AN ENGINEERING PROJECT REPORT

On

RECOGNITION OF ALPHABETS IN NEPALI SIGN LANGUAGE

Submitted By

Atul Pokharel	(170308)
Nabin Raj Dhungel	(170322)
Nabin Joshi	(170323)
Rakshya Bhetwal	(170334)

Under the Supervision of

Dr. Aman Shakya

Asst. Prof. Ankit Bhattarai

Submitted To

The Department of Information and Communications Technology in partial fulfillment of requirement for the degree of Bachelor of Engineering in Computer

Cosmos College of Management & Technology

(Affiliated to Pokhara University)

Tutepani, Lalitpur, Nepal

September, 2022

Copyright

The author has agreed that the Library, University of Pokhara, Cosmos College of Management & Technology, May make this engineering project report freely available for inspection, Moreover, the author has agreed that permission for extensive copying of this report for scholarly purpose may be granted by the supervisor who supervised the work recorded here in or, in their absence, by the college authority in which the project work was done. Copying or any other use of this report for financial gain without approval of the college and author's permission is strictly prohibited.

Request for the permission to copy or to make and other use of the materials in this report in the whole or in part should be addressed to:

Cosmos College of Management & Technology
©Copyright 2022

Certificate

The undersigned certify that they have read & recommended to the Department of Electronics & Communication IT & Computer Civil, a final year project work entitled "The Title of the Project Work" submitted by (Names of the Students with Roll Numbers) in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

_____

Name & Post of Project Supervisor
(Project Supervisor)
Department of . . . . . .
Name of Institute

_____

Name External Examiner
(External Examiner)
Designation
Name of Institute

_____

Name
Head of the Department
Department of IT and Computer Engineering
Cosmos College of Management and Technology

Acknowledgment

The warm response received by this proposal, both from teachers and students, has encouraged the learning group to present our ideas and vision within time. Every topic of this proposal has been written in simple and lucid language so that any one going through it will be clearly able to understand the subject. Similarly, the context has been revised thoroughly to make it more comprehensive.

We would like to express our sincere gratitude to our supervisor Asst. Prof. Ankit Bhattarai for his guidance and support throughout the work. Our deepest appreciation to the entire team of Cosmos College of Management and Technology, for granting us this valuable opportunity and helping us reach our milestone by giving valuable feedbacks and suggestions. For the improvement of the subject matter of this proposed project, constructive criticism, suggestions and feedbacks for the improvement of this project would be highly appreciated.

- Project Team

Abstract

Speech impairement and hearing impairement are disabilities which affect the disability of individuals to communicate using speech and hearning. People affected with such disabilities use other media of communication such as sign language. There remains a challenge for non-sign language communicators to interact with sign language speakers. In the realm of multimodal communication, sign language is, and continues to be, one of the most understudies areas. In line with recent advances in the field of deep learning and machine vision, there are far reaching implications and applications for sign language interpretation. In this paper, we propose a method to create a vision based application which offers sign language translation to text thus aiding communication between signers and non-signers. The dataset we will be using is the Nepali Sign Language Dataset.

Keywords: *ASL, Adam, CNN, ConvNet, Deep Learning, FPN, Keras, Machine Vision, MobileNet, NSL, ReLu, RMSprop, R-CNN, Tensorflow, YOLO*

Contents

List of Figures

List of Tables

List of Abbreviations

ASL	American Sign Language
CNN	Convolutional Neural Network
DFD	Data flow Diagram
ML	Machine Learning
NSL	Nepali Sign Language
ReLu	Rectified linear Unit
RMSE	Root Mean Square Error
RMSprop	Root Mean Square prop
UML	Unified Modelling Language
YOLO	You Only Look Once

1. Introduction

1.1 Background

Sign language is a unique way of communication used by people with impaired hearing and speech. Non-verbal people use sign language gestures often to express their thoughts and emotions. However, this type of communication often goes understudied and also is difficult for non-signers to understand. Communication with signers usually requires Deaf Interpreters [1], who are not always available and are in high demand – the number of interpreters needed is expected to increase by 19% over 10 years. To facilitate communication across these communities and attempt to fulfill the need for this high demand, we aim to build a model that can translate Nepali Sign Language (NSL) to written roman form using computer vision techniques. Hand gestures are also beneficial in noisy settings when compared to speech instructions, in conditions where speech commands would be distressing, as well as for conveying spatial associations and quantitative information. These systems installed at public places like airports, banks, hospitals can ease the communication between the Deaf community and the hearing community.

