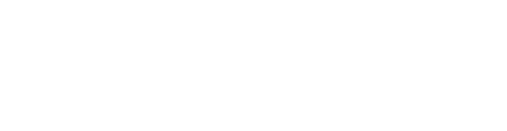
ARTIFICIAL NEURAL NETWORK CS-307



**Project Report**

Sign Language Recognition

DEPARTMENT OF COMPUTER SCIENCE, UET TAXILA

**Table of Contents**

[**Abstract:** 2](#_Toc14311)

[**Introduction:** 3](#_Toc14312)

[**Literature:** 4](#_Toc14313)

[**Methodology:** 5](#_Toc14314)

[**Dataset:** 5](#_Toc14315)

[**Data Processing:** 6](#_Toc14316)

[**Architecture:** 6](#_Toc14317)

[**Realtime System Evaluation:** 9](#_Toc14318)

[**Conclusion:** 14](#_Toc14319)

[**References** 15](#_Toc14320)

# Abstract:

The listening or hearing impaired (deaf/dumb) people use a set of signs, called sign language instead of speech for communication among them. However, it is very challenging for nonsign language speakers to communicate with this community using signs. It is very necessary to develop an application to recognize gestures or actions of sign languages to make easy communication between the normal and the deaf community. Though American sign language (ASL) has gained recognition from the American society, few ASL applications have been developed with educational purposes. Those designed with real-time sign recognition systems are also lacking. Leap motion controller facilitates the real-time and accurate recognition of ASL signs. It allows an opportunity for designing a learning application with a real-time sign recognition system that seeks to improve the effectiveness of ASL learning. The project proposes an ASL learning application prototype. The application would be a whack-a-mole game with a real-time sign recognition system embedded. Since both static and dynamic signs (J, Z) exist in ASL alphabets, Long-Short Term Memory Recurrent Neural Network with kNearest-Neighbour method is adopted as the classification method is based on handling of sequences of input. Characteristics such as sphere radius, angles between fingers and distance between finger positions are extracted as input for the classification model. The model is trained with 2600 samples, 100 samples taken for each alphabet. The experimental results revealed that the recognition rate for 26 ASL alphabets yields an average of 99.44% accuracy rate and 91.82% in 5-fold cross-validation with the use of leap motion controller. In this we are using CCN for the American Sign Language (ASL) recognition. The model using classic MNIST trained on 27,455 cases and test on 7172 cases for the static signs excluding J and Z because of gesture motion (3D).

# Introduction:

Communication is an essential tool in human existence. It is a fundamental and effective way of sharing thoughts, feelings and opinions. However, a substantial fraction of the world's population lacks this ability. Many people are suffering from hearing loss, speaking impairment or both. A partial or complete inability to hear in one or both ears is known as hearing loss. On the other hand, mute is a disability that impairs speaking and makes the affected people unable to speak. If deaf-mute happens during childhood, their language learning ability can be hindered and results in language impairment, also known as hearing mutism. These ailments are part of the most common disabilities worldwide. Statistical report of physically challenged children during the past decade reveals an increase in the number of neonates born with a defect of hearing impairment and creates a communication barrier between them and the rest of the world.

According to the World Health Organization (WHO) report, the number of people affected by hearing disability in 2005 was approximately 278 million worldwide. Ten (10) years later, this number jumped to 360 million, a roughly 14% increment. Since then, the number has been increasing exponentially. The latest report of WHO revealed that 466 million people were suffering from hearing loss in 2019, which amount to 5% of the world population with 432 million (or 83%) of them being adults, and 34 million (17%) of them are children.

The WHO also estimated that the number would double (i.e. 900 million people) by 2050. In these fast-growing deaf-mute people, there is a need to break the communication barrier that adversely affects the lives and social relationships of deaf-mute people.

However, an evident solution to this issue is present in the world of Machine Learning and Image Detection. Implementing predictive model technology to automatically classify Sign Language symbols can be used to create a form of real-time captioning for virtual conferences like Zoom meetings and other such things. This would greatly increase access of such services to those with hearing impairments as it would go hand-in-hand with voice-based captioning, creating a two-way communication system online for people with hearing issues.

Moreover, sometimes combinations of hand mouth facial expressions are used to communicate. There are mainly three variants of sign language. They are:

1. Non-manual features: Tongue, facial expression, body poses, and hand gestures - all of them are used to communicate.
2. Word level sign spelling: Each gesture represents a whole word.
3. Finger vocabulary: One gesture represents one alphabet/numbers.

