CS 725 Foundations of Machine Learning
Project Report
Hand Gesture Recognition

Abisek R K 21Q050004
Tejpal Singh 21Q050008
Naveen Badathala 213050052
Mohan Rajasekhar Ajjampudi 213050060

November 26, 2021

Contents

1 Introduction

Hand Gesture Recognition is very important for human-computer interaction in various applications of AR/VR, Robotics etc. A large portion of previous research on this topic requires specialized hardware like depth sensors and not light-weight enough to run on real-time. To address this problem, we developed a real time application which can be used to recognize Hand Gestures effectively. Along with Default Gestures i.e the gestures which are pre-trained to the application, our application is smart enough to learn and recognize custom gestures taken from the user.

2 Data Collection and Generation

To predict default gestures, we used ASL alphabet dataset,[1] which is a collection of American Sign Language(ASL) alphabet images. It is a group of 28 different folders including space and delete gestures. Each alphabet class consists of 3000 RGB images with a resolution of 200 X 200 each.

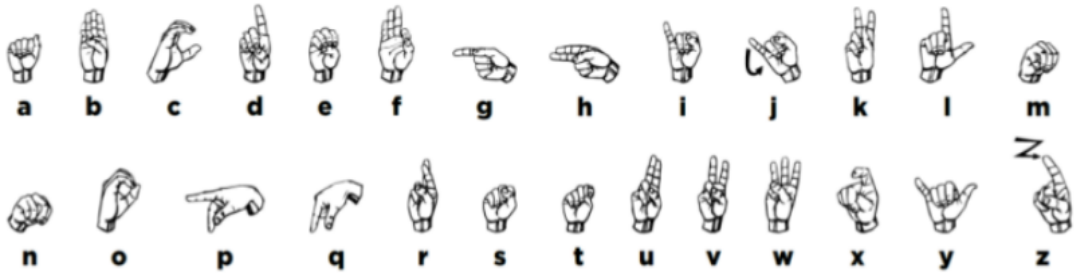

Figure 1: ASL alphabet

For each image in the dataset, we generated 21 landmarks features i.e x, y and z coordinates using Mediapipe Library.[2] Mediapipe Hands is an efficient light-weight solution to track hand coordinates in real time. It gives 21 3-D landmarks of hand for a given single frame. These 21 landmarks are spread across the hand covering important locations of hand.

Figure 2: 21 landmarks of hand

3 Data Preprocessing

- There are some images in our dataset where Mediapipe is unable to detect landmarks. Hence we removed those images from our dataset before training which reduces each class count to around 2k images.
- Oversampling techniques are followed before training to balance all classes data.
- For custom gesture recognition, the images are captured only when the landmarks are detected.
- Z-value in landmarks corresponds to image depth, if it exists per coordinate. Also, it is not of importance while dealing with static gestures. We dropped Z-value from our features set because of this limitation.
- We applied Principal Component Analysis and Linear Discriminant analysis on our training set data to understand the distribution variance of classes. Number of principal components to explain 95%, 99% Variance is 8 and 12 respectively.
- But we proceeded with using all the 42 features to train our models as we are implementing custom gestures recognition along with default gestures.