As per the data of 2018, in Nepal [2], the prevalence of deafness is about 2.8% which accounts to about 7 lakhs population. About 10% of them were unaware of their problem because of mild impairment and 7% of them were suffering with disabling impairment. There are approximately around two hundred sign languages in use around the world today. And several efforts have been made to translate these languages into English language and other languages. There are many apps available for American Sign Language(ASL). However, there has not been significant progress in case of Nepali Sign Language(NSL). Nepali Sign language [3] is somewhat standardized language based informally on the variety of Kathmandu, with lesser input from varieties of Pokhara and elsewhere. It uses both static and dynamic fingerspelling (i.e. manual alphabet) and uses only single hand. Working with Nepali sign language for gesture recognition using digital image processing, static gesture was only considered at the initial phase. There are 36 consonant and 13 vowels in total 49 alphabet set or manual alphabet for Nepali language and the same set represented using gesture for Nepali sign language. Vowels are basically dynamic gesture and consonant are static gesture. Considering we plan to perform only the consonant set gesture recognition.

Figure 1.1: Consonant alphabet set of Nepali Sign Language [5]

1.2 Rationale

People with hearing and speech impairment while communicating and travelling face a lot of challenges. They cannot adapt to the surrounding environment quickly and respond to other normal people and expressing themselves is hard. It is also not possible to get an interpreter anywhere, any moment. At this age of technology, it is quintessential to make these people feel part of the society by helping them communicate smoothly. This requires an intelligent system to be developed and be taught without restricting the freedom of movement. Also there has been no notable research and progress in the field of Nepali Sign Language. Hence, there is need of a system that is able to recognize gestures corresponding to Nepali sign Language and translate it.

1.3 Statements of the Problems

Speech impaired and hearing impaired people use hand signs and gestures to communicate. Non-signers face difficulty in understanding their language. In context of Nepal, there has been no significant research on Nepali Sign Language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information.

Given a hand gesture, implementing such an application which detects predefined Nepali sign language (NSL) in a real time through hand gestures and providing facility for the user to be able to view the interpretation in a text form, also allowing them to communicate efficiently so that the problems faced by them can be accommodated with technological assistance and the barrier of expressing can be overshadowed [4].

1.4 Objectives

- To build a system that will be able to correctly identify alphabtes that corresponds to the hand gestures of Nepali Sign Language .
- To serve aspiring individuals to learn and talk in sign language.

2. Literature Review

Most of the recent works related to hand gesture interface technique has been categorized as: glove-based method and vision-based method. Glove-based method requires a user to wear a cumbersome device, and generally carry a load of cables that connect the device to a computer. Vision-based method consists of various techniques such as: Model-based and State-based.

Jhuma Sunuwar and Ratika Pradhan (2015) [5], proposed use of object descriptor representing boundary information of hand to make classification easy and simple by transforming the gesture to shape number. Further, chain code was used to represent the boundary which was sampled using grid lines to refine the chain code obtained for the contour. This approach proved to be simple approach for gesture recognition and was found satisfactory after processing some gestures of NSL.