In this paper, we are using Finger vocabulary ASL [1] data set for the experiment. There are mainly two types of approaches that can be seen in current literature. One is using a special device to capture the gesture and recognize it. The second approach is to use deep learning to recognize the gesture that the hands are making. The downside of this approach is that the design and the computation for these deep learning models are quite expensive and the special device tools are not so cheap and available in the current market that they can be used by mass people daily.

We are using Keras, as Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. TensorFlow is a free and open-source software library for machine learning and artificial intelligence.

# Literature:

Literature review of the problem shows that there have been several approaches to address the issue of gesture recognition in video using several different methods. Past research has suggested several methods for the recognition of ASL, including the usage of motion gloves, Kinect Sensor, image processing with cameras and leap motion controllers.

In one [2] the authors used Hidden Markov Models (HMM) to recognize facial expressions from video sequences combined with Bayesian Network Classifiers and Gaussian Tree Augmented Naive Bayes Classifier.

In another paper by Deep Kothadiya [3], deep learning-based model that detects and recognizes the words from a person’s gestures. LSTM and GRU, are used to recognize signs from isolated Indian Sign Language (ISL) video frames. The proposed model, consisting of a single layer of LSTM followed by GRU, achieves around 97% accuracy over 11 different signs.

Francois et al. [4] also published a paper on Human Posture Recognition in a Video Sequence using methods based on 2D and 3D appearance. The work mentions using PCA to recognize silhouettes from a static camera and then using 3D to model posture for recognition. This approach has the drawback of having intermediary gestures which may lead to ambiguity in training and therefore a lower accuracy in prediction.

C.K.M. Lee [5] with fellow researchers published a paper on American sign language recognition and training method with recurrent neural network. Since both static and dynamic signs (J, Z) exist in ASL alphabets, Long-Short Term Memory Recurrent Neural Network with k-Nearest-Neighbour method is adopted as the classification method is based on handling of sequences of input. Characteristics such as sphere radius, angles between fingers and distance between finger positions are extracted as input for the classification model.

The model is trained with 2600 samples, 100 samples taken for each alphabet. The experimental results revealed that the recognition rate for 26 ASL alphabets yields an average of 99.44% accuracy rate and 91.82% in 5-fold cross-validation with the use of leap motion controller.

Similar work by Kumud et al. [6] defines how to do Continuous Indian Sign Language Recognition. The paper proposes frame extraction from video data, pre-processing the data, extracting key frames from the data followed by extracting other features, recognition and finally optimization.

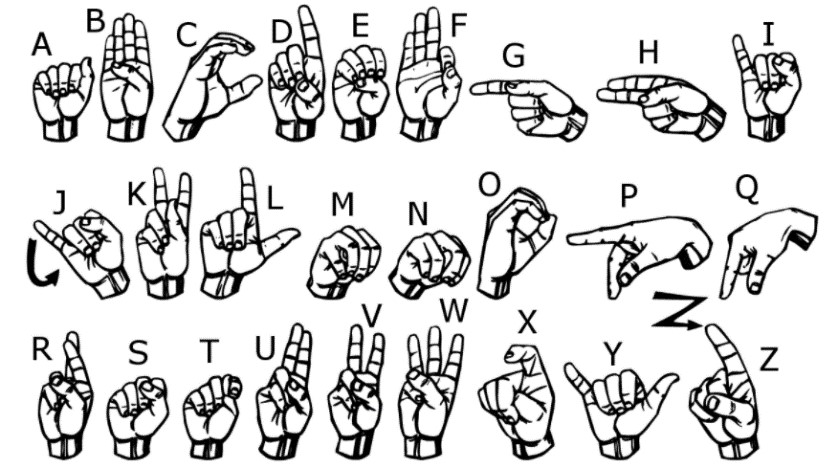
Pre-processing is done by converting the video to a sequence of RGB frames. Each frame having the same dimensions. Skin color segmentation used to extract skin region, with the help of HSV. The images obtained were converted to binary form. The key frames were extracted by calculating a gradient between the frames. And the features were extracted from the key frames using oriental histogram. Classification was done by Euclidean Distance, Manhattan Distance, Chess Board Distance and Mahalanobis Distance.

# Methodology:

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing, and is used by many hearing people as well.

