INDIAN SIGN LANGUAGE RECOGNITION

A Summer Project Report submitted

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

by

ASHISH SINGLA

Roll No.: 00520902719

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

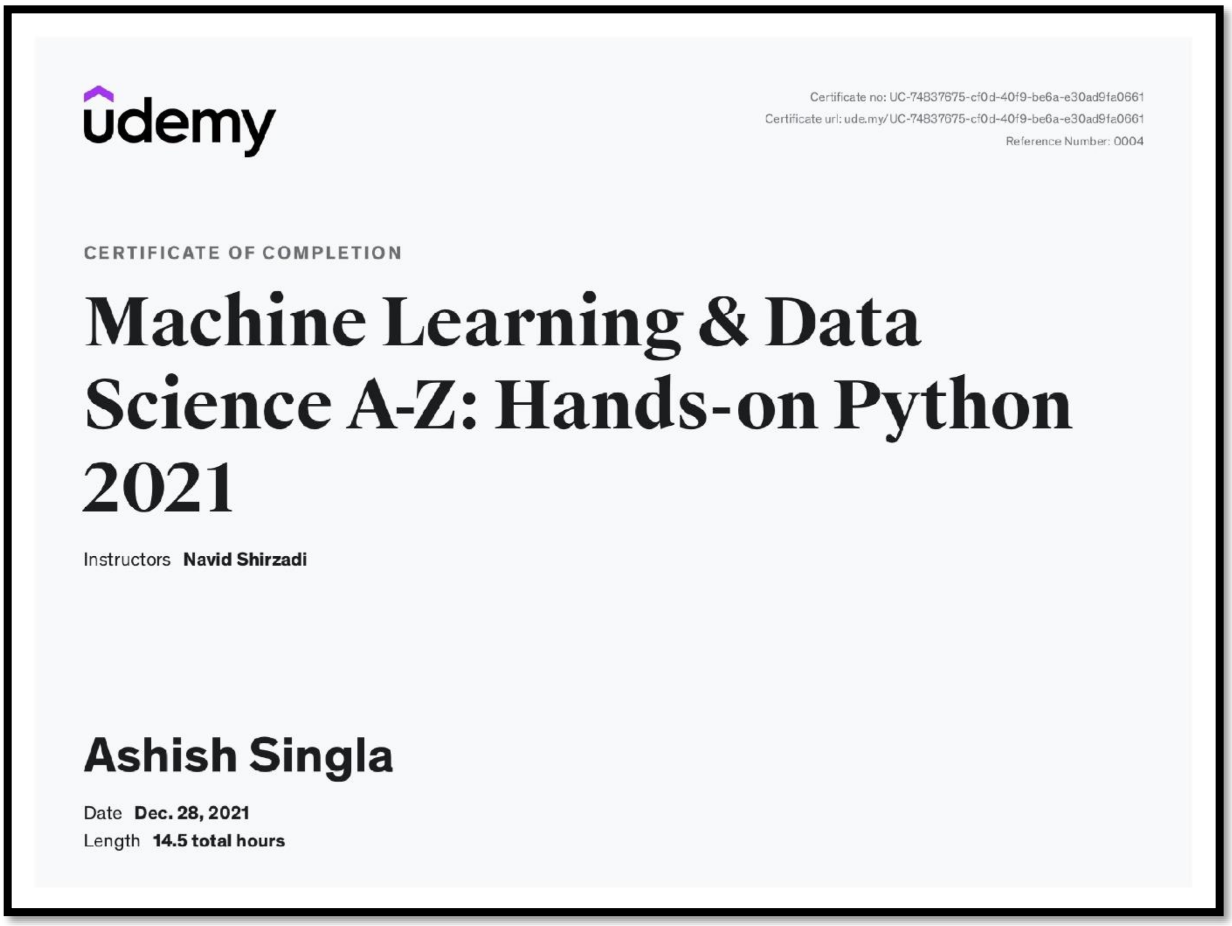
G.B. PANT GOVT. ENGINEERING COLLEGE, NEW DELHI.

(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)

JANUARY, 2022

TABLE OF CONTENTS

CERTIFICATE



udemy

Certificate no: UC-74837675-cf0d-40f9-be6a-e30ad9fa0661
Certificate url: ude.my/UC-74837675-cf0d-40f9-be6a-e30ad9fa0661
Reference Number: 0004

CERTIFICATE OF COMPLETION

Machine Learning & Data Science A-Z: Hands-on Python 2021

Instructors **Navid Shirzadi**

Ashish Singla

Date **Dec. 28, 2021**
Length **14.5 total hours**

CANDIDATE’S DECLARATION

I hereby certify that the work which is being presented in the work entitled **“Indian Sign Language Recognition”** in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering affiliated to **Guru Gobind Singh Indraprastha University, New Delhi** and submitted to the Department of Computer Science and Engineering, G.B.Pant Govt. Engineering College, is an authentic record of my own work carried out during the academic session **2021 - 2022.** The matter represented in this report has not been submitted by me for award of any other degree of this or any other institute/university.

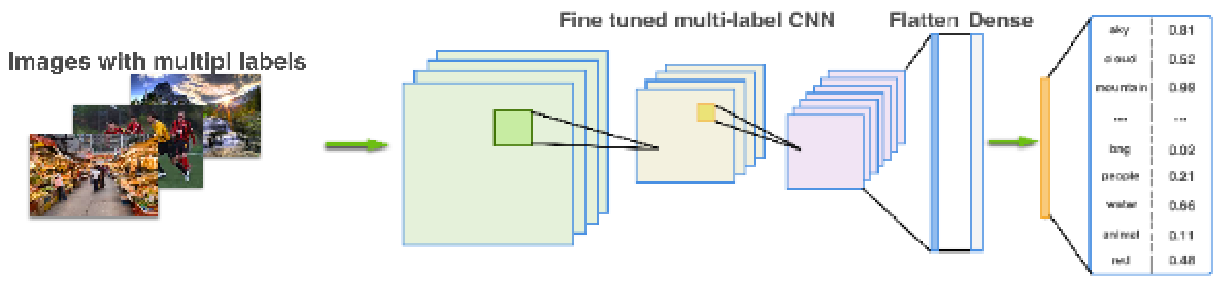
Date: January, 2022 **Name: Ashish Singla**

Roll No.: 00520902719

ABSTRACT

Communication is a basic prerequisite for an individual to express feelings, thoughts, and ideas, yet the vast majority of individuals find it difficult to interact with the hearing and speech challenged community. Hearing and speech disabled people can engage with the rest of society through sign language. The Indian government has passed the Rights of Persons with Disabilities (RPWD) Act, 2016, which acknowledges Indian Sign Language (ISL) and requires the use of sign language interpreters in all government-aided organizations and public sector proceedings. Unfortunately, a large percentage of the Indian population is not familiar with the semantics of the gestures associated with ISL. To bridge this communication gap, this project develops a model that uses Convolutional Neural Networks to detect and classify Indian Sign Language.

The model seeks to categories 36 ISL gestures representing 0-9 digits and A-Z alphabets by converting them to their text equivalents using OpenCV and Keras implementation of CNNs. The developed and used dataset consists of 350 photos for each gesture, which were input into the CNN model for training and testing. The proposed model was successfully implemented and achieved 99.03% accuracy for the test images.



1. INTRODUCTION

Based on research works conducted by various colleges and universities across India, approximately 6.4 per cent of India’s population suffers from some level of hearing loss and disabilities. Nearly 7 million Indians are deaf or dumb, and 1.9 million suffer from some type of speech disability according to the 2011 census. Many children, especially in developing countries such as India, lack access to proper diagnosis and follow-up infrastructure and are thus doomed to a life of frustration and unrealized potential.