Drish Mali, Rubash Mali, Sushila Sipai and Sanjeeb Prasad Pandey (2018) [2], proposed a system to translate NSL using CNN. The system proved to demonstrate considerable result for the recognition of signs of Nepali Sign Language. The accuracy of the model was high due use of red gloves for detecting hand of the signer. Even with the addition of two more gestures, it was observed that the accuracy of the model decreased by 0.83

There has been no consequential research and progress in NSL. However, several contribution has been made for American Sign Language (ASL). In Literature survey we have gone through some similar works that are implemented in the domain of sign language recognition.The summaries of each of the project works are mentioned below:

Mujahid, A., Awan, M. J., Yasin, A., Mohammed, M. A., Damaševičius, R., Maskeliūnas, R. and Abdulkareem, K. H. (2021) [6], proposed a lightweight model based on YOLO (You Only Look Once) v3 and DarkNet-53 CNN for gesture recognition without additional preprocessing, image filtering, and enhancement of images. The proposed model achieved high accuracy even in a complex environment, and it successfully detected gestures even in low-resolution picture mode. The proposed model was evaluated on a labeled dataset of hand gestures in both Pascal VOC and YOLO format. Further, they compared their model with Single Shot Detector (SSD) and Visual Geometry Group (VGG16), which achieved an accuracy between 82% and 85%.

Rasha Amer Kadhim and Muntadher Khamees (2020) [7], built a real-time ASL recognition system was built with a ConvNet algorithm using real colouring images from a PC camera. The model was the first ASL recognition model to categorize a total of 26 letters, including (J & Z), with two new classes for space and delete, which was explored with new datasets. It was built to contain a wide diversity of attributes like different lightings, skin tones, backgrounds, and a wide variety of situations. The experimental results achieved a high accuracy of about 98.53% for the training and 98.84% for the validation.

Abdulwahab A. Abdulhussein and Firas A. Raheem (2020) [8], put forward a model based on Gesture recognition of static ASL using Deep Learning.The contribution consisted of two solutions to the problem. The first one was resized with Bicubic static ASL binary images. Besides that, Robert edge detection method was used for good recognition of hand boundary. The second solution was to classify the 24 alphabets static characters of ASL using Convolution Neural Network (CNN) and Deep Learning.Their work resulted in classification accuracy equaling to 99.3% and the error of loss function was 0.0002.

Wen, Feng and Zhang, Zixuan and He, Tianyiyi and Lee, Chengkuo (2021) [9], proposed an artificial intelligence enabled sign language recognition and communication system comprising sensing gloves, deep learning block, and virtual reality interface. Non-segmentation and segmentation assisted deep learning model were used to achieve the recognition of 50 words and 20 sentences. Significantly, the segmentation approach splited entire sentence signals into word units. Then the deep learning model recognized all word elements and reversely reconstructed and recognizeed sentences. Furthermore, new/never-seen sentences created by new-order word elements recombination could also be recognized with an average correct rate of 86.67%. Then, the sign language recognition results were projected into virtual space and translated into text and audio.

3. Requirement Analysis

3.1 Feasibility Study

3.1.1 Technical Feasibilty

End-to-end open-source deep learning framework Tensorflow has been the backbone of this project. Various libraries of Keras, Tensorflow has been used in the development of this project.

3.1.2 Financial Feasibilty

The costs to develop this project has not exceeded more than the resource cost.

- **Hardware Cost**

Requirements	Details	Cost (Rs.)
Processor	AMD Ryzen 7 4500 or equivalent Intel processor	45,000
RAM	4 GB or higher	4,200
HDD/SSD	500 GB (SSD)	9,500
GPU	NVIDIA GTX 1660 ti	49,500
Webcam	Inbuilt	-

Table 3.1: Hardware specific cost

- **Software Cost**

Requirements	Version	Cost (Rs.)
Python	3.6	Open Source
OpenCV	2.4 or higher	Open Source
Tensorflow	2.0	Open Source
Keras	2.3.0 or higher	Open Source

Table 3.2: Software specific cost

3.1.3 Legal Feasibilty

While developing this project we have addressed all the legal requirements. All the data required for this project has been created by our own project team. We have considered protecting our Intellectual Property (IP) rights and minimizing the risk of infringing the IP rights of others. This project has been developed without breaching any law in the country.

3.2 Functional and Non-Functional Requirements

3.2.1 Functional Requirements

- **Gesture Recognition:**
  Software should automatically recognize the gesture through the video input.
- **Authentic Representation:**
  Software should give out the correct meaning of the gesture.
- **Cross platform support:**
  Software should run on as many platforms as possible.