## **Dataset:**

The dataset format is patterned to match closely with the classic MNIST [1]. 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). As in the figure below



The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1, pixel2 …. pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255. The original hand gesture image data represented multiple users repeating the gesture against different backgrounds. The Sign Language MNIST data came from greatly extending the small number (1704) of the color images included as not cropped around the hand region of interest. To create new data, an image pipeline was used based on ImageMagick and included cropping to hands-only, Gray-scaling, resizing, and then creating at least 50+ variations to enlarge the quantity.



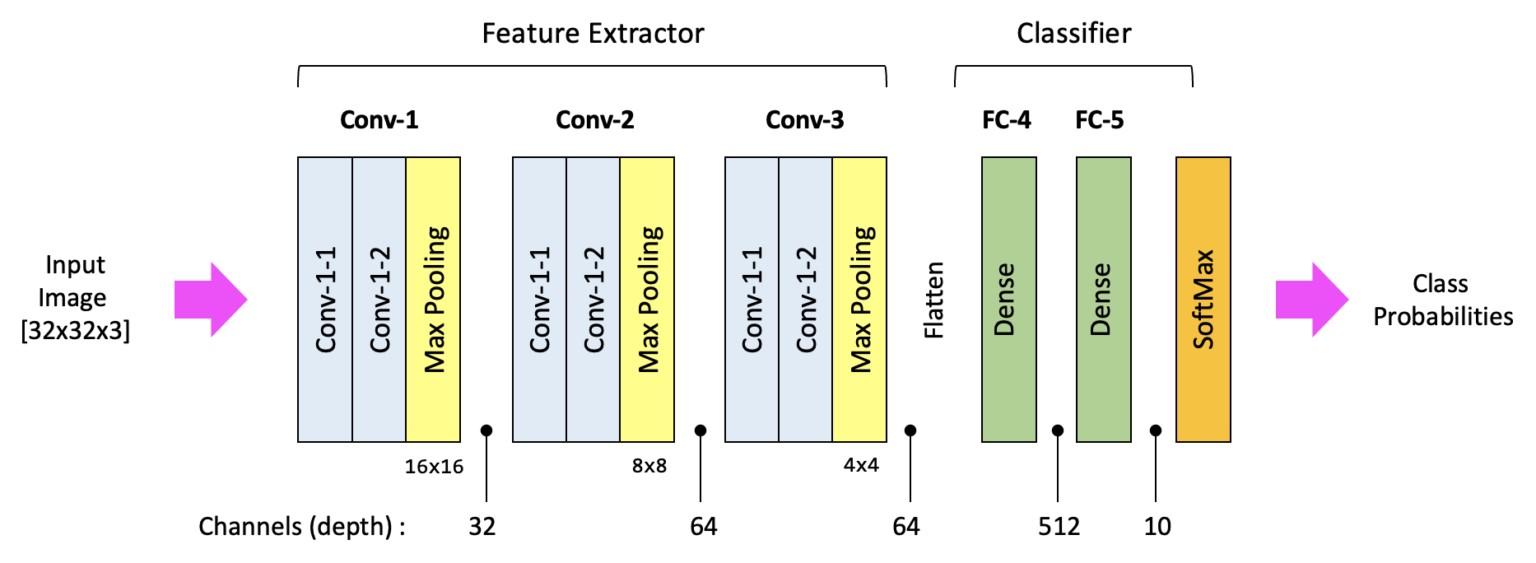
## **Data Processing:**

In order to avoid overfitting problem, we need to expand artificially our dataset. We can make your existing dataset even larger. The idea is to alter the training data with small transformations to reproduce the variations. Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more. By applying just, a couple of these transformations to our training data, we can easily double or triple the number of training examples and create a very robust model.

## **Architecture:**

After processing the images, the CNN model must be compiled to recognize all of the classes of information being used in the data. Normalization of the data must also be added to the data, equally balancing the classes with less images. We are using Keras as Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. It was developed by the Google Brain team for Google's internal use in research and production.

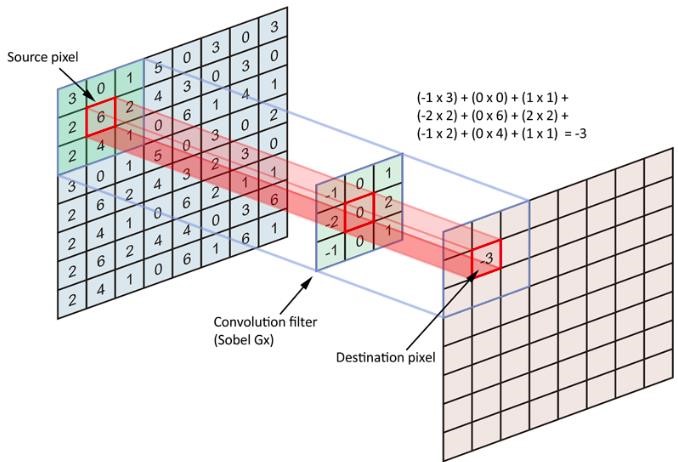
Implementation of CNN in Keras and TensorFlow, as show in figure