Sign language is a means of communication that conveys a message through visual cues such as hand gestures, body movements, and facial expressions. It is one of the most widely used visual means of exchanging thoughts among deaf and dumb people. But a very small portion of the Indian population is able to communicate through Sign Language. This factor does not play in favour of people suffering from disabilities as they are denied almost all opportunities and have to live a life of hardships.

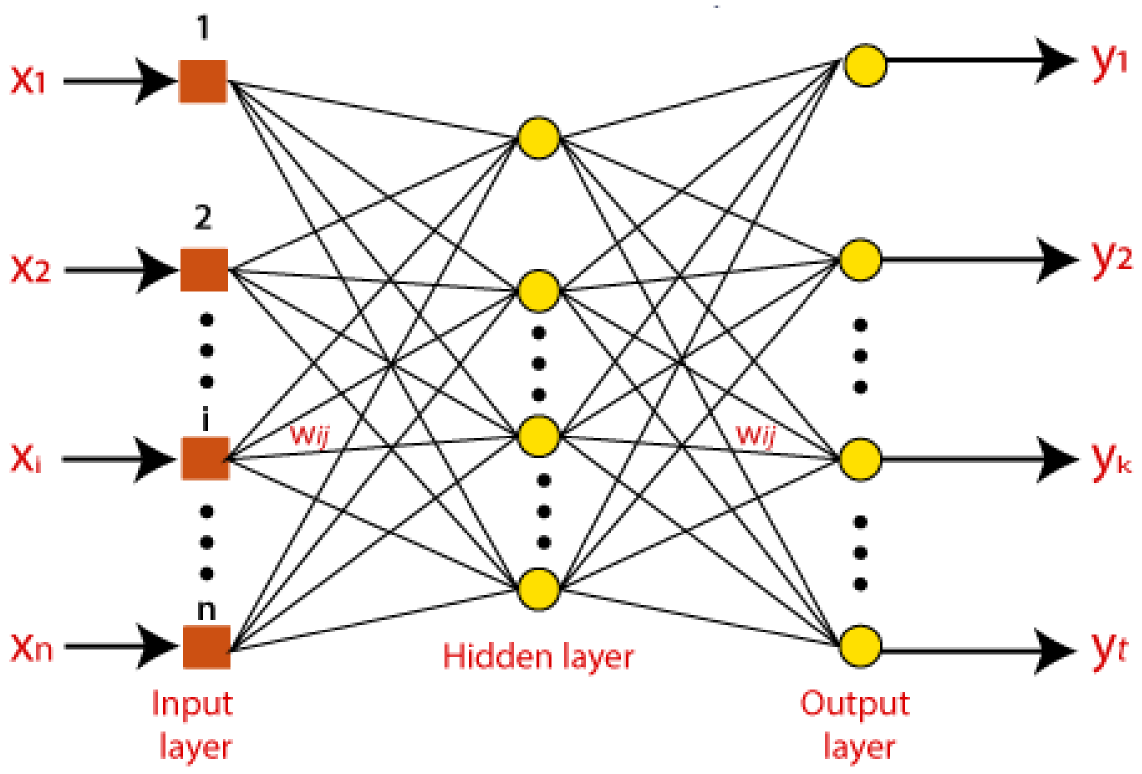
All over the world, different sign languages like Indian sign language (ISL), American Sign Language (ASL), Chinese sign language (CSL), etc. are being used to communicate with the deaf and dumb people of our society. Indian Sign Language, unlike other sign languages, is a two-hand gesture sign language, though some of the letters of the alphabet (C, G, I, L, O, U, V) and all the numbers (0-9) only require one hand representation.

Therefore, addressing the issue, this project implements the application of Convolutional Neural Networks and OpenCV to recognize and classify 36 different ISL hand gestures i.e., alphabets from A-Z, and numbers from 0-9.

2. THEORY

2.1. NEURAL NETWORKS

Neural networks are a class of machine learning algorithms used to model complex patterns in datasets using multiple hidden layers and non-linear activation functions. A neural network takes an input, passes it through multiple layers of hidden neurons (mini-functions with unique coefficients that must be learned), and outputs a prediction representing the combined input of all the neurons.



Neural networks are artificial systems that were inspired by biological neural networks. These systems learn to perform tasks by being exposed to various datasets and examples without any task-specific rules. The idea is that the system generates identifying characteristics from the data they have been passed without being programmed with a pre-programmed understanding of these datasets.

Neural networks are based on computational models for threshold logic. Threshold logic is a combination of algorithms and mathematics. Neural networks are based either on the study of the brain or on the application of neural networks to artificial intelligence.

These networks are trained iteratively using optimization techniques like gradient descent. After each cycle of training, an error metric is calculated based on the difference between prediction and target. The derivatives of this error metric are calculated and propagated back through the network using a technique called backpropagation. Each neuron's coefficients (weights) are then adjusted relative to how much they contributed to the total error. This process is repeated iteratively until the network error drops below an acceptable threshold.

2.2. MACHINE LEARNING

Machine Learning is the process of creating models that can perform a certain task without the need for a human explicitly programming it to do something.

There are 3 types of Machine Learning which are based on the way the algorithms are created. They are:

- **Supervised Learning –** You supervise the learning process, meaning the data that you have collected here is labelled and so you know what input needs to be mapped to what output. This helps you correct your algorithm if it makes a mistake in giving you the answer.

- **Unsupervised Learning –** The data collected here has no labels and you are unsure about the outputs. So, you model your algorithm such that it can understand patterns from the data and output the required answer. You do not interfere when the algorithm learns.

- **Reinforcement Learning –** There is no data in this kind of learning, nor do you teach the algorithm anything. You model the algorithm such that it interacts with the environment and if the algorithm does a good job, you reward it, else you punish the algorithm. With continuous interactions and learning, it goes from being bad to being the best that it can for the problem assigned to it.

2.2.1. SUPERVISED LEARNING

Supervised Learning is the process of making an algorithm to learn to map an input to a particular output. This is achieved using the labelled datasets that you have collected. If the mapping is correct, the algorithm has successfully learned.

Else, you make the necessary changes to the algorithm so that it can learn correctly. Supervised Learning algorithms can help make predictions for new unseen data that we obtain later in the future.

This is similar to a teacher-student scenario. There is a teacher who guides the student to learn from books and other materials. The student is then tested and if correct, the student passes. Else, the teacher tunes the student and makes the student learn from the mistakes that he or she had made in the past. That is the basic principle of Supervised Learning.

2.2.1.1. TYPES OF SUPERVISED LEARNING

Supervised Learning has been broadly classified into 2 types:

- Regression
- Classification

2.2.1.1.1. REGRESSION

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2.2.1.1.2. CLASSIFICATION

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Below are some popular Classification algorithms which come under supervised learning:

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

2.2.1.2. ADVANTAGES OF SUPERVISED LEARNING

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as fraud detection, spam filtering, etc.

2.2.1.3. DISADVANTAGES OF SUPERVISED LEARNING

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

2.2.2. UNSUPERVISED LEARNING

Unsupervised Learning, as discussed earlier, can be thought of as self-learning where the algorithm can find previously unknown patterns in datasets that do not have any sort of labels. It helps in modelling probability density functions, finding anomalies in the data, and much more. To give you a simple example, think of a student who has textbooks and all the required material to study but has no teacher to guide. Ultimately, the student will have to learn by himself or herself to pass the exams. This sort of self-learning is what we have scaled into Unsupervised Learning for machines.

2.2.2.1. TYPES OF UNSUPERVISED LEARNING

Unsupervised Learning has been split up majorly into 2 types:

- Clustering
- Association

2.2.2.1.1. CLUSTERING

Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

2.2.2.1.2. ASSOCIATION

An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

2.2.2.2. UNSUPERVISED LEARNING ALGORITHMS

Some common unsupervised learning algorithms are:

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

2.2.2.3. ADVANTAGES OF UNSUPERVISED LEARNING

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

2.2.2.4. DISADVANTAGES OF UNSUPERVISED LEARNING

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

2.3. EVOLUTION OF NEURAL NETWORKS

Hebbian learning deals with neural plasticity. Hebbian learning is unsupervised and deals with long term potentiation. Hebbian learning deals with pattern recognition and exclusive-or circuits; deals with if-then rules.