3.2.2 Non-Functional Requirements

- **Availability:**
  The software is being developed such that it will be available at all times.
- **Reliability:**
  The software is being developed such that it will provide accurate meaning of gestures.
- **Scalability:**
  The software is being developed such that it will be able to handle all the basic gestures.
- **Maintainability:**
  The software is being developed in such a way that it will be easily readable and maintainable.

3.3 Hardware and Software Requirements

3.3.1 Hardware Requirements

- Processor: AMD Ryzen 7 4500 or equivalent Intel processor
- RAM: 4 GB or higher
- HDD: 32 GB available disk space
- GPU: NVIDIA GTX 1660 ti or higher or any AMD equivalent card
- Camera: Webcam

3.3.2 Software Requirements

- Programming: Python, OpenCV, TensorFlow, Keras
- Operating System: Windows 11

3.4 Tools and Environments

3.4.1 Software Tool

- **The Jupyter Notebook**

  The Jupyter Notebook is an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and narrative text. It provides us with an easy-to-use, interactive data science environment that doesn't only work as an integrated development environment (IDE), but also as a presentation or educational tool. We can Jupyter Notebooks for all sorts of data science tasks including data cleaning and transformation, numerical simulation, exploratory data analysis, data visualization, statistical modeling, machine learning, deep learning, and much more.

3.4.2 Libraries

The Libraries are listed below:

- **TensorFlow**

  TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning and developing neural networks faster and easier. TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

- **Keras**

  Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

- **OpenCV**

  OpenCV(Open Source Computer Vision) is an open source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAV.

4. Methodology

4.1 Software Process Model

As the development process includes constant improvement with the code as well as dataset, we have been using incremental model for software development.
In this model, we have been designing and testing the system in different phases incrementally to built the desired model.

