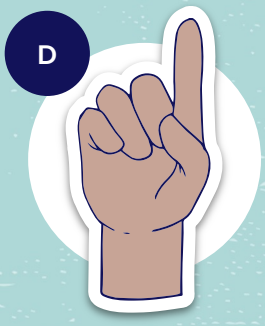


Sign Language Alphabetic Recognition

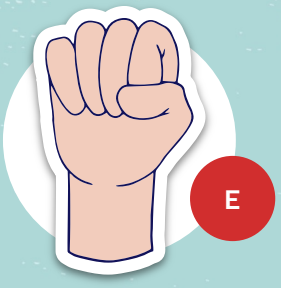


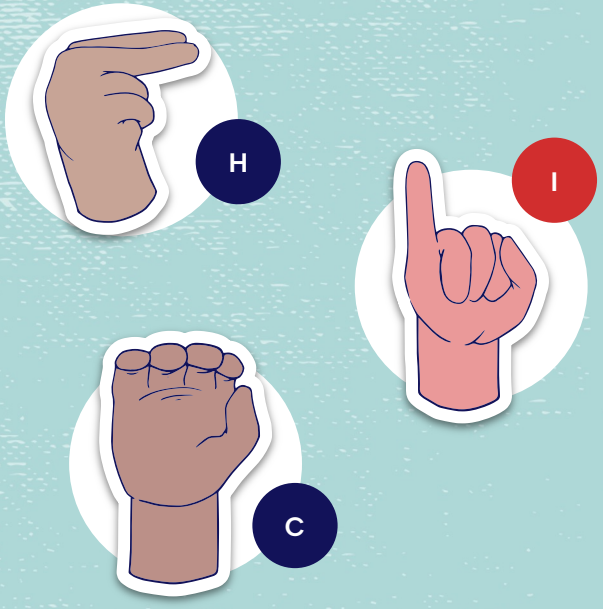
TABLE OF CONTENTS

01. Introduction

Sign language is a visual-gestural language used by deaf and hard-of-hearing individuals for communication.

Accurately recognizing and interpreting sign language alphabets poses a significant challenge

The objective is to develop machine learning models that interpret images of hand gestures, enabling accurate prediction of sign language alphabet letters



02.

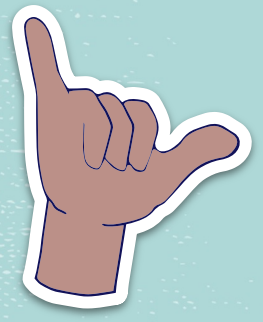
Related works

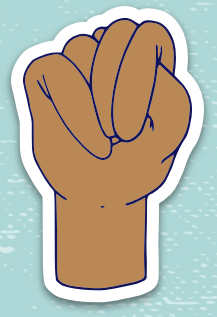
1st

Used a (CNN) model to categorize American Sign Language images of 24 English alphabets. Python's Keras library is used to build the model. The dataset format involves labels for each letter and 27,455 images with 784 columns. Converting the data into a (28,28,1) matrix achieved a 99.7% accuracy rate.

2nd

identified complex visual patterns with high validation accuracy using a CNN model, convolutional layers, and pooling layers. Training for more epochs increased accuracy, so the model's accuracy ranged from 0.4278 to 0.9526. Also, monitoring performance is crucial to avoid overfitting.



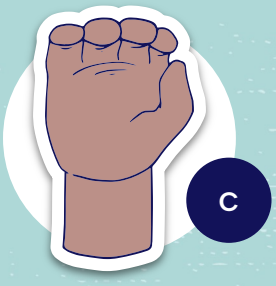


3rd

uses a Keras sequential model that reduces training and validation loss and improves accuracy by using dropout layers, deep regularization, batch normalization, and 15 epochs. Achieving 0.99885 test accuracy, max pooling and several layers are evaluated for maximum accuracy.

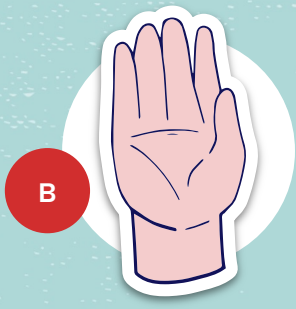
4th

The model uses a convolutional neural network architecture to predict signs, using 29 files and 87,000 images of American Sign Language alphabets, with a good validation accuracy of 0.97.



03.

