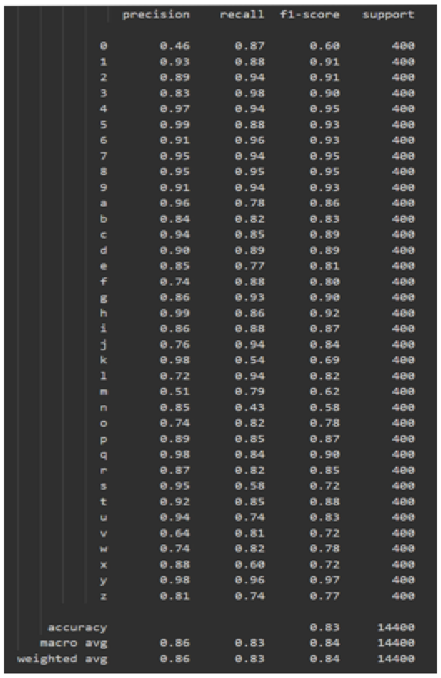
Data	Accuracy	Loss
Training data	97.88%	0.0679
Validation data	96.94.%	0.1036
Test data	85.69%	0.6285

11.2.1. Train Accuracy and Loss: The ResNet50 model achieves a train accuracy of 97.88% and train loss of 0.0679.

11.2.2. Validation Accuracy and Loss: It has a validation accuracy of 96.94% and validation loss of 0.1036.

11.2.3. Test Accuracy and Loss: The model performs well on test data with accuracy of 85.69% and loss of 0.6285.

11.2.4. Classification Report

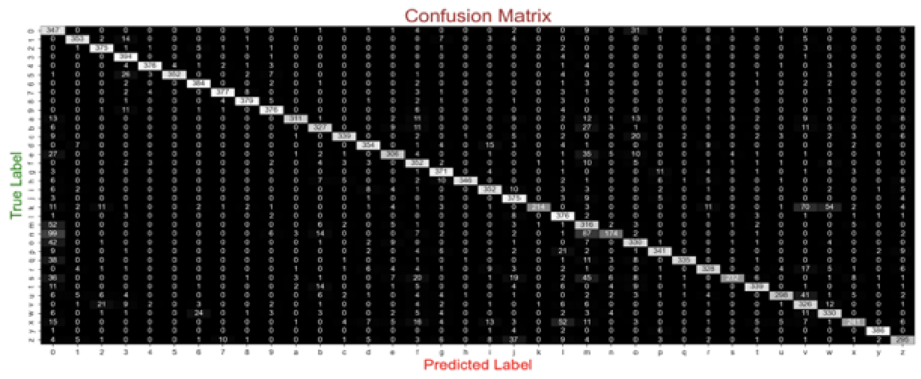


```
              precision    recall  f1-score   support

           0       0.46      0.87      0.60       400
           1       0.93      0.88      0.91       400
           2       0.89      0.94      0.91       400
           3       0.83      0.98      0.90       400
           4       0.97      0.94      0.95       400
           5       0.99      0.88      0.93       400
           6       0.91      0.96      0.93       400
           7       0.95      0.94      0.95       400
           8       0.95      0.95      0.95       400
           9       0.91      0.94      0.93       400
           a       0.96      0.78      0.86       400
           b       0.84      0.82      0.83       400
           c       0.94      0.85      0.89       400
           d       0.90      0.89      0.89       400
           e       0.85      0.77      0.81       400
           f       0.74      0.88      0.80       400
           g       0.86      0.93      0.90       400
           h       0.99      0.86      0.92       400
           i       0.86      0.88      0.87       400
           j       0.76      0.94      0.84       400
           k       0.98      0.54      0.69       400
           l       0.72      0.94      0.82       400
           m       0.51      0.79      0.62       400
           n       0.85      0.43      0.58       400
           o       0.74      0.82      0.78       400
           p       0.89      0.85      0.87       400
           q       0.98      0.84      0.90       400
           r       0.87      0.82      0.85       400
           s       0.95      0.58      0.72       400
           t       0.92      0.85      0.88       400
           u       0.94      0.74      0.83       400
           v       0.64      0.81      0.72       400
           w       0.74      0.82      0.78       400
           x       0.88      0.60      0.72       400
           y       0.98      0.96      0.97       400
           z       0.81      0.74      0.77       400

    accuracy                           0.83     14400
   macro avg       0.86      0.83      0.84     14400
weighted avg       0.86      0.83      0.84     14400
```