Back propagation solved the exclusive-or issue that Hebbian learning could not handle. This also allowed for multi-layer networks to be feasible and efficient. If an error was found, the error was solved at each layer by modifying the weights at each node. This led to the development of support vector machines, linear classifiers, and max-pooling. The vanishing gradient problem affects feedforward networks that use back propagation and recurrent neural network. This is known as deep-learning.

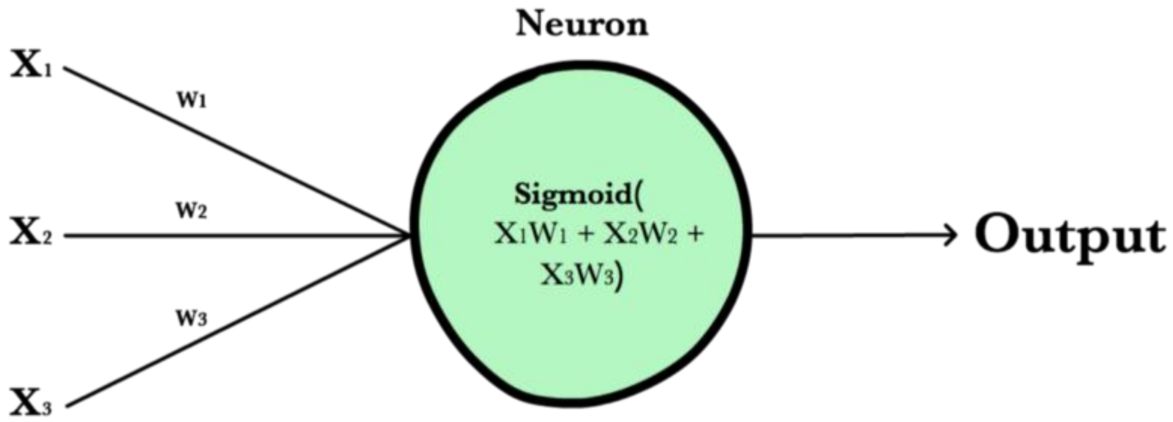
Hardware-based designs are used for biophysical simulation and neurotrophic computing. They have large scale component analysis and convolution creates new class of neural computing with analog. This also solved back-propagation for many-layered feedforward neural networks.

Convolutional networks are used for alternating between convolutional layers and max-pooling layers with connected layers (fully or sparsely connected) with a final classification layer. The learning is done without unsupervised pre-training. Each filter is equivalent to a weights vector that has to be trained. The shift variance has to be guaranteed to dealing with small and large neural networks. This is being resolved in Development Networks.

2.4. NEURON

A "neuron" in an artificial neural network is a mathematical approximation of a biological neuron. It takes a vector of inputs, performs a transformation on them, and outputs a single scalar value. It can be thought of as a filter.

A neuron takes a group of weighted inputs, applies an activation function, and returns an output.



Inputs to a neuron can either be features from a training set or outputs from a previous layer’s neurons. Weights are applied to the inputs as they travel along synapses to reach the neuron. The neuron then applies an activation function to the “sum of weighted inputs” from each incoming synapse and passes the result on to all the neurons in the next layer.

2.5. SYNAPSE

Synapses are like roads in a neural network. They connect inputs to neurons, neurons to neurons, and neurons to outputs. In order to get from one neuron to another, you have to travel along the synapse paying the “toll” (weight) along the way. Each connection between two neurons has a unique synapse with a unique weight attached to it. When we talk about updating weights in a network, we’re really talking about adjusting the weights on these synapses.

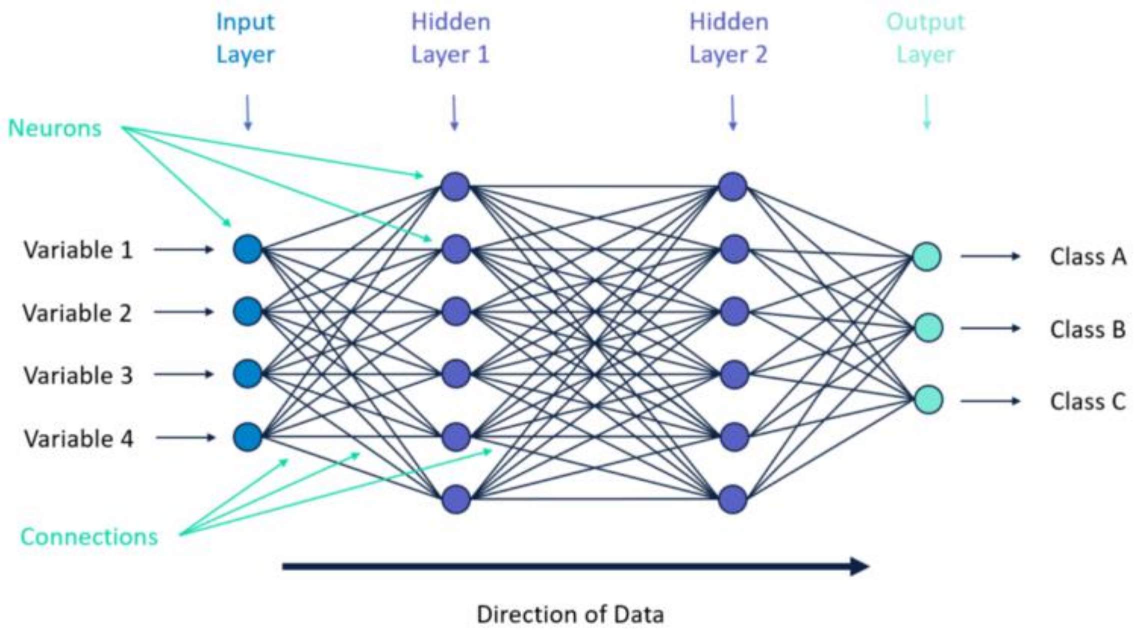
2.6. WEIGHTS

Weights are values that control the strength of the connection between two neurons. That is, inputs are typically multiplied by weights, and that defines how much influence the input will have on the output. In other words: when the inputs are transmitted between neurons, the weights are applied to the inputs along with an additional value (the bias).

2.7. BIAS

Bias terms are additional constants attached to neurons and added to the weighted input before the activation function is applied. Bias terms help models represent patterns that do not necessarily pass through the origin. For example, if all your features were 0, would your output also be zero? Is it possible there is some base value upon which your features have an effect? Bias terms typically accompany weights and must also be learned by your model.

2.8. LAYERS



There are three types of layers in neural networks:

- **Input Layer:** Holds the data your model will train on. Each neuron in the input layer represents a unique attribute in your dataset (e.g., height, hair color, etc.).
- **Hidden Layer:** Sits between the input and output layers and applies an activation function before passing on the results. There are often multiple hidden layers in a network. In traditional networks, hidden layers are typically fully-connected layers - each neuron receives input from all the previous layer’s neurons and sends its output to every neuron in the next layer. This contrasts with how convolutional layers work where the neurons send their output to only some of the neurons in the next layer.

- **Output Layer:** The final layer in a network. It receives input from the previous hidden layer, optionally applies an activation function, and returns an output representing your model's prediction.

2.9. WEIGHTED INPUT

A neuron's input equals the sum of weighted outputs from all neurons in the previous layer. Each input is multiplied by the weight associated with the synapse connecting the input to the current neuron. If there are 3 inputs or neurons in the previous layer, each neuron in the current layer will have 3 distinct weights—one for each synapse.