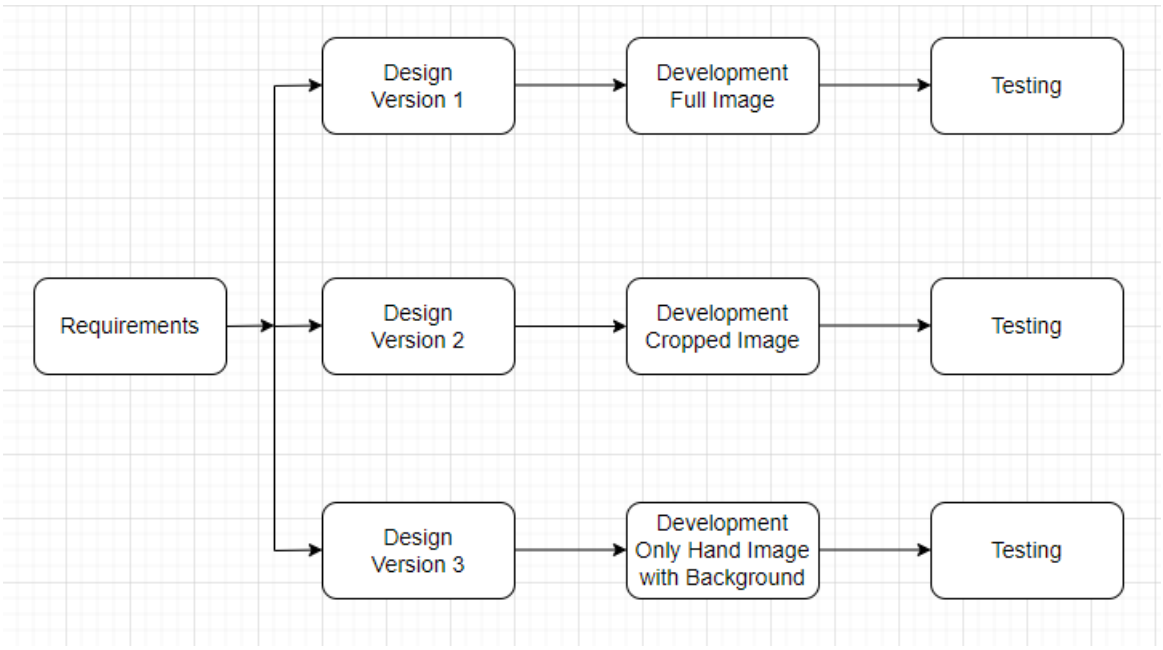


Figure 4.1: Incremental Model

In the first increment, we created the dataset and use CNN model and generate the output but the accuracy seems to be very low, and the prediction was not accurate. then we realize this situation may occur due to the low dataset size, Hence in the next increment we increase the dataset size and change the optimizer and activation of the CNN model, in this increment there is the issue with the testing and evaluation and in real-time not many alphabets were predicted correctly, then in the third increment we use the plain background for the dataset and still increase the dataset size and change the resolution of the image that leads to some satisfying result.

4.2 System Workflow

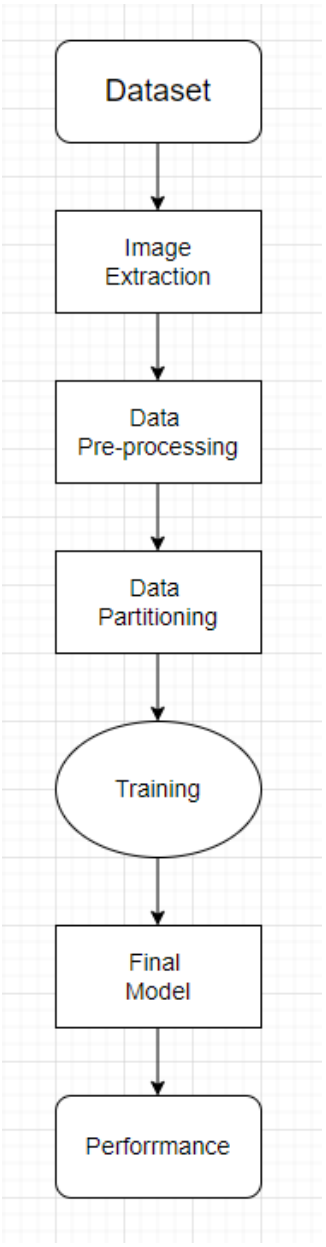


Figure 4.2: System Workflow

4.2.1 Dataset Creation

We used OpenCV library to capture images through web-cam. To collect the images for our dataset, we first defined the labels for each of the alphabets and number of images that we were going to collect for each of the classes. Then we set up the path for each of the class folders in which the respective images were going to be stored. The image path was set inside a folder named 'collectedimages' in our local computer. The input image was of 640*480 dimension. The file size was 27.4 KB of each image with it's horizontal and vertical resolution both being 120 dpi. The bit depth of the original image was 124.

Similarly, Opencv library also provides various filters that we applied on our model. Filters like grayscale conversion were applied on the target images. Grayscale conversion simply converts the image into black and white images.The input images we collected were at first normalized and later converted into a grayscale image. The above functions are discussed in detail in further subsections.

Few of the collected images are shown below:

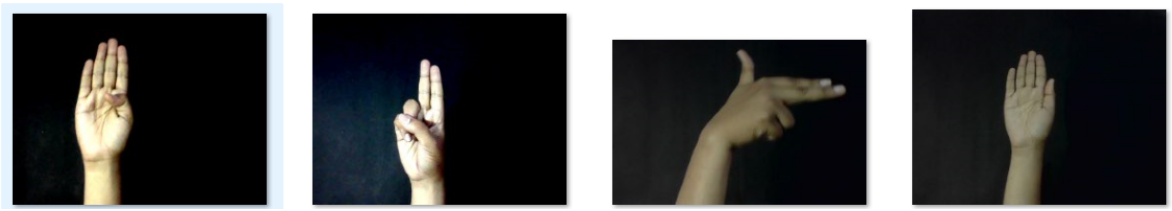


Figure 4.3: Collected images

4.2.2 Image pre-processing

Step 1 Reading Image

In this step, we simply have stored the path to our image dataset into a variable and then we create a function to load folders containing images into arrays so that computer can deal with it.