Data



Dataset 1

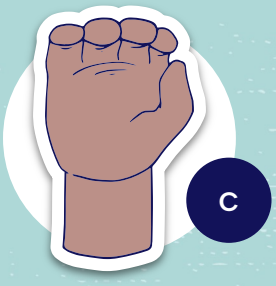
- created by gathering photographs of a volunteer's hand performing various ASL signs.
- It is presented in a CSV format with labels and pixel values in single rows
- Images are 28x28 pixels in size and are grayscale.
- The image's pixel values as well as a label, ranging from 0 to 25 representing the 26 different letters.
- Cases for the letters J = 9 and Z = 25 are not included because they require gesture movements.

Dataset	Number of examples	Number of features	Number of labels	Image dimensions
Training set	27455	785	26	28 x 28 pixels
Testing set	7172	785	26	28 x 28 pixels

Dataset 2

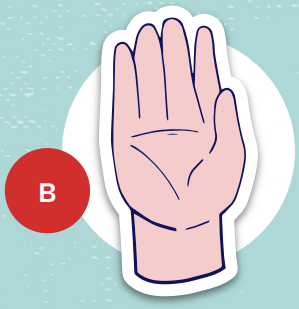
- Images of hands forming the alphabet in American Sign Language (ASL)
- The data is separated in 26 folders which represent the various classes, which contain A-Z alphabets
- The data set contains 1815 images which are 400x400 pixels, 70 images for each alphabet

Number of examples	Number of labels	Image dimensions
1815	26	400x400 pixels



04.

Methods



Methods

Random Forest

- Supervised learning technique
- Uses many decision trees together to provide predictions
- Each tree is trained on a distinct portion of the training set
- Two methods are used to introduce randomness :
  - Random feature selection at each split
  - Random data sampling for each tree

SVM

- Supervised learning technique
- Divides various classes in a high-dimensional feature space by identifying the best hyperplane
- Maximizing the margin between each class's closest data points and the hyperplane
- Uses a method known as the kernel trick to handle data

CNN

- Deep learning method
- Convolutional, pooling, and fully linked layers are some of the layers that make up a CNN
- Convolutional layers apply filters
- The feature maps are down sampled using pooling layers
- Fully linked layers to create forecasts

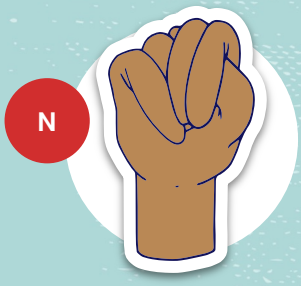
Why these methods?

Random Forest

Provides a measure of feature importance, indicating which image features contribute the most to the classification of hand gestures.

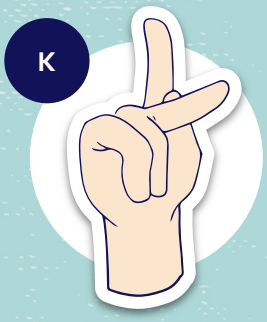
SVM

Performs well in high-dimensional spaces and can learn complex decision boundaries by mapping the data into a higher-dimensional feature space using the kernel trick.



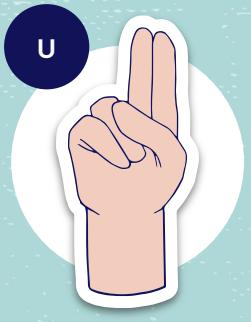
CNN

Inherently translation invariant, meaning they can recognize patterns regardless of their position in the image.

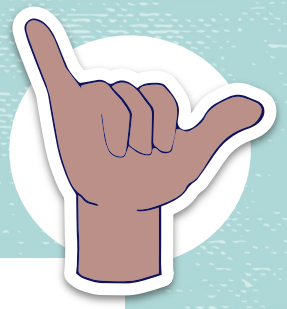


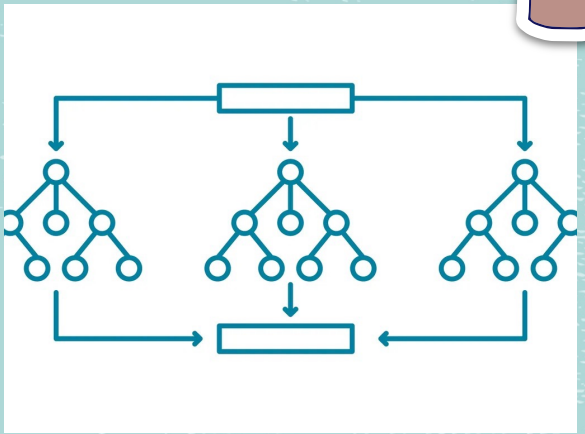
05.