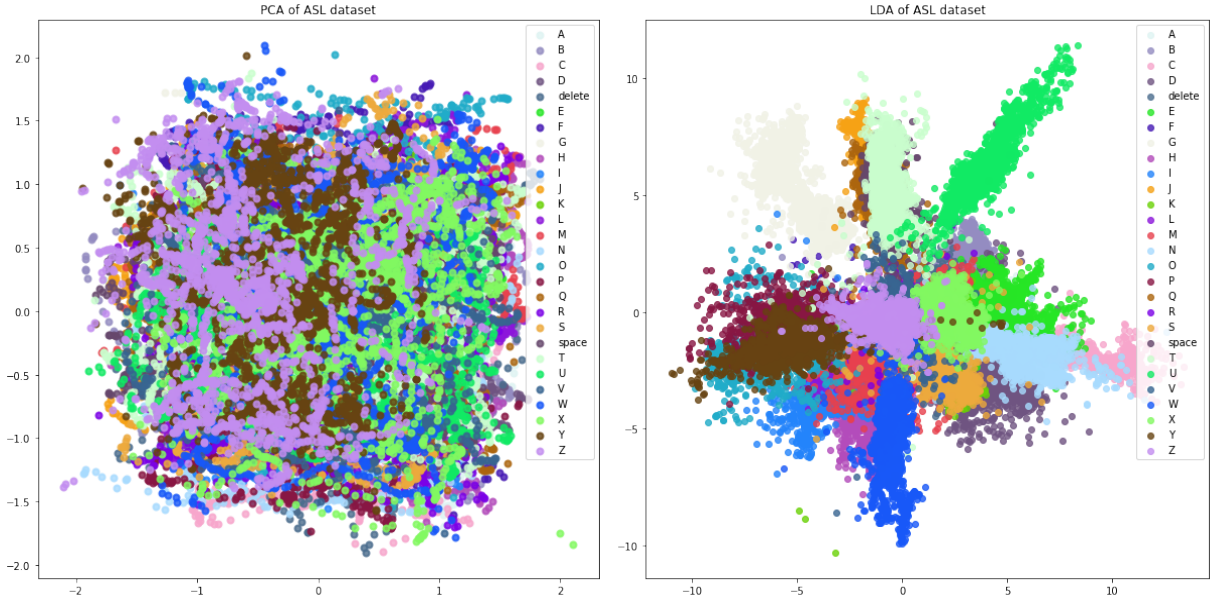

Figure 3: PCA and LDA of ASL Dataset

4 Methodology

Our Methodology involves taking preprocessed images and feeding them to Mediapipe to get coordinates data as feature set. These features are forwarded to different Machine learning and Deep Learning models for training. These learned models are able to predict hand gestures. In Default Gestures flow, preprocessed images are passed through "Mediapipe"

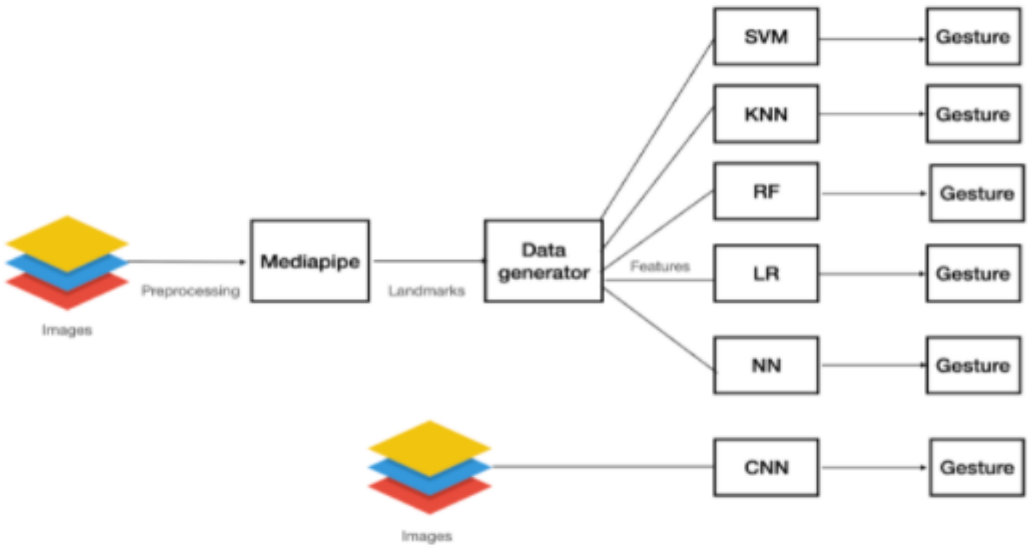

Figure 4: Process Flow

library to obtain corresponding 21 Landmarks data i.e cartesian coordiantes for each image. ML/DL models are trained on these image features represented by their landmark coordinates. The user will be able to input a gesture through webcam, which will be passed through the mediapipe and ML models layers, eventually predicting the text (or voice) equivalent to the input gesture.