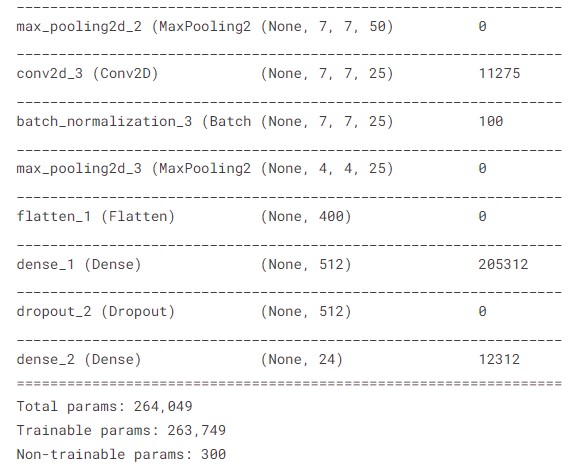
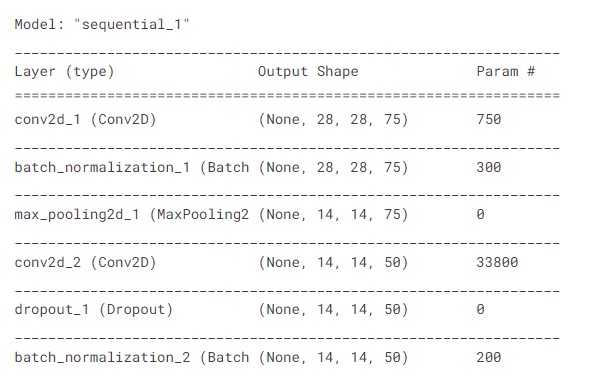


Designed to enable fast experimentation with deep neural networks, Keras focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer is François Chollet, a Google engineer. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

A Convolutional Neural Network is a special type of an Artificial Intelligence implementation which uses a special mathematical matrix manipulation called the convolution operation to process data from the images.

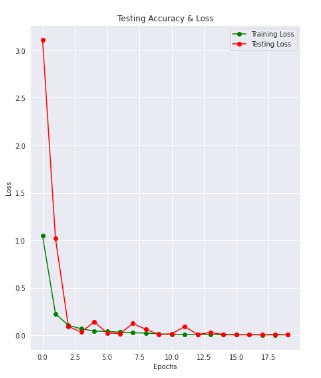
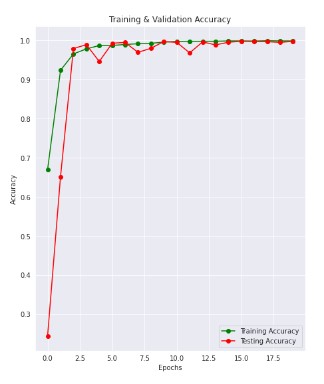
* A convolution does this by multiplying two matrices and yielding a third, smaller matrix.
* The Network takes an input image, and uses a filter (or kernel) to create a feature map describing the image.
* In the convolution operation, we take a filter (usually 2x2 or 3x3 matrix) and slide it over the image matrix. The corresponding numbers in both matrices are multiplied and added to yield a single number describing that input space. This process is repeated all over the image. This working can be seen in the following figure





The summary of the model is given below, we achieved the accuracy from 65% to on average about 98% in the final run. Below figures are the model