Experiments



Random Forest Model:





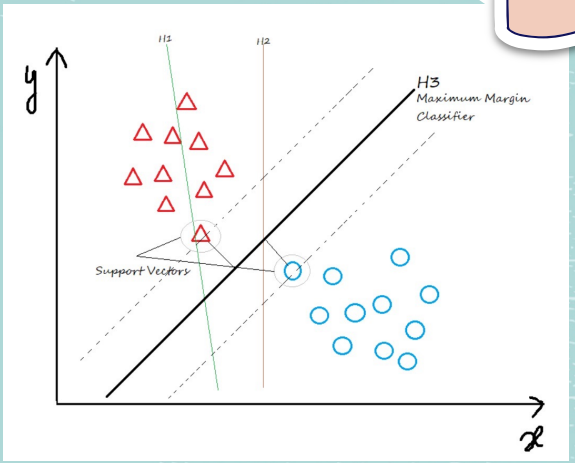
1- Data Preparation: Reshape training and test data for RandomForestClassifier compatibility.

2- Hyperparameter Tuning: Use GridSearchCV to find optimal RandomForestClassifier hyperparameters.

3- Model Evaluation: Assess model performance on the test set using metrics like accuracy, precision, recall, and F1-score.

4- Confusion Matrix Visualization: Generate and visualize multilabel confusion matrices for clarity using Matplotlib and Seaborn.

SVM Model:



1- Data Preprocessing: Standardize features with StandardScaler.

2- Feature Engineering: Apply Randomized PCA for dimensionality reduction.

3- SVM Classifier: Use SVM with RBF kernel, and address imbalanced classes with class_weight.

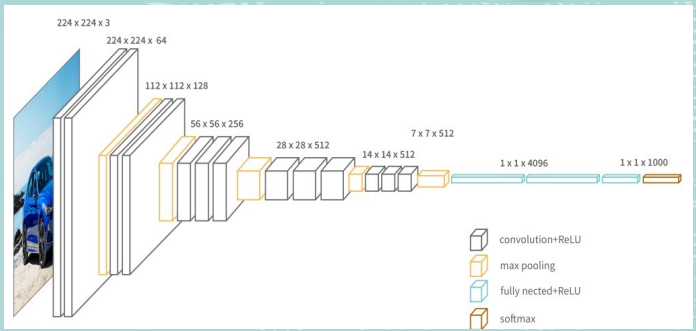
4- Dataset Splitting: Split into 1452 training and 363 testing data points.

5- Hyperparameter Tuning: Employ GridSearchCV for SVM (C, gamma) hyperparameters.

6- Evaluation: Assess model via cross-validated scores, then report mean, standard deviation, and optimal hyperparameters.

7- Computational Resources: Measure training time using %time, and recommend Jupyter with GPU for efficiency.

CNN Model:



1- Data Preparation:

- Label Binarization.
- Reshape input data for CNN.
- Augment image data.

2- Architecture:

- Convolutional layers (75, 50, 25) with decreasing filters.
- Batch normalization, ReLU, and dropout.
- MaxPooling for downsampling.
- Dense layers with ReLU and softmax.

3- Batch Tuning:

- Tried batch sizes 32 and 128.
- Batch = 32: 40% accuracy for 1st dataset, higher for 2nd.
- Batch = 128: High accuracy for both.

4- Training:

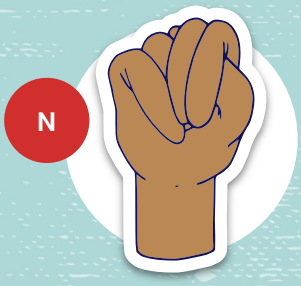
- Early stopping and learning rate reduction.
- Adam optimizer, categorical cross-entropy loss.
- 20 epochs with GPU.

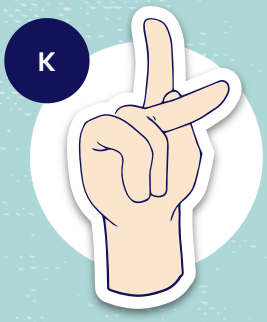
5- Evaluation:

- Assess on training and validation.
- Print accuracy and loss metrics.