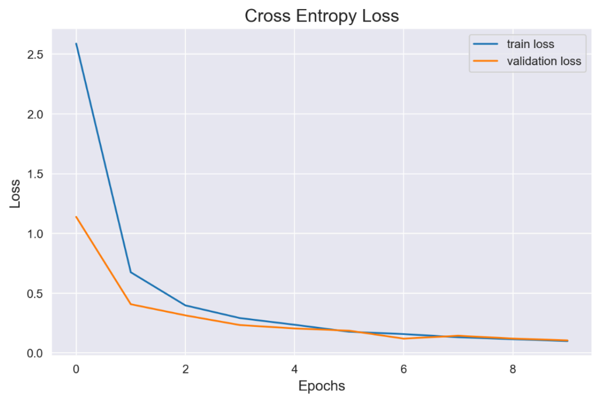
The classification report indicates that the model is effective for most classes with elevated F-1 scores, but struggles with some classes like 0,m, and n with lower F-1 scores and could benefit from further improvement in classifying these classes.

11.2.5. Confusion Matrix

The confusion matrix suggests that the model has a good performance on this dataset, with high accuracy and a lower false negative rate than false positive rate.

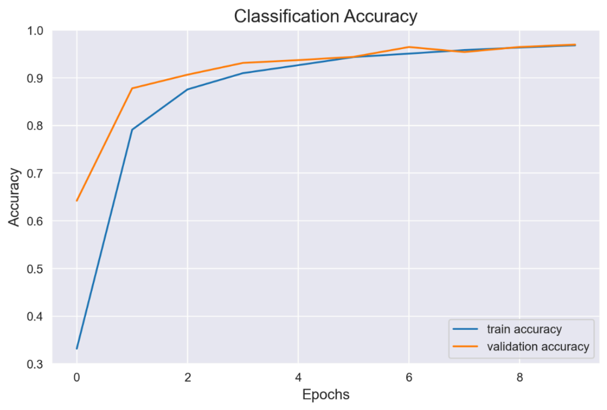


11.2.6. Cross Entropy Loss



The cross-entropy loss of train and validation data is decreasing over time. This suggests that the model is learning better in classifying the data.

11.2.7. Classification Accuracy



The training and validation accuracy increase with epochs. As the curves converge, it suggests that the model is not overfitting

11.3. Custom CNN

Data	Accuracy	Loss
Training data	92.55%	0.2384

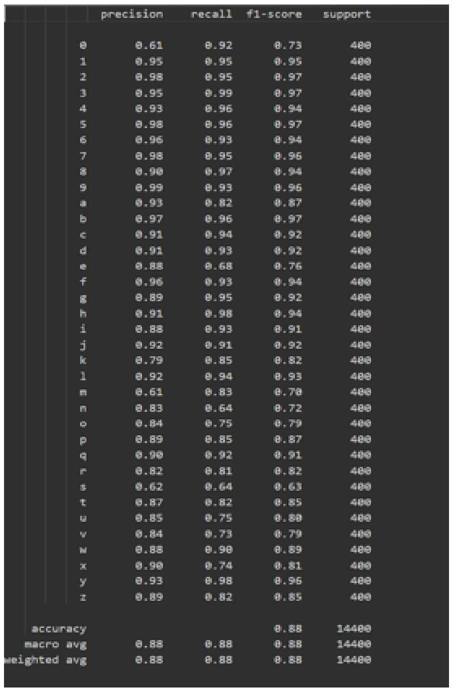
Validation data	89.61%	0.3419
Test data	87.51%	0.5620