Single Input

$$\begin{aligned} Z &= Input \cdot Weight \\ &= XW \end{aligned}$$

Multiple Inputs

$$\begin{aligned} Z &= \sum_{i=1}^{n} x_i w_i \\ &= x_1 w_1 + x_2 w_2 + x_3 w_3 \end{aligned}$$

2.10. ACTIVATION FUNCTIONS

Activation functions live inside neural network layers and modify the data they receive before passing it to the next layer. Activation functions give neural networks their power—allowing them to model complex non-linear relationships. By modifying inputs with non-linear functions neural networks can model highly complex relationships between features. Popular activation functions include relu and sigmoid.

Activation functions typically have the following properties:

- **Non-linear** - In linear regression we're limited to a prediction equation that looks like a straight line. This is nice for simple datasets with a one-to-one relationship between inputs and outputs, but what if the patterns in our dataset were non-linear? (e.g., x2, sin, log). To model these relationships, we need a non-linear prediction equation. Activation functions provide this non-linearity.

- **Continuously differentiable -** To improve our model with gradient descent, we need our output to have a nice slope so we can compute error derivatives with respect to weights. If our neuron instead outputted 0 or 1 (perceptron), we wouldn't know in which direction to update our weights to reduce our error.
- **Fixed Range -** Activation functions typically squash the input data into a narrow range that makes training the model more stable and efficient.

2.11. LOSS FUNCTIONS

A loss function, or cost function, is a wrapper around our model's predict function that tells us "How good" the model is at making predictions for a given set of parameters. The loss function has its own curve and its own derivatives. The slope of this curve tells us how to change our parameters to make the model more accurate! We use the model to make predictions. We use the cost function to update our parameters. Our cost function can take a variety of forms as there are many different cost functions available. Popular loss functions include: MSE (L2) and Cross-entropy Loss.

2.12. TYPES OF NEURAL NETWORKS

There are several types of neural networks that can be used:

- The first is a multilayer perceptron which has three or more layers and uses a nonlinear activation function.
- The second is the convolutional neural network that uses a variation of the multilayer perceptron.
- The third is the recursive neural network that uses weights to make structured predictions.
- The fourth is a recurrent neural network that makes connections between the neurons in a directed cycle. The long short-term memory neural network uses the recurrent neural network architecture and does not use activation function.
- The final two are sequence to sequence modules which uses two recurrent networks and shallow neural networks which produces a vector space from an amount of text.

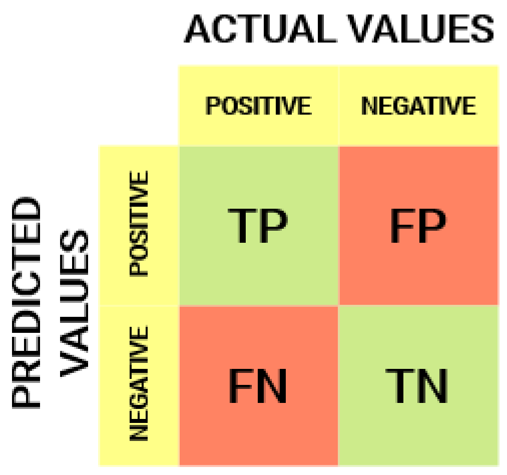
2.13. LIMITATIONS

The neural network is for a supervised model. It does not handle unsupervised machine learning and does not cluster and associate data. It also lacks a level of accuracy that will be found in more computationally expensive neural network. Based on Andrew Trask's neural network. Also, the neural network does not work with any matrices where X's number of rows and columns do not match Y and W's number of rows.

2.14. CONFUSION MATRIX

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

Some terms related to confusion matrix:

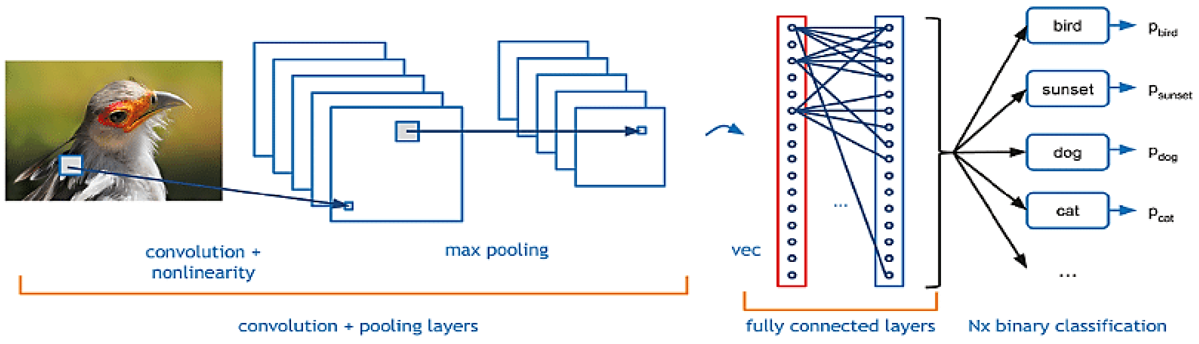
- **True Positive (TP)**
  - The predicted value matches the actual value

    - The actual value was positive and the model predicted a positive value
- **True Negative (TN)**
    - The predicted value matches the actual value
    - The actual value was negative and the model predicted a negative value
- **False Positive (FP):** Type 1 error
    - The predicted value was falsely predicted
    - The actual value was negative but the model predicted a positive value
    - Also known as the Type 1 error
- **False Negative (FN):** Type 2 error
    - The predicted value was falsely predicted
    - The actual value was positive but the model predicted a negative value
    - Also known as the Type 2 error
- **Accuracy:** Overall, how often is the classifier correct?
    - (TP+TN)/total
- **Misclassification Rate:** Overall, how often is it wrong?
    - (FP+FN)/total
    - equivalent to 1 minus Accuracy
    - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
    - TP/actual yes
    - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
    - FP/actual no.
- **True Negative Rate:** When it's actually no, how often does it predict no?
    - TN/actual no.
    - equivalent to 1 minus False Positive Rate
    - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
    - TP/predicted yes
- **Prevalence:** How often does the yes condition actually occur in our sample?
    - actual yes/total

3. CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces. CNN have their "neurons" arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks.

A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.



3.1. CONVOLUTIONAL LAYERS USED

- **Keras Conv2D**
  It is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

- **BatchNormalization**
  Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

- **MaxPool2D**
  Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension. The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output_shape = math.floor((input_shape - pool_size) / strides) + 1 (when input_shape >= pool_size)

- **Flatten**
  Flatten is used to flatten the input. For example, if flatten is applied to a layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4).

- **Dropout**
  The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

- **Dense**
  In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks. The dense layer’s neuron in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix-vector multiplication.

4. MODULES

4.1. TensorFlow

TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations.

4.2. OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

4.3. Itertools

Itertools is a module in python, it is used to iterate over data structures that can be stepped over using a for-loop. Such data structures are also known as iterables. This module incorporates functions that utilize computational resources efficiently.

4.4. Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. Even for the delete operation, the Numpy array is faster. As the array size increase, Numpy gets around 30 times faster than Python List. Because the Numpy array is densely packed in memory due to its homogeneous type, it also frees the memory faster. NumPy is written in C, and executes very quickly as a result. By comparison, Python is a dynamic language that is interpreted by the CPython interpreter, converted to bytecode, and executed. While it's no slouch, compiled C code is always going to be faster.

4.5. OS

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os. path* modules include many functions to interact with the file system.