06.

Results and Discussions





SVM with RBF kernel

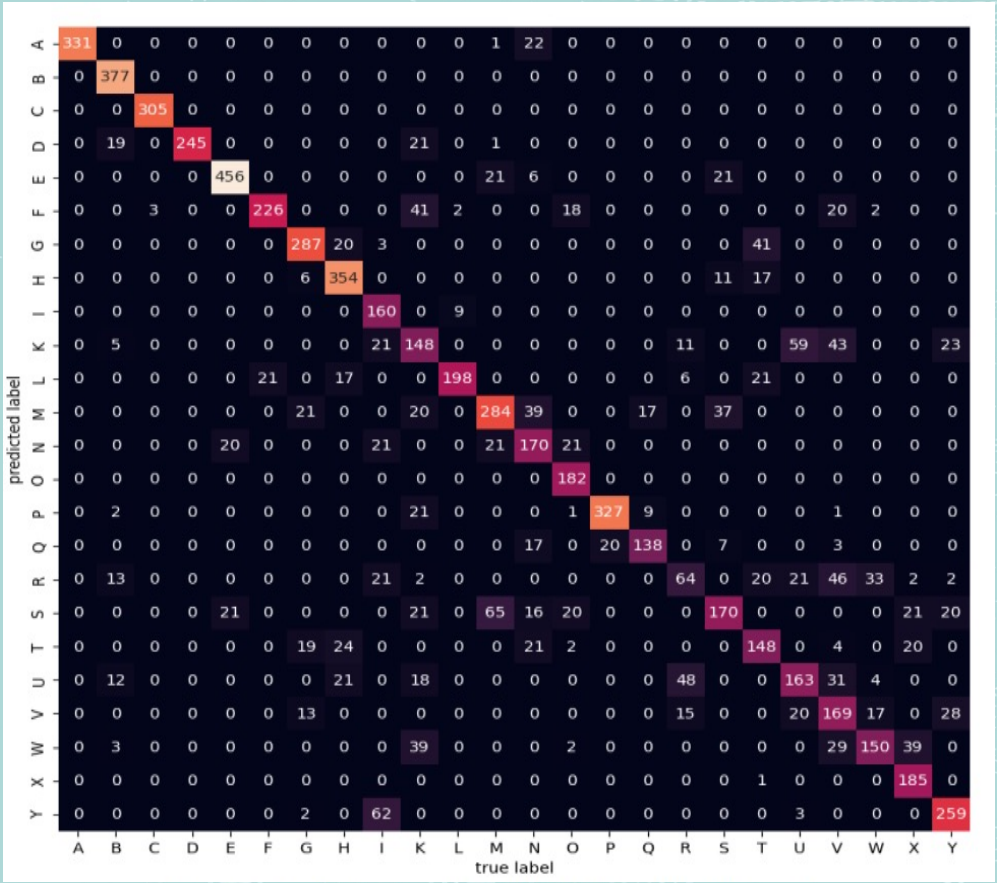
Dataset #1

Gamma =0.005 , C=1

performance metrics

Performance Metric	Value
Accuracy	0.77
F1 score	0.77
Precision	0.79
Recall	0.77

Confusion matrix



True vs Predicted labels

SVM with RBF kernel

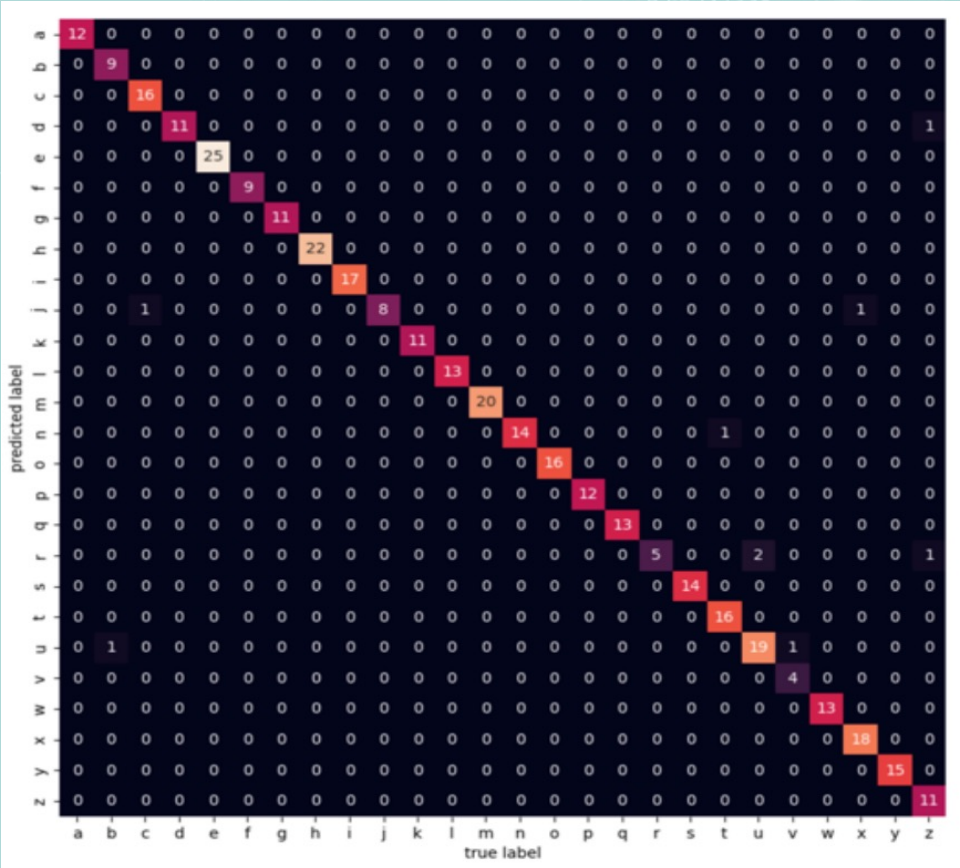
Dataset #2

Gamma=0.001, C=10

performance metrics

Performance Metric	Value
Accuracy	0.98
RBF F1 score	0.98
Precision	0.98
Recall	0.98