11.3.1 Train Accuracy and Loss: The Custom CNN model has an overall training accuracy of 92.55% and loss of 0.2384.

11.3.2. Validation Accuracy and Loss: It secures a validation accuracy of 89.61% and loss of 0.3419.

11.3.3. Test Accuracy and Loss: The model performs well on test data with accuracy of 87.51% and loss of 0.5620.

11.3.4. Classification Report

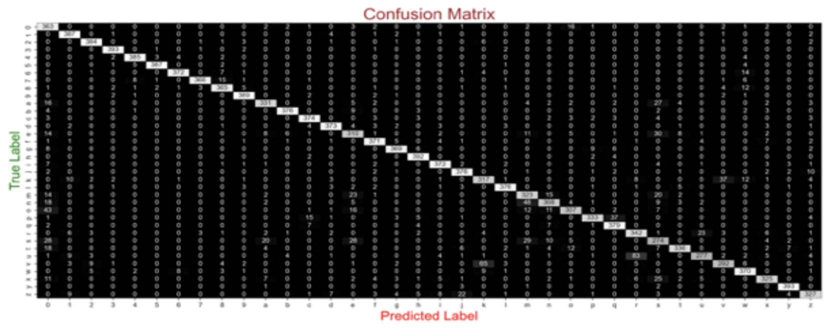


```
              precision    recall  f1-score   support

           0       0.61      0.92      0.73       400
           1       0.95      0.95      0.95       400
           2       0.98      0.95      0.97       400
           3       0.95      0.99      0.97       400
           4       0.93      0.96      0.94       400
           5       0.98      0.96      0.97       400
           6       0.96      0.93      0.94       400
           7       0.98      0.95      0.96       400
           8       0.90      0.97      0.94       400
           9       0.99      0.93      0.96       400
           a       0.93      0.82      0.87       400
           b       0.97      0.96      0.97       400
           c       0.91      0.94      0.92       400
           d       0.91      0.93      0.92       400
           e       0.88      0.68      0.76       400
           f       0.96      0.93      0.94       400
           g       0.89      0.95      0.92       400
           h       0.91      0.98      0.94       400
           i       0.88      0.93      0.91       400
           j       0.92      0.91      0.92       400
           k       0.79      0.85      0.82       400
           l       0.92      0.94      0.93       400
           m       0.61      0.83      0.70       400
           n       0.83      0.64      0.72       400
           o       0.84      0.75      0.79       400
           p       0.89      0.85      0.87       400
           q       0.90      0.92      0.91       400
           r       0.82      0.81      0.82       400
           s       0.62      0.64      0.63       400
           t       0.87      0.82      0.85       400
           u       0.85      0.75      0.80       400
           v       0.84      0.73      0.79       400
           w       0.88      0.90      0.89       400
           x       0.90      0.74      0.81       400
           y       0.93      0.98      0.96       400
           z       0.89      0.82      0.85       400

    accuracy                           0.88     14400
   macro avg       0.88      0.88      0.88     14400
weighted avg       0.88      0.88      0.88     14400
```

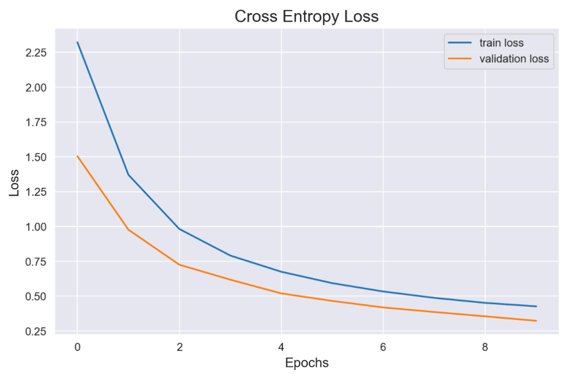
The classification report indicates that the model performs effectively for majority of classes, except classes like m and s with low F-1 scores and could benefit from further improvement in classifying certain classes.

11.3.5. Confusion Matrix

The confusion matrix suggests that the model has a good performance on this dataset, with high accuracy and a lower false negative rate than false positive rate.

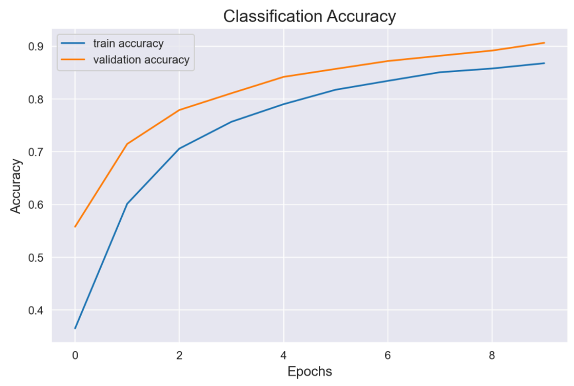


11.3.6. Cross Entropy Loss



The cross-entropy loss of train and validation data is decreasing over time. This suggests that the model is learning to better classify the data.