4.6. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats .
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

4.7. Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

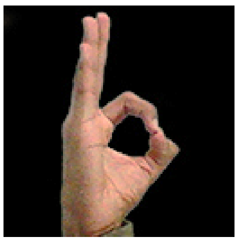
5. DATASET

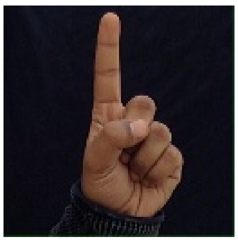
5.1. OVERVIEW

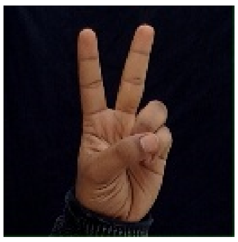
Data collection is the first and foremost step in any supervised machine learning — deep learning modelling. However, there is no public image dataset dedicated to the alphabet and number gestures in ISL. So, to overcome this problem, a self-generated ISL dataset composed of 36 classes, representing 0-9 number gestures and A-Z alphabet gestures was created by clicking pictures of them using a mobile camera from a Realme phone. Images for each gesture were taken by varying hand positions slightly procuring a total of 350 images of each class. Thus, a total of 12,600 images were captured for the dataset. Then we converted those images to a size of 128 px x 128 px.

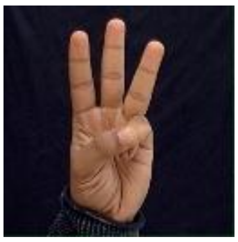
5.2. SAMPLES

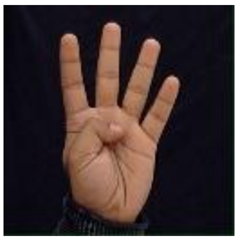
The dataset contains images of the following signs:

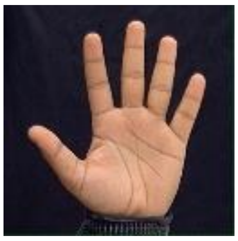

0

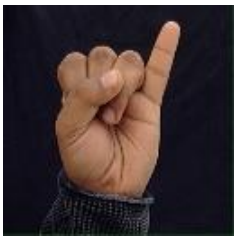

1

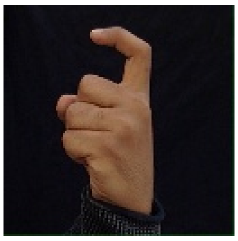

2

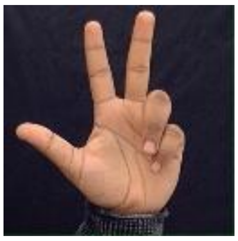

3

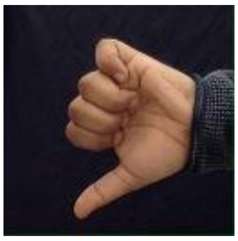

4

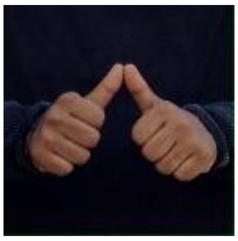

5

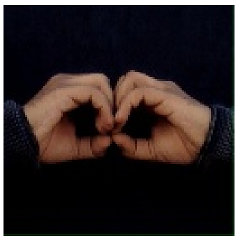

6

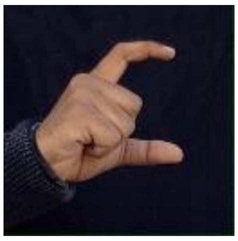

7

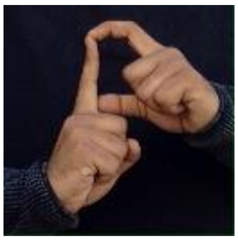

8

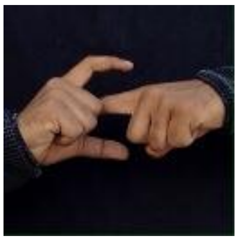

9

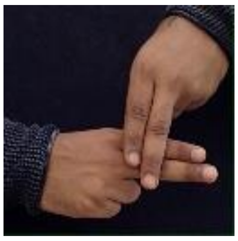

A

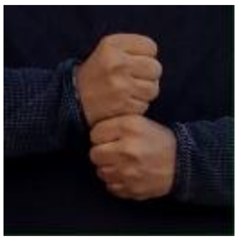

B

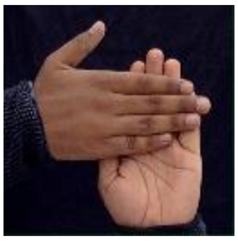

C

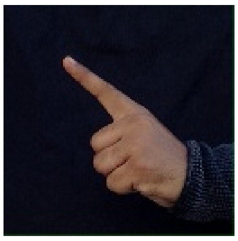

D

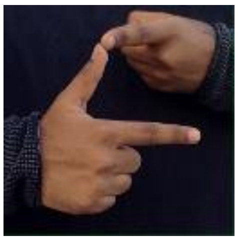

E

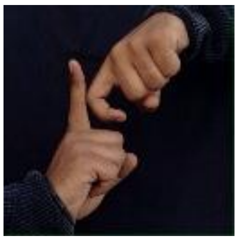

F

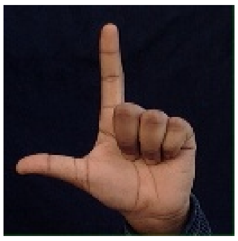

G

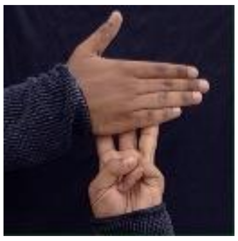

H

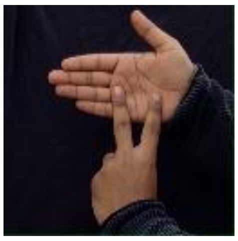

I

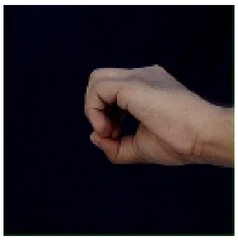

J

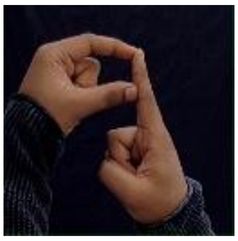

K

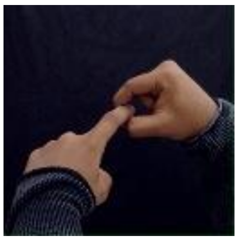

L

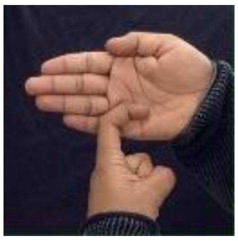

M

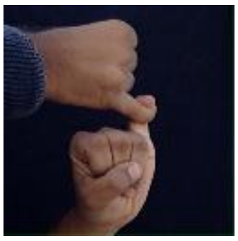

N

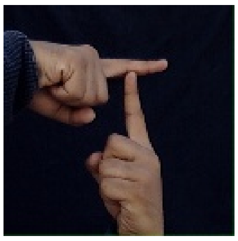

O

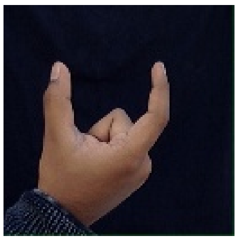

P

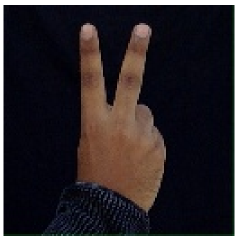

Q

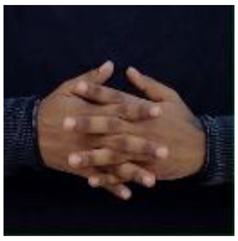

R

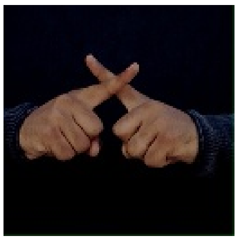

S

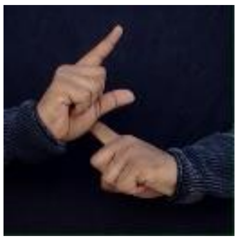

T

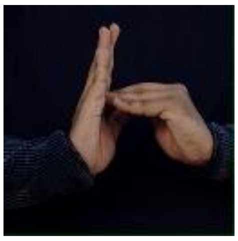

U


V


W


X


Y


Z

6. TRAINING

Before designing and training a Convolutional Neural Network (CNN), the dataset was loaded from the folder it was saved at, and the name of the folder for each gesture class was made class names for the images loaded. Before moving further, the images were resized to 128 x 128 pixels.