In Custom Gestures flow, User will have an option to add custom gesture on the go. User can define a custom gesture and input a few samples of the gesture. From the samples received as input, important landmark coordinates will be identified and added to the existing training data. Our light-weight ML models will be retrained on this new data within minutes. The user can then use these new gestures as if they were already present before.

5 Machine learning Models

We took this problem as multi-class classification problem. We applied various machine learning and deep learning techniques maintaining Train-test split of 80-20.

5.1 Logistic Regression

Logistic Regression classifier in the multiclass setting, the training algorithm uses the one-vs-rest (OvR) scheme if the multi-class option is set to multinomial.[3] We used solver 'lbfgs' with maximum iterations as 1000 to converge to optimum.

Parameters: Solver: lbfgs, maximum iterations = 1000
Accuracy Score on test data: 0.9601

5.2 K-Nearest Neighbors

KNN classifier implements the multi-class classification problem based on k-nearest neighbors vote.[4] There are 28 neighbours comprising 26 alphabets and space,delete gestures.

Parameters: n_neighbors=28 and weights parameter is kept as uniform.
Accuracy Score on test data: 0.9326

5.3 Random Forest

Random Forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.[5]

Parameters: n_estimators = 100.(The number of trees in the forest)
Accuracy Score on test data: 0.97921

5.4 Neural Networks

The neural network model contains two hidden layers with 100 and 50 nodes each with 'relu' activation function and the output layer comprises of 28 nodes with 'softmax' activation function.[6] The output labels are encoded alphabetically from 0 to 27.

Parameters: Loss function: Sparse categorical cross entropy, used for multi-class classification model. Optimizer: Adam.
Accuracy score on test data: 0.9864

5.5 Convolutional Neural Networks

The CNN model starts with two layers of convolution layers followed by a max pooling layer of filter size 3x3. This set is repeated 3 times by increasing the number of filters in sizes 32,64,128 respectively and each layer uses 'relu' activation function. The output layer uses a 'softmax' activation function and is preceded by a dropout layer.[6]

Parameters: Optimizer: Adam. Loss function: sparse categorical cross entropy.
Accuracy Score: We got a train accuracy score of 0.9915 and test accuracy score of 0.9839.
Observations: The CNN model although giving a high test accuracy is performing poorly on new test data and is unable to predict the correct gestures most of the time. This can

be due to the differences in background, skin color and hand orientation. Also, the training time of CNN model is relatively larger.

5.6 Support Vector Machines

SVM classifier, in multiclass support is handled according to a one-vs-one scheme. [7] one-vs-one approach splits the dataset into one dataset for each class versus every other class.

Parameters: We tested using different kernels such as linear, polynomial, rbf kernels. Out of which polynomial kernel performed the best with more accuracy.

- Linear Kernel Accuracy: 0.9720
- Polynomial Kernel Accuracy: 0.9861
- Radial Basis Function Accuracy: 0.9783

Observations: SVM model gave the best accuracy out of all the ml models we tested and we employed it for detecting the custom gestures as well.

Following figures shows the feature importance of 42 features of the SVM model.

