HandTalk: Translating Sign language to Text

Aakash - 2021002
Parveen - 2021079
Shubham Sharma - 2021099
Pourav Surya - 2021271

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Motivation

The motivation behind developing a machine learning based sign language detection system stems from the need to bridge communication gaps between deaf or hard-of hearing individuals and the hearing population. This underrepresentation leads to social and professional barriers for those who rely on sign language as their primary means of communication.

Literature Review

- A New Benchmark on American Sign Language Recognition using Convolutional Neural Network: This study proposes a novel convolutional neural network (CNN) model to enhance American Sign Language (ASL) recognition accuracy. Evaluated on four publicly available ASL datasets, the model, applied to alphabet and numeral images, achieves a 9% improvement in accuracy over existing methods

- Real-Time Sign Language Detection Using CNN : To detect real-time sign language, a dataset is prepared on which a customized CNN model is trained. In the findings, it was observed that the customized CNN model can achieve the highest 98.6% accuracy.

- Deep convolutional neural networks for sign language recognition : To address the lack of mobile selfie sign language datasets, they created one with five subjects performing 200 signs from five angles and various backgrounds, capturing 60 frames per sign. Their approach achieved a 92.88% recognition rate

Dataset

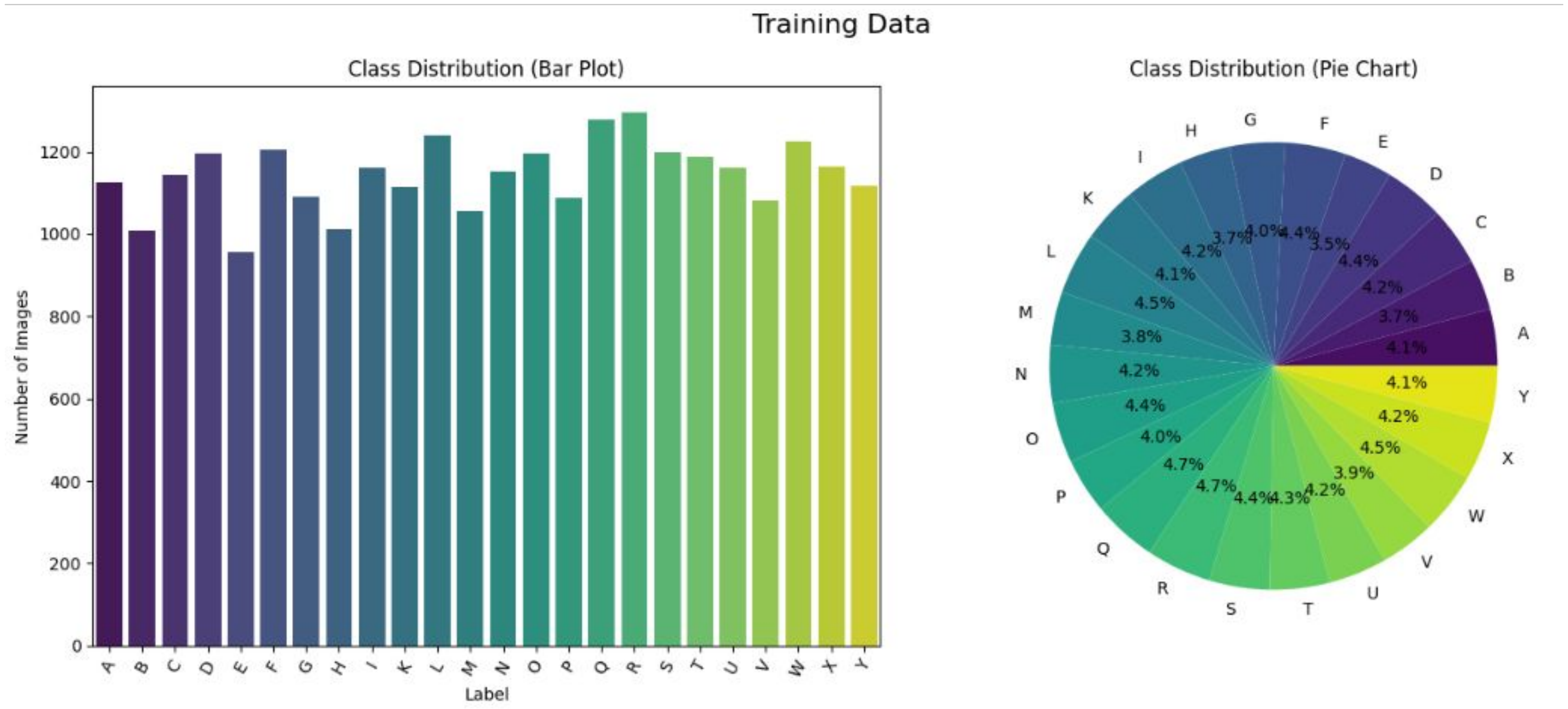
The Dataset used for the training and testing purpose is Sign Language MNIST. This dataset contains the American Sign Language data. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). There is a total of 784 pixel which represent a single 28x28 pixel image with grayscale values between 0-255.