Then the data was split into 2 sets; the Training data which was 80% of the whole dataset and the Test data which was the remaining 20%. Once the dataset was pre-processed, it was ready to be fed into the CNN model along with their class labels. The model comprised the following layers:

• Convolutional Layers

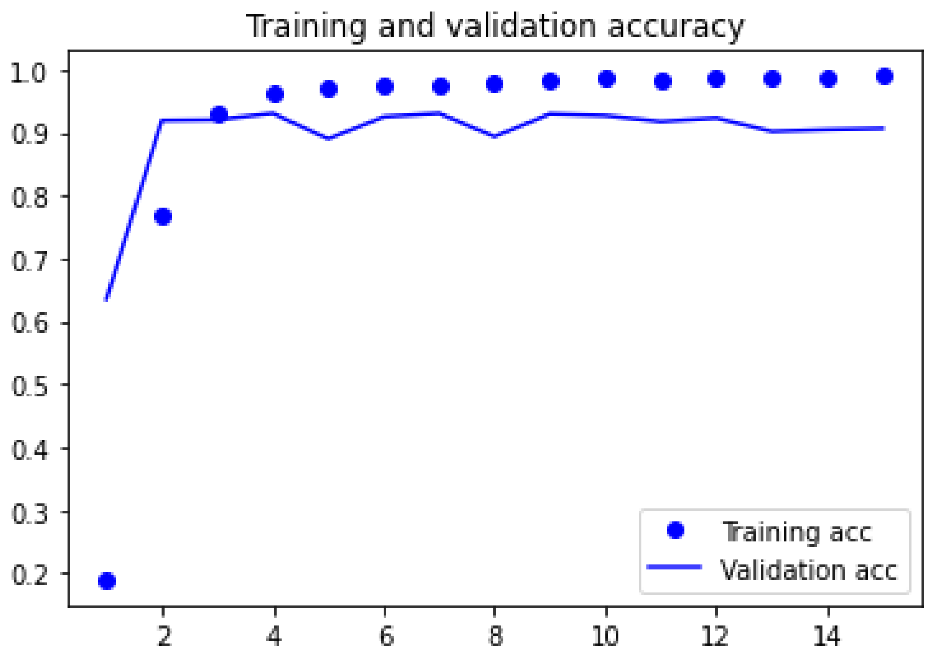
• Max Pooling Layers

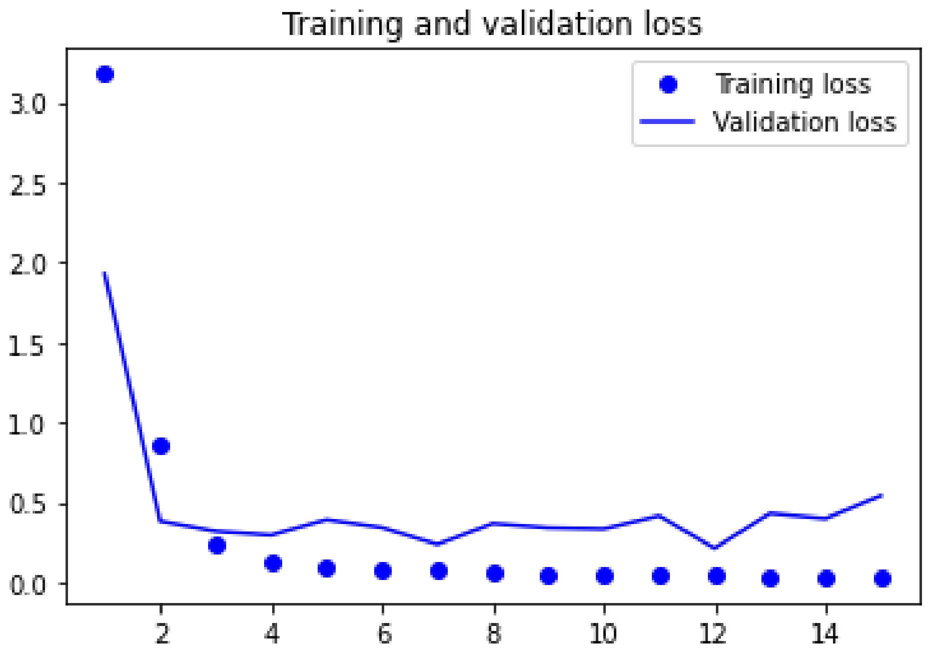
• Batch Normalization

• Dropout Layer

• Fully Connected Layers (Dense Layers)

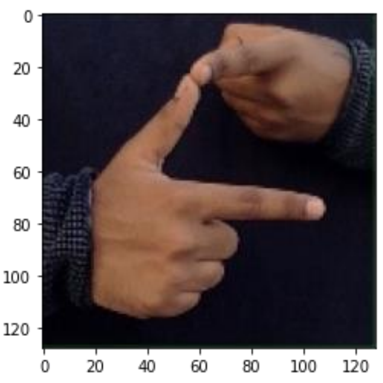
After the model was successfully compiled, we fit the model on the training dataset for 15 epochs using the two training callbacks discussed above and with a 10% validation data split. The model was saved after training was over.

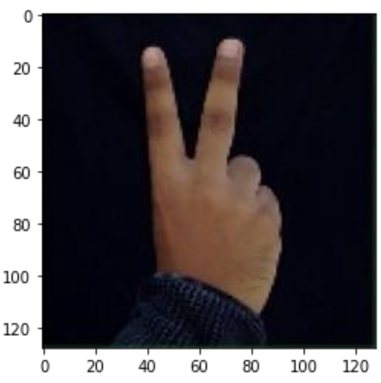


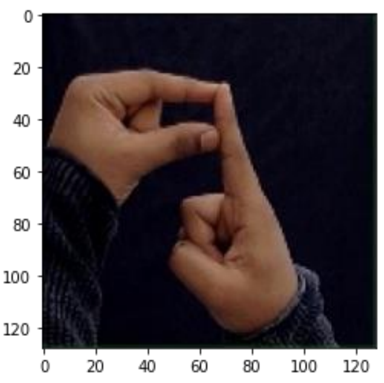

Training and validation loss
Training loss
Validation loss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
12
14

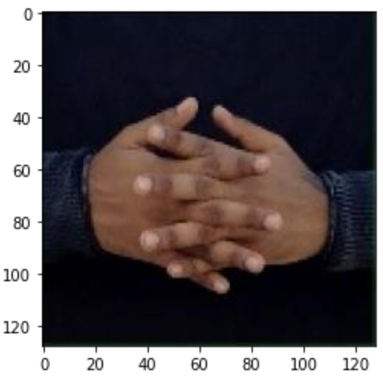
7. TESTING

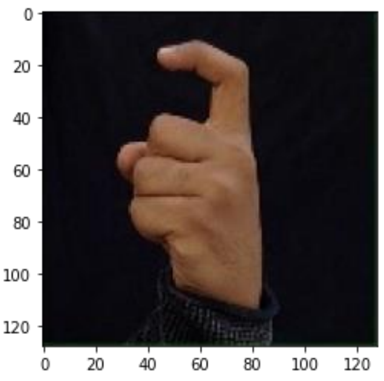
The model was loaded, and the testing images were fed to it, to test the proposed model with images it had not seen before. Some testing outputs are:

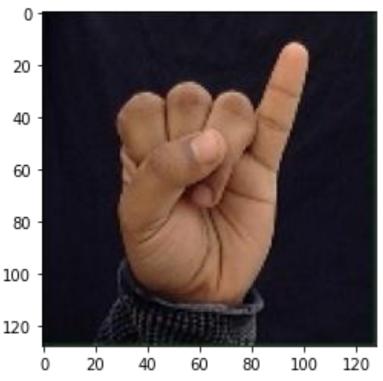

J

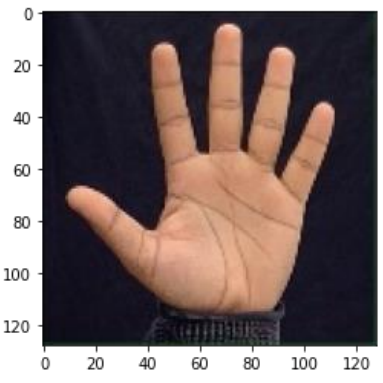

2

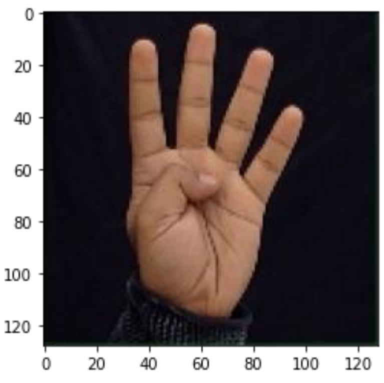

T

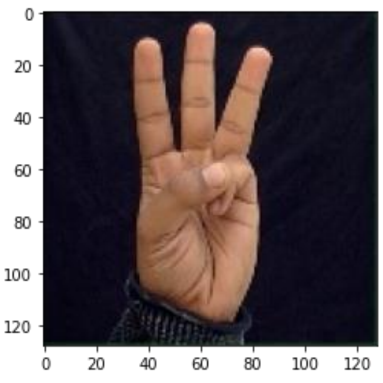

W


7


6

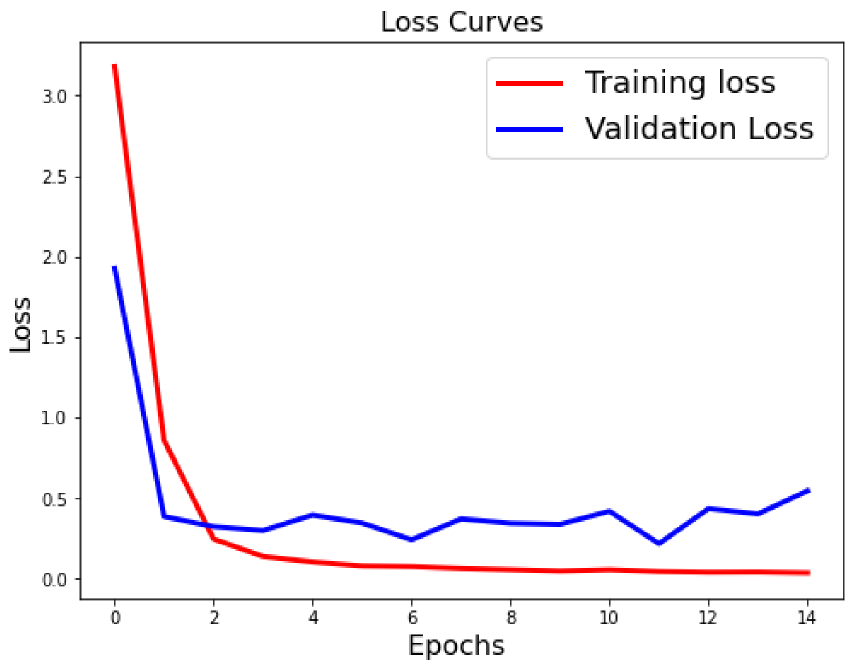

5

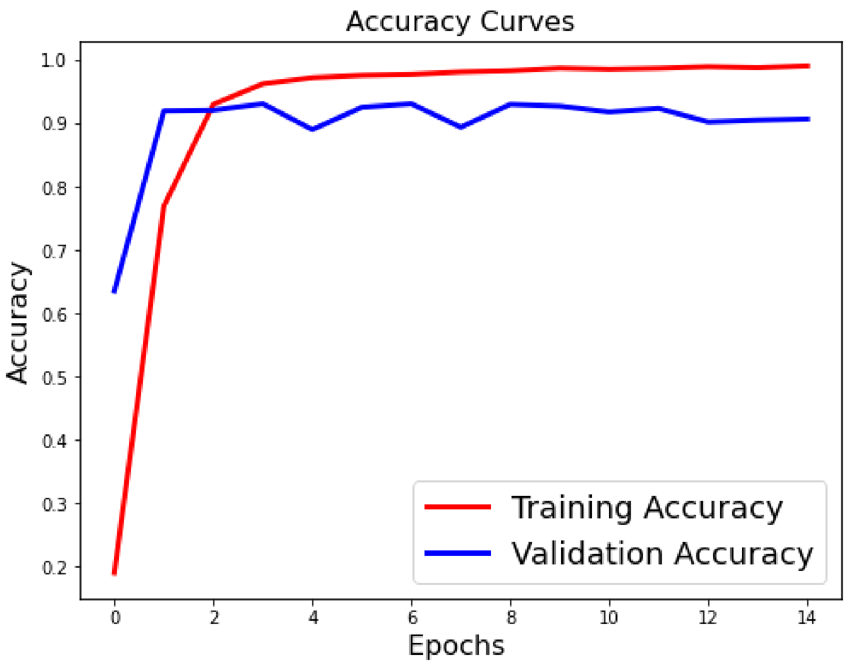

4


3

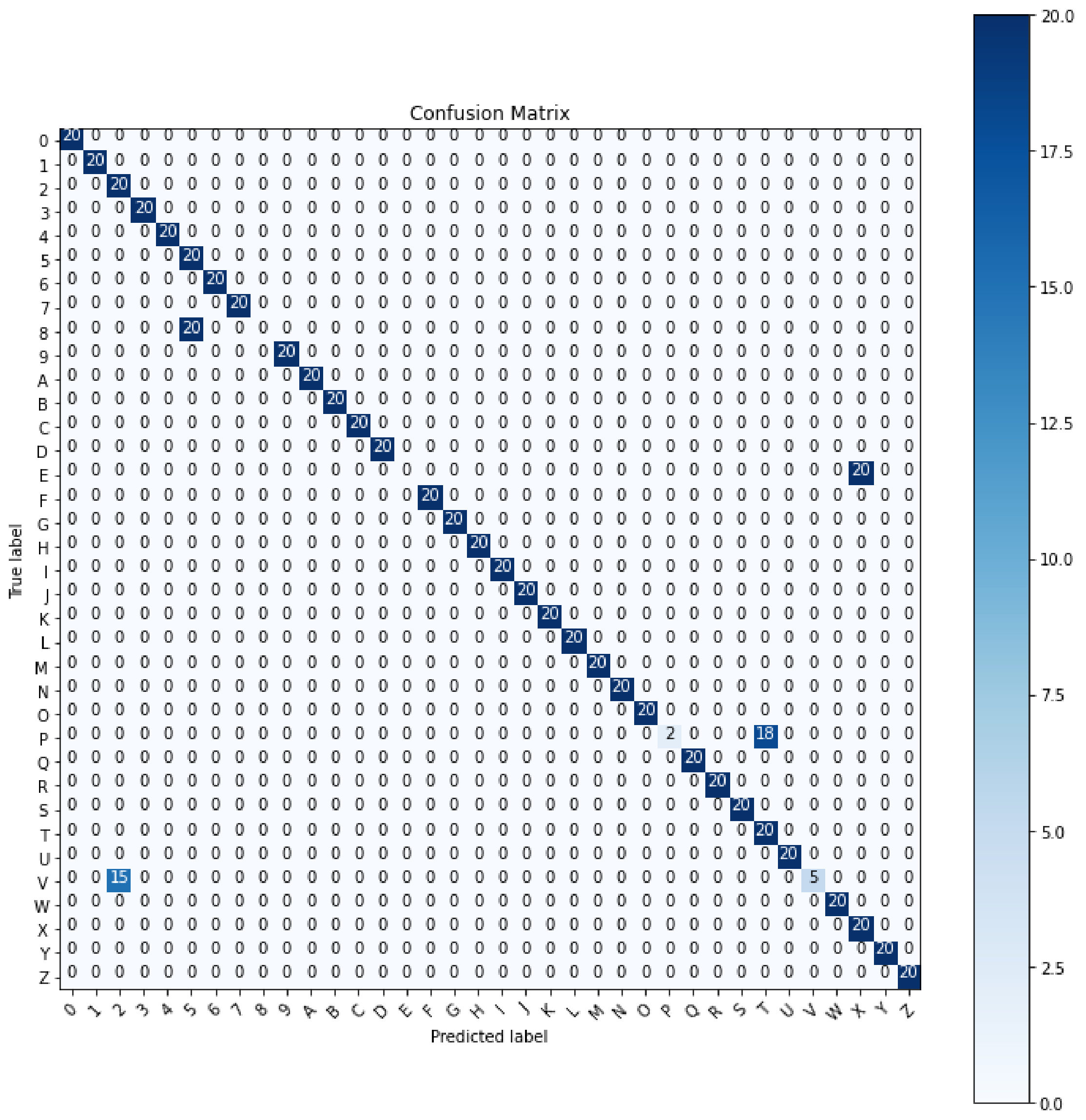
8. RESULTS

The model was trained and tested on 350 images of each ISL alphabet and numeral sign. A collection of 10800 randomly selected images out of the total 12600 images were fed into the model for training. The model achieved 90.65% validation accuracy. Post training, the model was tested on 1800 test images, and an accuracy of 99.03% and a loss value of 0.032 was obtained, with the model correctly classifying 1782 test images out of the total test dataset of 1800 images.





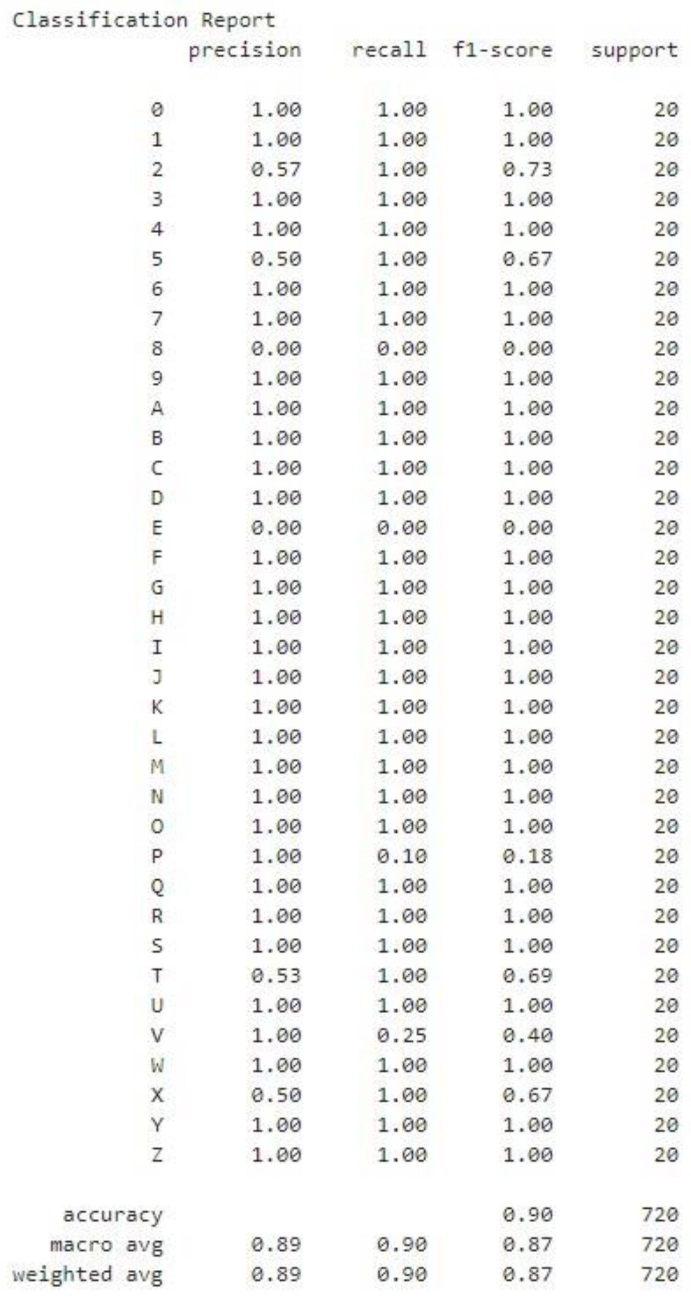
8.1. CONFUSION MATRIX



9. CONCLUSIONS

We successfully implemented a CNN based real-time Indian Sign Language (ISL) hand gesture recognition model using OpenCV and Keras. The model aimed to classify 36 ISL hand gestures representing A-Z alphabet and 0-9 numeral signs in real-time. The proposed CNN model was successfully implemented and achieved 90.65% validation accuracy on the training images. On testing the model on test data, an accuracy score of 99.03% was obtained.

Such a sign language recognition system can prove to be very helpful for the deaf and dumb population of the country in communicating with others.



```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        20
           2       0.57      1.00      0.73        20
           3       1.00      1.00      1.00        20
           4       1.00      1.00      1.00        20
           5       0.50      1.00      0.67        20
           6       1.00      1.00      1.00        20
           7       1.00      1.00      1.00        20
           8       0.00      0.00      0.00        20
           9       1.00      1.00      1.00        20
           A       1.00      1.00      1.00        20
           B       1.00      1.00      1.00        20
           C       1.00      1.00      1.00        20