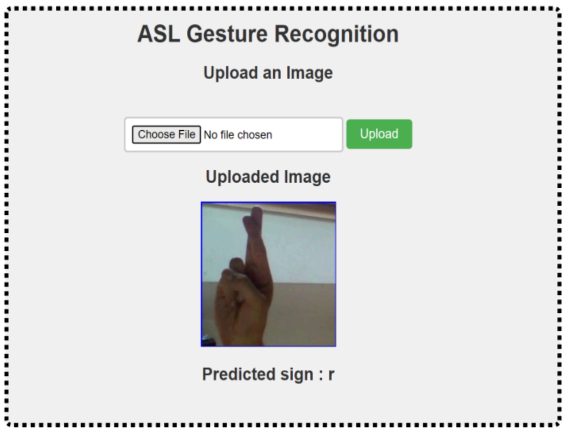
11.3.7. Classification Accuracy



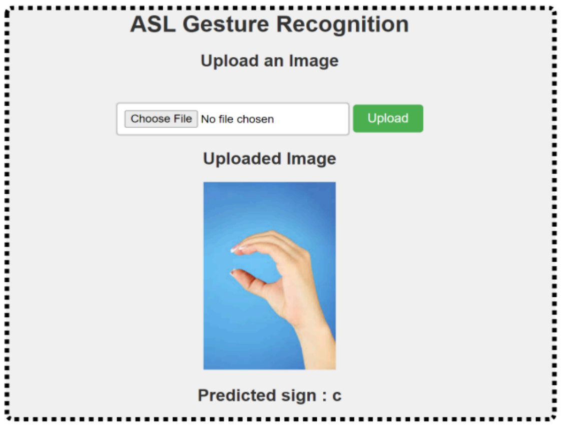
The classification accuracy of train and validation accuracy is increasing over time.

12. Predictions and Results

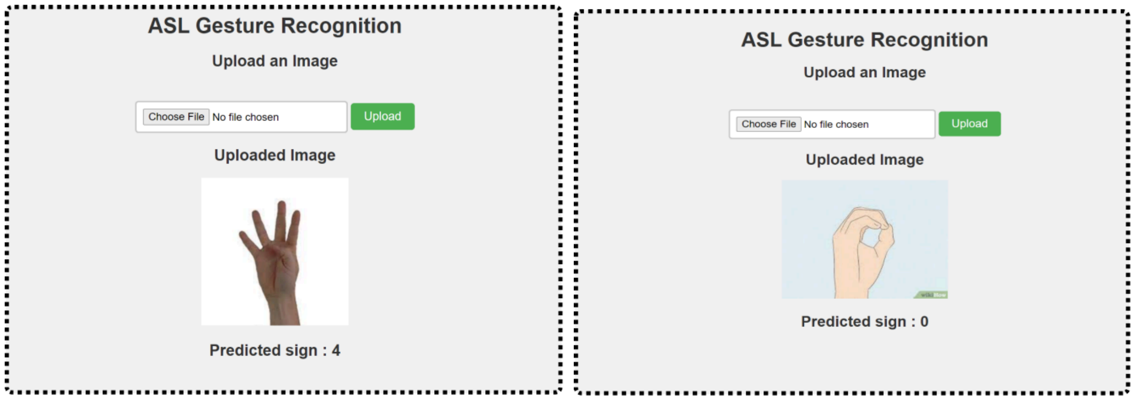
Assessing the model's practical performance on previously unseen images involved deploying it on a web interface through Flask. Relevant images were uploaded, and the subsequent analysis focused on the accuracy of the model's class predictions. Presented below are sample results obtained from the web interface.



Predicted sign : r



Predicted sign : c



Predicted sign : 4

Predicted sign : 0

Sample prediction outputs on Flask app

13. Future Work

In the future, the project aims to expand by focusing on developing a user-friendly interface and extending the model's capabilities to recognize continuous sign language. This extension will involve predicting sign language from videos and real-time camera input.

14. Conclusion

In summary, this project achieved the development of an American Sign Language (ASL) recognition system using machine learning (ML) techniques. While the VGG16 model demonstrates superior performance compared to the ResNet50 and Custom CNN models with slight variations, the use of a voting classifier ensemble method is advocated to minimize misclassification errors and obtain high-confidence class predictions from each model. By deploying this ASL recognition system, the aim is to bridge communication gaps between ASL users and individuals unfamiliar with sign language, thereby reducing feelings of frustration and isolation. Through the advancement of this technology, the project endeavors to promote inclusivity and improve communication accessibility for diverse user demographics.