Confusion matrix



True vs Predicted labels

Random Forest

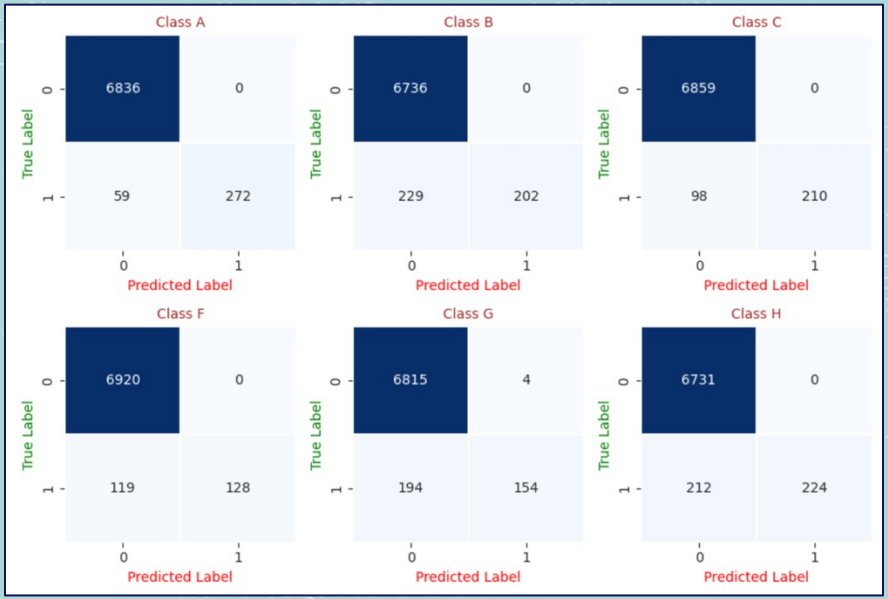
Dataset #1

n_estimators = 20 , best max_depth = 200

performance metrics

Performance Metric	Value
Accuracy	0.32
F1 score	0.99
Precision	0.32
Recall	0.49

Confusion matrix



Random Forest

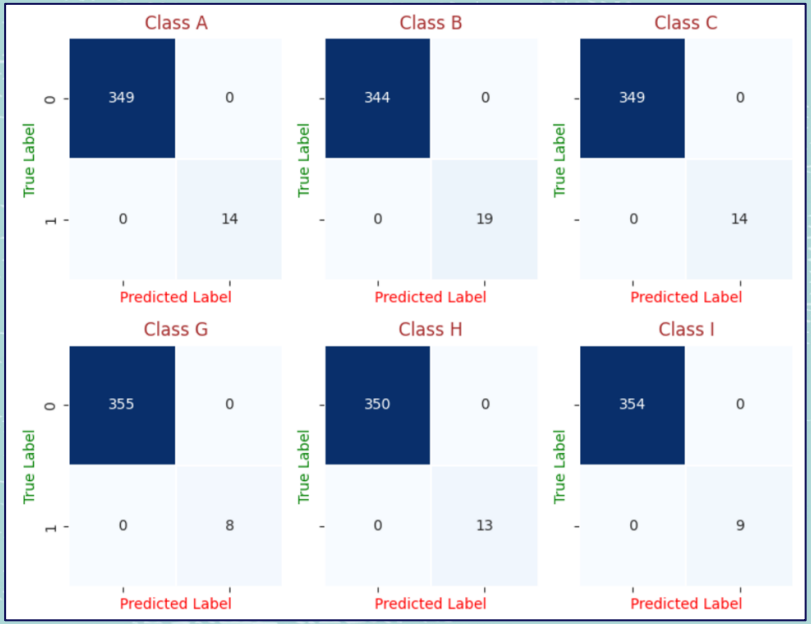
Dataset #2

n_estimators = 20 , best max_depth = 100

performance metrics