           D       1.00      1.00      1.00        20
           E       0.00      0.00      0.00        20
           F       1.00      1.00      1.00        20
           G       1.00      1.00      1.00        20
           H       1.00      1.00      1.00        20
           I       1.00      1.00      1.00        20
           J       1.00      1.00      1.00        20
           K       1.00      1.00      1.00        20
           L       1.00      1.00      1.00        20
           M       1.00      1.00      1.00        20
           N       1.00      1.00      1.00        20
           O       1.00      1.00      1.00        20
           P       1.00      0.10      0.18        20
           Q       1.00      1.00      1.00        20
           R       1.00      1.00      1.00        20
           S       1.00      1.00      1.00        20
           T       0.53      1.00      0.69        20
           U       1.00      1.00      1.00        20
           V       1.00      0.25      0.40        20
           W       1.00      1.00      1.00        20
           X       0.50      1.00      0.67        20
           Y       1.00      1.00      1.00        20
           Z       1.00      1.00      1.00        20

    accuracy                           0.90       720
   macro avg       0.89      0.90      0.87       720
weighted avg       0.89      0.90      0.87       720
```

10. REFRENCES

- D. S. Breland, S. B. Skriubakken, A. Dayal, A. Jha, P. K. Yalavarthy and L. R. Cenkeramaddi, "Deep Learning-Based Sign Language Digits Recognition From Thermal Images With Edge Computing System," in IEEE Sensors Journal, vol. 21, no. 9, pp. 10445-10453, 1 May1, 2021, doi: 10.1109/JSEN.2021.3061608. (https://ieeexplore.ieee.org/document/9360817)

- OpenCV (https://opencv.org/)

- Keras Layers Api (https://keras.io/api/layers/)

- Data School (https://www.dataschool.io/)

- Towards Data Science (https://towardsdatascience.com/)

- Analytics Vidhya (https://www.analyticsvidhya.com)

- How Do Convolutional Layers Work in Deep Learning Neural Networks? (machinelearningmastery.com)