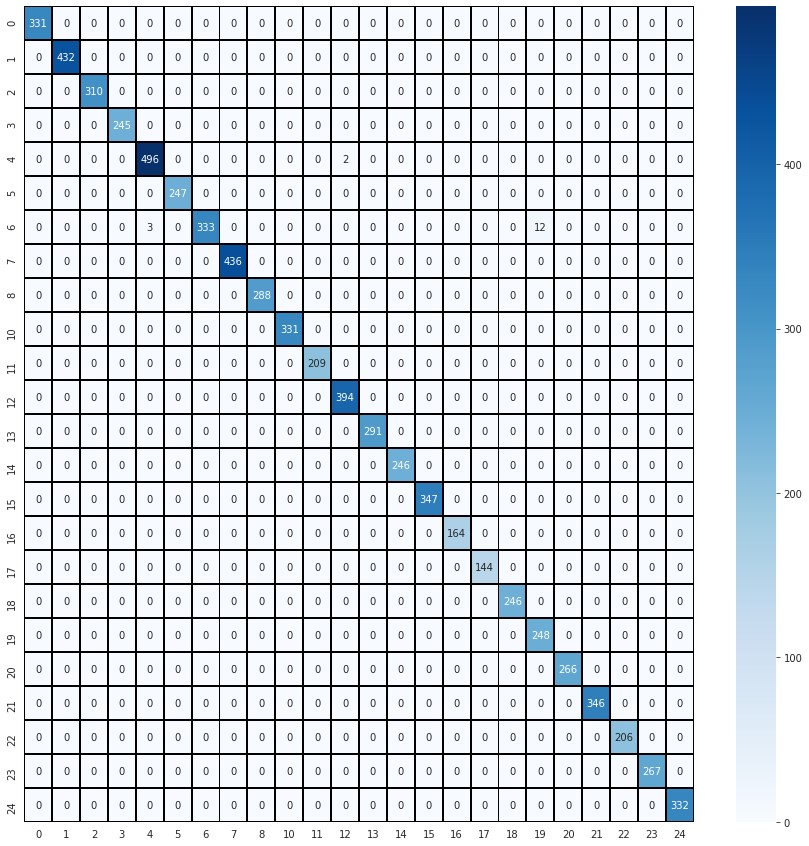
summary and evaluation graphs;



From the confusion matrix, it can be seen that the models can label maximum of these data quite correctly. The confidence value of the gesture it recognizes is quite high also.

Accuracy of the model is 99.76296424865723 % (as tested and give by the model)

(and no cases for J or Z because of gesture motions)



The model developed accurately detects and classifies Sign Language symbols with about 95% training accuracy.

Now the model is trained and tested and have 90% + accuracy rate. Its time to develop and test the model on real application using webcam input and run this developed model on real time video stream.

# Realtime System Evaluation:

Now, using two popular live video processing libraries known as Mediapipe and Open-CV, we can take webcam input and run our previously developed model on real time video stream.

*import os*

*os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3' import tensorflow as tf import cv2 import mediapipe as mp from keras.models import load\_model import numpy as np import time*

*model = load\_model('smnist.h5')*

*mphands = mp.solutions.hands hands = mphands.Hands()*

*mp\_drawing = mp.solutions.drawing\_utils*

*cap = cv2.VideoCapture(0) \_, frame = cap.read() h, w, c = frame.shape*

*analysisframe = ''*

*letterpred = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y']*

*while True:*

*\_, frame = cap.read()*

*k = cv2.waitKey(1) if k%256 == 27: # ESC pressed print("Escape hit, closing...")*

*break*

*framergb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) result = hands.process(framergb) hand\_landmarks = result.multi\_hand\_landmarks if hand\_landmarks: for handLMs in hand\_landmarks:*

*x\_max = 0 y\_max = 0 x\_min = w y\_min = h for lm in handLMs.landmark:*

*x, y = int(lm.x \* w), int(lm.y \* h) if x > x\_max: x\_max = x if x < x\_min: x\_min = x if y > y\_max: y\_max = y if y < y\_min: y\_min = y y\_min -= 20 y\_max += 20 x\_min -= 20 x\_max += 20 cv2.rectangle(frame, (x\_min, y\_min), (x\_max, y\_max), (0, 255, 0), 2) mp\_drawing.draw\_landmarks(frame, handLMs, mphands.HAND\_CONNECTIONS) cv2.imshow("Frame", frame)*

*cap.release() cv2.destroyAllWindows()*

The second to last part of the program is capturing a single frame on cue, cropping it to the dimensions of the bounding box.

*while True:*

*\_, frame = cap.read()*

*k = cv2.waitKey(1) if k%256 == 27: # ESC pressed print("Escape hit, closing...") break elif k%256 == 32:*

*# SPACE pressed # SPACE pressed analysisframe = frame showframe = analysisframe cv2.imshow("Frame", showframe) framergbanalysis = cv2.cvtColor(analysisframe, cv2.COLOR\_BGR2RGB) resultanalysis = hands.process(framergbanalysis) hand\_landmarksanalysis = resultanalysis.multi\_hand\_landmarks if hand\_landmarksanalysis:*

*for handLMsanalysis in hand\_landmarksanalysis:*

*x\_max = 0 y\_max = 0 x\_min = w y\_min = h for lmanalysis in handLMsanalysis.landmark:*