Performance Metric	Value
Accuracy	0.98
F1 score	0.98
Precision	0.98
Recall	0.98

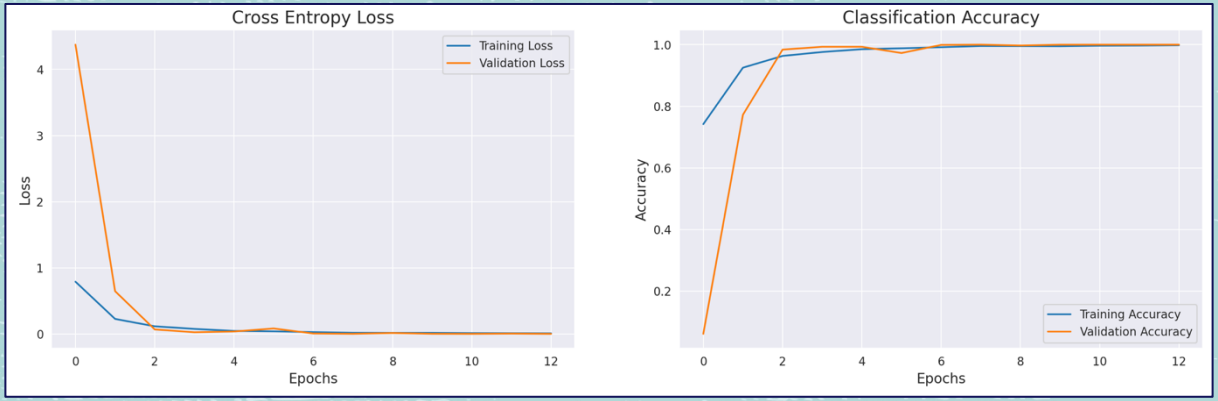
Confusion matrix



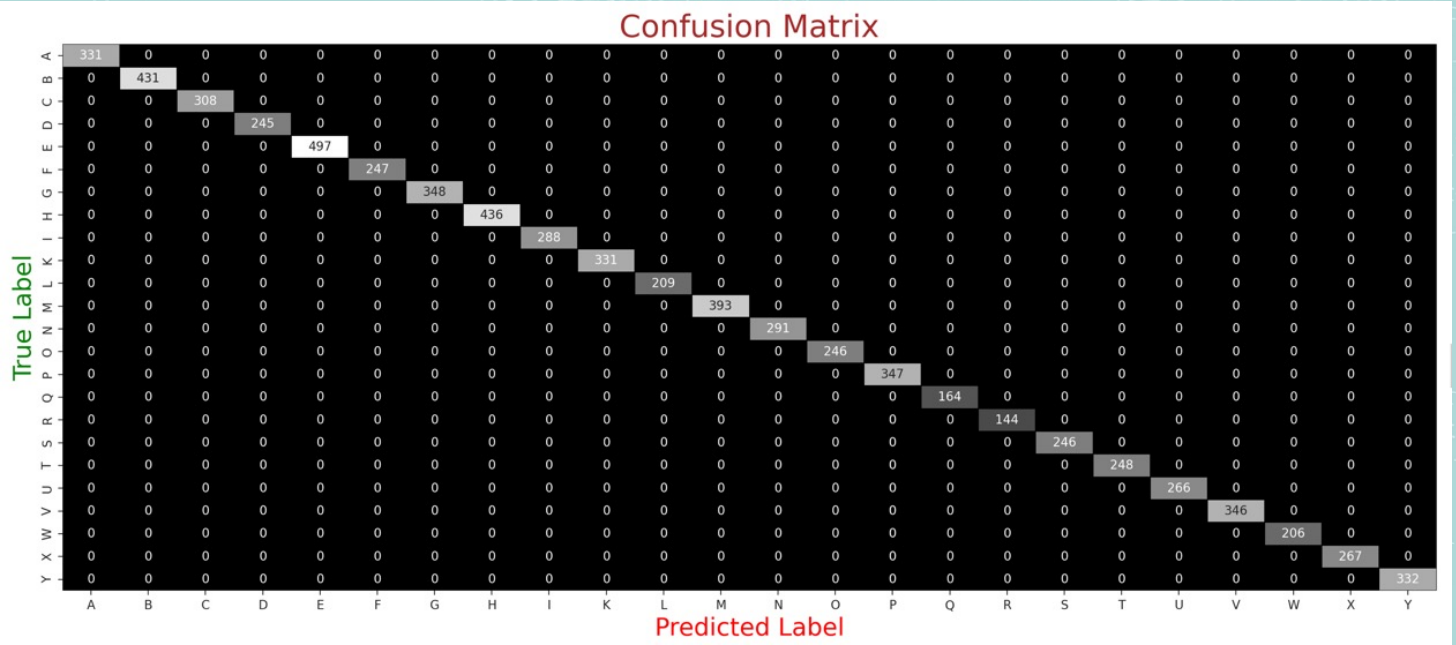
CNN

Dataset #1

Evaluation	Value
Accuracy of the model for training data	100.0
Loss of the model for training data	0.00058
Accuracy of the model for validation data	100.0
Loss of the model for validation data	0.0016