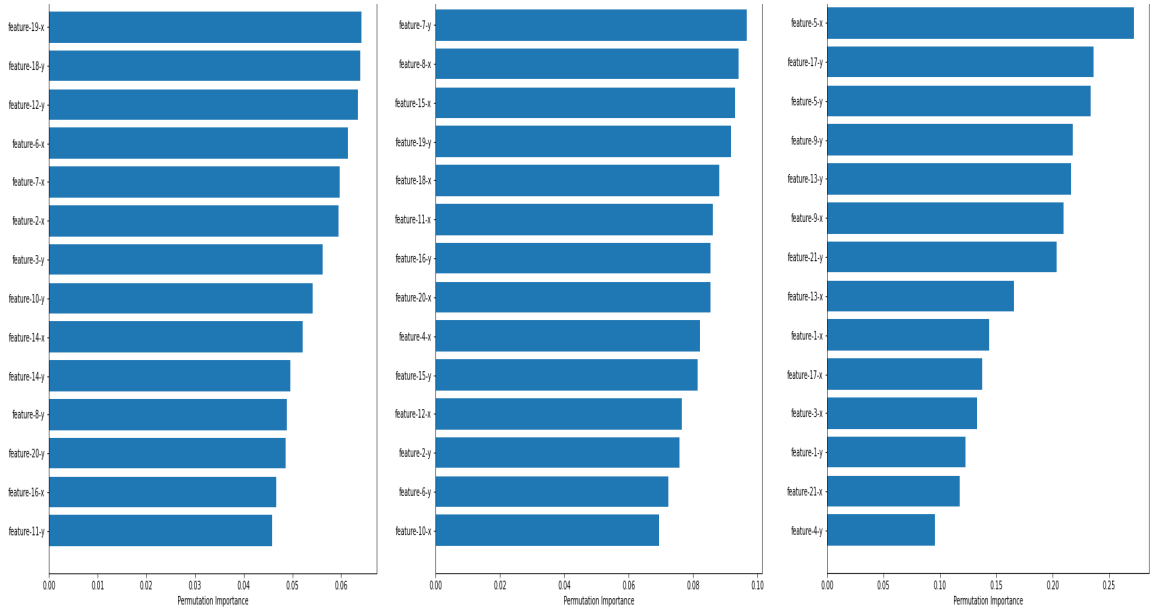

Figure 5: Feature Importance of SVM model

Following figures shows the Precison, Recall, F1-Score values of all 28 classes of ASL dataset for SVM Model along with multi-class ROC Curve.

	precision	recall	f1-score
A	0.974425	0.974425	0.974425
B	0.983412	0.992823	0.988095
C	0.996071	0.998031	0.997050
D	0.992537	0.992537	0.992537
E	0.971047	0.990909	0.980877
F	1.000000	0.986434	0.993171
G	1.000000	0.998165	0.999082
H	0.998039	0.996086	0.997062
I	0.995204	0.990453	0.992823
J	0.991379	0.997831	0.994595
K	0.992248	0.994175	0.993210
L	0.996350	0.992727	0.994536
M	0.937500	0.979899	0.958231
N	0.991690	0.970190	0.980822
O	0.977011	0.992991	0.984936
P	0.991453	0.977528	0.984441
Q	0.992167	0.992167	0.992167
R	0.983838	0.970120	0.976931
S	0.980645	0.987013	0.983819
T	0.991150	0.986784	0.988962
U	0.965831	0.970252	0.968037
V	0.971698	0.969412	0.970554
W	1.000000	0.977728	0.988739
X	1.000000	0.980535	0.990172
Y	0.993348	0.986784	0.990055
Z	0.996000	0.990060	0.993021
del	0.953782	0.974249	0.963907
space	0.951613	0.979253	0.965235
accuracy	0.986196	0.986196	0.986196
macro avg	0.984587	0.985341	0.984910
weighted avg	0.986360	0.986196	0.986231

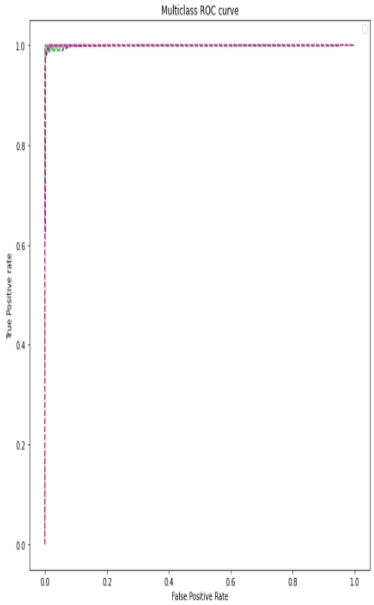

Figure 6: Results and ROC curve of SVM Model

6 Results

Following table shows the F1-score results of the various ml models we tested without oversampling the data.