Methodology

The major methodology part consists of applying different model on the trained data, calculating and analyzing their accuracy, hyperparameter tuning wherever needed and finding the model that performs best in terms of accuracy or loss.

The models used for classification purpose were Logistic Regression, Decision Tree, Random Forest, Perceptron, Multi Layer perceptron. Each model is trained on the best parameter found using grid search cross validation with 3 fold cross validation, and accuracy, precession, recall and F1 score is calculated and analyzed.

Data Visualization and Processing



All the grayscale pixel values of the images are normalized to scale them to a range of [0, 1].

Model Training process

First several hyper-parameters are defined for each model. Then the hyperparameters values are tuned using 3 fold grid search cross validation for each of the model.

After finding the best value of the parameters, each model is trained on those parameters value, and performance metric is calculated along with the confusion matrix.

Results

After performing the grid search on the parameters described for the specific model, the best parameters for the following models were found. The models were then trained on their respective best parameters and accuracy, precision, recall and F1 score was calculated to find the best model out of these models.

Results

Logistic Regression

Hyperparameter	Values (Grid Search)	Best Value
penalty	{'l1', 'l2', 'elasticnet', None}	'l1'
C	{0.01, 0.1, 1, 10}	0.1
solver	{'saga'}	'saga'
max_iter	{100, 200}	200

Random Forest

Hyperparameter	Values (Grid Search)	Best Value
n_estimators	{50, 100, 200}	200
criterion	{'gini', 'entropy'}	'gini'
max_depth	{None, 10, 20}	20
min_samples_split	{2, 10}	2
min_samples_leaf	{1, 5}	1

Decision Tree

Hyperparameter	Values (Grid Search)	Best Value
criterion	{'gini', 'entropy'}	'entropy'
splitter	{'best', 'random'}	'best'
max_depth	{None, 10, 20, 30}	None
min_samples_split	{2, 10, 20}	2
min_samples_leaf	{1, 5, 10}	1

Perceptron

Hyperparameter	Values (Grid Search)	Best Value
penalty	{'l1', 'l2', 'elasticnet', None}	'l1'
alpha	{0.0001, 0.001, 0.01}	0.001
max_iter	{1000}	1000
tol	{1e-3}	0.001

Multi-Layer Perceptron

Hyperparameter	Values (Grid Search)	Best Value
hidden_layer_sizes	{(128,), (128, 64), (256, 128)}	(256, 128)
activation	{'relu', 'tanh'}	'relu'
solver	{'adam', 'sgd'}	'adam'
alpha	{0.0001, 0.001}	0.001
learning_rate	{'constant', 'adaptive'}	'constant'
max_iter	{500}	500

Performance Metrics

All the models with their best parameters are trained on the train dataset and then were test on the testing dataset. For all the models accuracy, precision, recall and F1-score are calculated on the test dataset and summarized in the below table.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.693809	0.724626	0.693809	0.698777
Decision Tree Classifier	0.478667	0.498118	0.478667	0.483469
Random Forest Classifier	0.811907	0.835269	0.811907	0.815962
Perceptron	0.265477	0.651270	0.265477	0.298552
MLP Classifier	0.761433	0.766137	0.761433	0.758345

It can be inferred that Random Forest Classifier is the best model, outperforming other models in terms of accuracy, precision, recall and F1 score. This indicates that the Random Forest model is the most effective at correctly classifying the instances, maintaining a good balance between precision and recall.

Timeline

- Week 1 (28th August - 3rd September): Gather and preprocess the sign language dataset. Convert images to grayscale and normalize the pixel values. Review relevant literature on sign language recognition and machine learning techniques.

- Week 2-3 (4th September - 17th September): Conduct exploratory data analysis (EDA) to understand the dataset. Visualize the data and identify any patterns or trends in the sign language gestures.

- Week 4-7 (18th September - 22nd October): Develop initial models using Logistic Regression, Decision Trees, and Support Vector Machines (SVM). Train models on the dataset and evaluate initial performance.

- Week 8 (30th October - 5th November): Experiment with advanced machine learning algorithms, such as Random Forests and Multi-Layer Perceptron (MLP). Optimize the models through hyperparameter tuning using techniques like Grid Search or Random Search.

- Week 9 (5th November - 12th November): Evaluate model performance on the test dataset using metrics such as accuracy, precision, recall, and F1 score. Compare different models to determine the best-performing one.

- Week 10-11 (13th November - 20th November): Test the final system and finalize the project report, documenting the methodology, results, and conclusions. Prepare the project presentation and demo for submission.

Individual team members' contributions

- Aakash: Decision Tree, Hyperparameter tuning, Dataset processing
- Parveen: Logistic Regression, Perceptron, Hyperparameter tuning, Dataset processing
- Shubham Sharma: EDA, Random Forest, Hyperparameter tuning
- Pourav Surya: MLP classifier, Confusion matrix, Testing on data set

THANK YOU