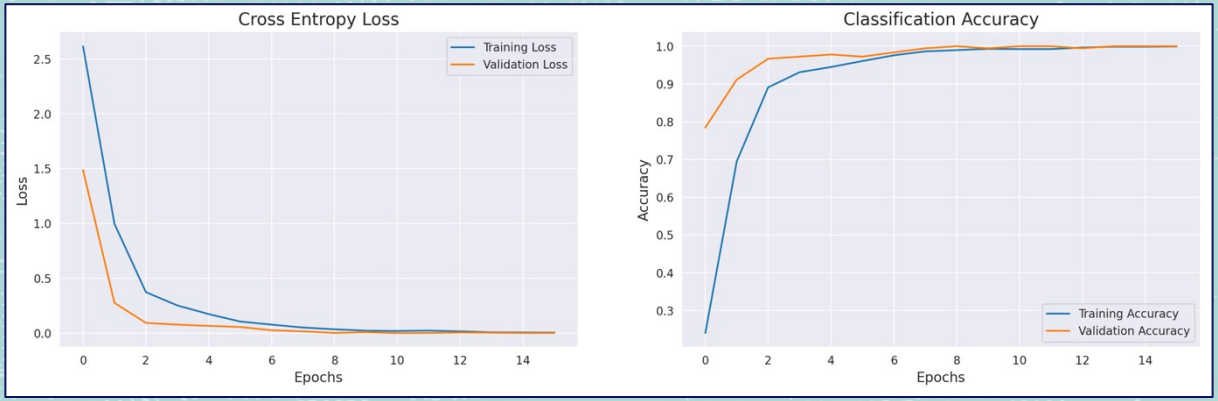
Confusion matrix



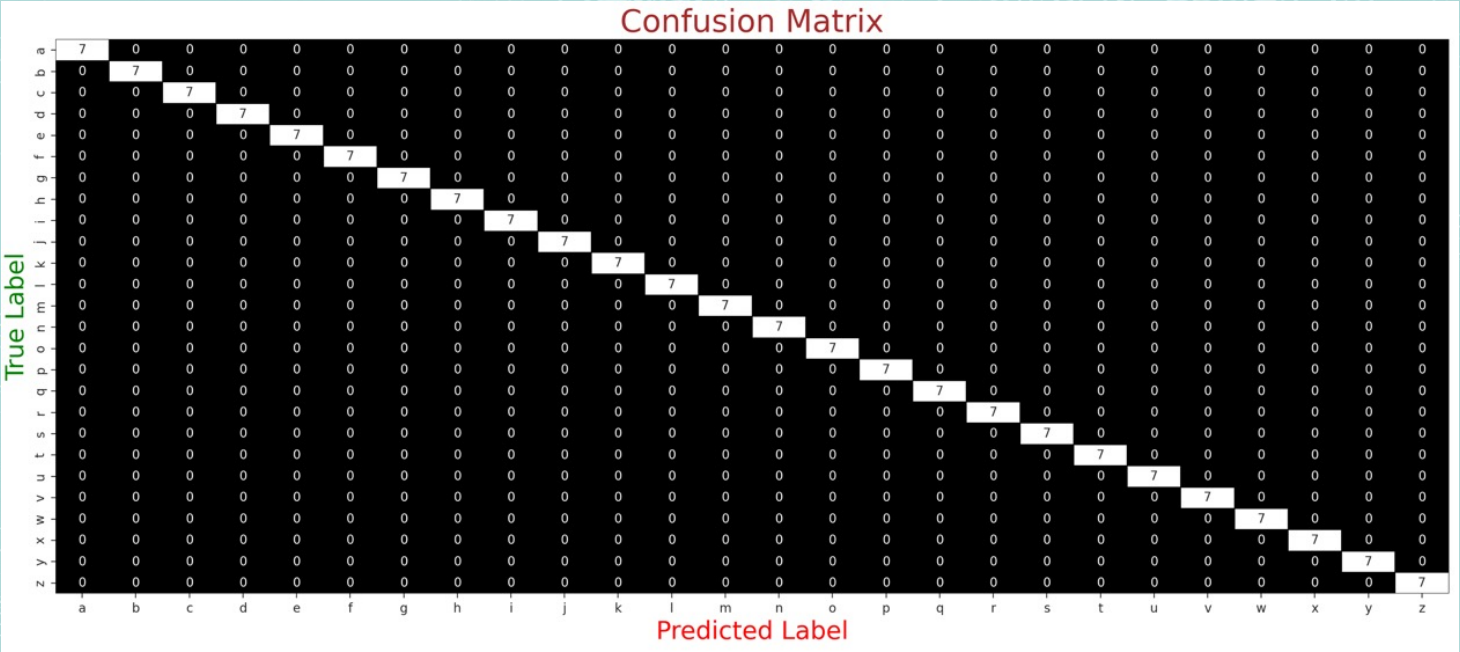
CNN

Dataset #2

Evaluation	Value
Accuracy of the model for training data	99.93
Loss of the model for training data	0.001061
Accuracy of the model for validation data	100.0
Loss of the model for validation data	0.00033
Accuracy of the model for testing data	100.0
Loss of the model for testing data	0.00025



Confusion matrix



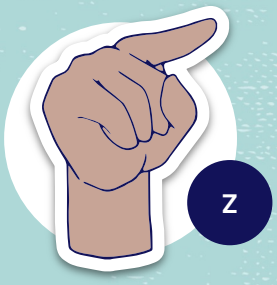
07. Conclusion

Key Findings:

Varying effectiveness across datasets, with CNN consistently outperforming.

Challenges:

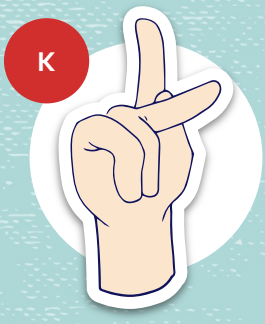
refining Random Forest for precision and addressing SVM misclassifications



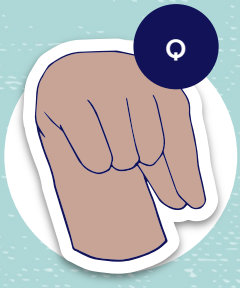
Future work:

exploring advanced neural architectures, real-time data integration, and ensemble methods for improved sign language recognition.

THANKS!



Do you have any questions?



Prepared by:

Layan AlTowaijri

Madawi Alfozan

May AlNajim

Alanoud Alghayth

Supervised by: Dr. Luluah Alhusain