*x, y = int(lmanalysis.x \* w), int(lmanalysis.y \* h) if x > x\_max: x\_max = x if x < x\_min: x\_min = x if y > y\_max:*

*y\_max = y if y < y\_min: y\_min = y y\_min -= 20 y\_max += 20 x\_min -= 20 x\_max += 20*

*analysisframe = cv2.cvtColor(analysisframe, cv2.COLOR\_BGR2GRAY) analysisframe = analysisframe[y\_min:y\_max, x\_min:x\_max] analysisframe = cv2.resize(analysisframe,(28,28))*

*nlist = [] rows,cols = analysisframe.shape for i in range(rows): for j in range(cols): k = analysisframe[i,j] nlist.append(k)*

*datan = pd.DataFrame(nlist).T colname = [] for val in range(784): colname.append(val) datan.columns = colname*

*pixeldata = datan.values pixeldata = pixeldata / 255 pixeldata = pixeldata.reshape(-1,28,28,1)*

This code looks very similar to the last portion of the program. This is due mainly to the fact that the process involving the production of the bounding box is the same in both parts. However, in this analysis section of the code, we make use of the image reshaping feature from

Open-CV to resize the image to the dimensions of the bounding box, rather than creating a visual object around it. Along with this, we also use NumPy and Open-CV to modify the image to have the same characteristics as the images the model was trained on. We also use pandas to create a data frame with the pixel data from the images saved, so we can normalize the data in the same way we did for the model creation.

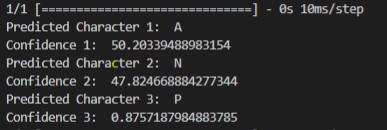
Finally, we need to run the trained model on the processed image and process the information output.

*prediction = model.predict(pixeldata) predarray = np.array(prediction[0]) letter\_prediction\_dict = {letterpred[i]: predarray[i] for i in range(len(letterpred))} predarrayordered = sorted(predarray, reverse=True) high1 = predarrayordered[0] high2 = predarrayordered[1] high3 = predarrayordered[2] for key,value in letter\_prediction\_dict.items(): if value==high1:*

*print("Predicted Character 1: ", key) print('Confidence 1: ', 100\*value) elif value==high2:*

*print("Predicted Character 2: ", key) print('Confidence 2: ', 100\*value) elif value==high3:*

*print("Predicted Character 3: ", key) print('Confidence 3: ', 100\*value) time.sleep(5)*



As shown, the model accurately predicts the character being shown from the camera. Along with the Predicted Character, the program also displays the confidence of the classification from the CNN Keras model.

# Conclusion:

In this paper, a sign language recognition system is designed and developed using deep learning approach. The overall procedure of developing this recognition system from training the data using CNN approach to sign language recognition is described. It is verified that with the large number of data samples being used to train a class the model can achieve accuracy of 99% in recognising sign language (as shown in graphs above) and in real-time video, there are few factors that can affect the accuracy of the system. When the light intensity is insufficient, the accuracy is relatively low compared to higher light intensity. Other than that, model is the main element in the recognition process. The model trained with large set of data, better it performs. The images that are used to train the classifier must be in variety of conditions in order to generate a robust model.

# References

1. "sign-language-mnist," [Online]. Available: https://www.kaggle.com/datasets/datamunge/signlanguage-mnist.
2. I. Cohen, N. Sebe, A. Garg, L. S. Chen and T. S. Huang, "Facial expression recognition from video sequences: temporal and static modeling," *Computer Vision and Image Undertaking,* vol. 91.
3. D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A.-B. Gil-González and J. M. Corchado, "Deepsign: Sign Language Detection and Recognition Using Deep Learning," *Electronics,* vol. 11, no. 11, 2022.
4. B. Boulay, F. Bremond and M. Thonat, "Human Posture Recognition in Video Sequence," *IEEE International Workshop on VSPETS (Visual Surveillance and Performance Evaluation of Tracking and Surveillance),* 2003.
5. C. Lee, K. K. Ng, C.-H. Chen, H. Lau, S. Chung and T. Tsoi, "American sign language recognition and training method with recurrent," *Expert Systems With Applications,* 2021.
6. K. Tripathi, N. Baranwal and G. C. Nandi, "Continuous dynamic Indian Sign Language gesture recognition with invariant backgrounds by Kumud Tripathi," *Conference on Advances in Computing Communications and Informatics (ICACCI),* 2015.