Model	Micro average	Macro Average	Weighted Average
Logistic Regression	0.959886	0.956627	0.959895
KNN	0.929760	0.928055	0.930067
SVM	0.986114	0.984919	0.986139
Random Forest	0.980106	0.978534	0.980074
Neural Networks	0.983760	0.982229	0.983793

Following table shows the F1-score results of the various ml models we have tested by oversampling each class to the highest class data.

Model	Micro average	Macro Average	Weighted Average
Logistic Regression	0.960130	0.956708	0.960227
KNN	0.932603	0.931085	0.933061
SVM	0.986196	0.984910	0.986231
Random Forest	0.979050	0.977642	0.979032
Neural Networks	0.984166	0.982567	0.984185

7 Error Analysis

To analyse misclassification errors, we grouped confusing letters of ASL alphabets which are almost same in gesture into two groups as shown below. One set of group consists of letters A, E, S and other consists of letters K, U, V.

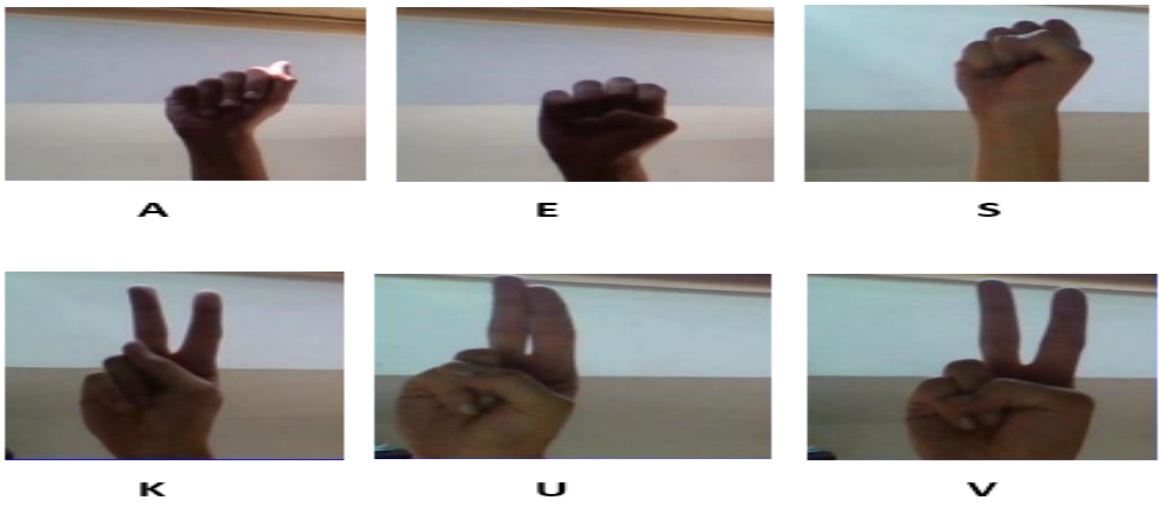

Figure 7: Misclassification Analysis of ASL Letters

We analysed our trained models accuracy results on this two set of groups to show how they are performing on this. Our best model, SVM is able to classify both the group of letters effectively. Also, we observed that the accuracy scores of these group of letters is lower compared to the remaining letters. For two groups, the detailed results of various models i.e Precision, Recall and F1-score are shown in below tables.

Model	Precision			Recall			F1-Score		
	A	E	S	A	E	S	A	E	S
Logistic Regression	0.9095	0.9595	0.9626	0.9514	0.9682	0.9459	0.9300	0.9638	0.9541
KNN	0.9055	0.9405	0.8645	0.9309	0.9341	0.9113	0.9180	0.9373	0.8873
SVM	0.9744	0.9710	0.9806	0.9744	0.9909	0.9870	0.9744	0.9809	0.9838
Random Forest	0.9794	0.9817	0.9535	0.9719	0.9773	0.9762	0.9756	0.9795	0.9647
Neural Networks	0.9896	0.9774	0.9642	0.9719	0.9818	0.9913	0.9806	0.9796	0.9776

Model	Precision			Recall			F1-Score		
	K	U	V	K	U	V	K	U	V
Logistic Regression	0.9863	0.8867	0.9616	0.9786	0.9314	0.9435	0.9825	0.9085	0.9525
KNN	0.8938	0.6599	0.8342	0.8660	0.8124	0.7694	0.8797	0.7282	0.8005
SVM	0.9922	0.9658	0.9717	0.9942	0.9703	0.9694	0.9932	0.9680	0.9706
Random Forest	0.9882	0.9402	0.9480	0.9748	0.9359	0.9435	0.9814	0.9381	0.9458
Neural Networks	0.9903	0.9771	0.9741	0.9922	0.9771	0.9741	0.9913	0.9771	0.9741

8 Challenges

- We identified that landmarks data is not derivable for some images due to various constraints like position of gesture, lighting conditions etc. This was addressed in training data and in case of custom gestures input, we captured the input images only when the landmarks are detected
- For Custom Gestures, Retraining model along with added gestures requires some computation time. To reduce this we tried various techniques like discovering most important features using PCA, LDA considering the prediction results accuracy.
- To increase the sample size while adding a new gesture, we explored different techniques involving oversampling input image data and augmenting images with different variations like rotate, flip, zoom etc. In the end, these techniques didn't show any improvements. Hence we restricted user input images to 10.

9 Application Details

We implemented an application using Tkinter and OpenCV to show how our trained models are predicting default ASL gestures and a dedicated interface to add custom gestures and predict them along with default ASL letters and convert them into text and speech in real-time.

There are two different flows involved here. In Default Gestures flow, User can select "Predict Default Gestures" option and proceed predicting corresponding letter associated with the gesture captured on screen. These can be extended to letters and sentences.

In Custom Gesture flow, user can capture images from the dedicated UI by clicking on "Add Custom Gesture(s)" option and once that is done, User can select "Predict Gestures" option which will retrain the model with this new gesture and allows the user to predict the newly added gesture along with the ASL default gestures.

10 Conclusion and Future work

We explored various machine learning and deep learning techniques to solve Hand Gesture Recognition. Our developed application is able to recognize custom gestures along with default gestures effectively in real-time. This can be further extended to recognize dynamic gestures.

11 Code and Demo links

- **Github Repository Link** :
  https://github.com/naveen-badathala/CS725-2021-Hand_Gesture_Recognition
- **Demo Link** :
  https://drive.google.com/file/d/1iqJGpvdR-TV2sI9IWbKrx9mZS9vaur4q/view

12 Bibliography

References

[1] https://www.kaggle.com/grassknoted/asl-alphabet

[2] https://google.github.io/mediapipe/solutions/hands.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[6] https://keras.io/guides/sequential_model/

[7] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[8] Mediapipe Paper Reference: https://arxiv.org/pdf/2006.10214.pdf

[9] https://google.github.io/mediapipe/solutions/hands.html

[10] https://techtutorialsx.com/2021/04/10/python-hand-landmark-estimation/

[11] https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

[12] https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

[13] https://machinelearningmastery.com/neural-network-models-for-combined-classification-and-regression/

[14] https://www.kaggle.com/yasinsoylu123/asl-recognition-keras-cnn/notebook

[15] https://www.datacamp.com/community/tutorials/random-forests-classifier-python

[16] https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

[17] UI Reference: https://github.com/sid-1998/Sign-Language-Recognition/blob/master/Gesture$_R ecognize_sign.py$