Step 2 Resize Image

Some images captured by webcam which is used to feed our algorithm may vary in size, therefore, we established a base size for all images fed into our algorithms by resizing them once into 64*64 dimension.

Step 3 RGB normalization

When we have unidirectional source of light then it creates shadow and different shades of colors. For eg: Intense light from right side of body whitens the right side of body and detecting skin or clothes from such image is not possible as different shades of colors are there. Some part is in highlighted and some in shadowed region. Hence, to reduce such effects of light, Normalization of color space is helpful. We use RGB Normalization so as to remove such highlighted regions, shadows which helped to make the object easier to detect.

4.2.3 Test-Train Split

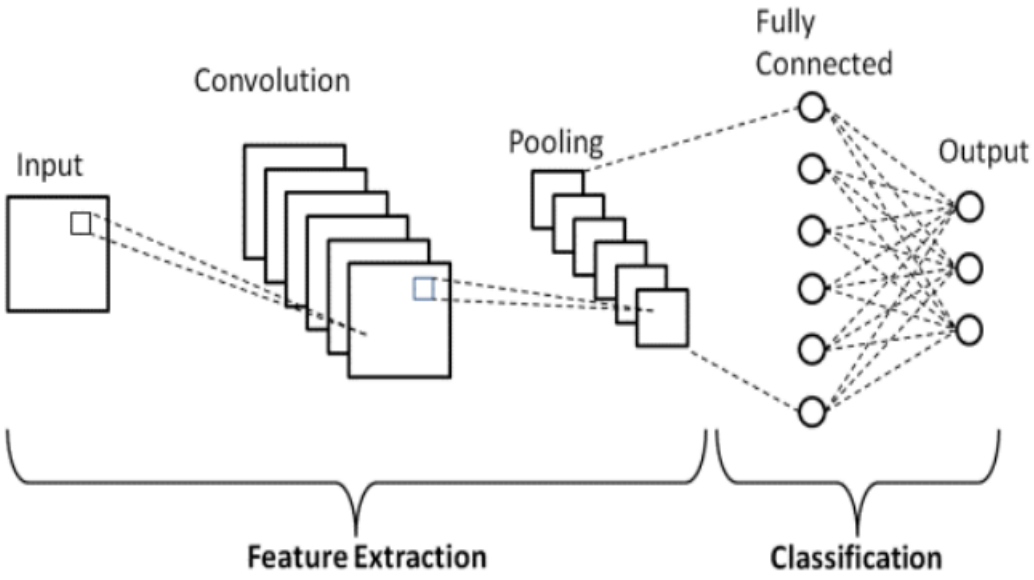
The whole data set collected was then split into training (70%) and testing (30%) subsets. The splitting was done by picking at random which results in a balance between the training data and testing data amongst the whole dataset. After the dataset was split CNN algorithm was implemented. The test and train dataset are loaded to the algorithm and the accuracy and root mean square error (RMSE) were determined from the output it generated. In total 37 symbols were taken as classes which was splitted as shown in Figure 4.4.

```
Total number of symbols:  37
Number of training images:  54390
Number of testing images:  23310
Number of evaluation images:  744
```

Figure 4.4: Test-Train split

4.2.4 Convolutional Neural Network

We are using Convolutional Neural Network(CNN) to train the model. CNN is a deep learning algorithm which takes input as images, assigns weights or biases to various aspects of image which helps in distinguishing the images from one another.



Source: https://www.upgrad.com/blog/wp-content/uploads/2020/12/1-4.png

Figure 4.5: Architecture of CNN Model

4.2.4.1 Why CNN?

- CNNs are automatic feature extractors.
- We can use max pooling to reduce size of image without compromising with image quality.
- CNN provides usage of convolutions, which act as feature emphasizer.

Furthermore, The OpenGenus IQ team conducted a performance study of various image classifiers in order to determine which was the best classifier for image classification. The findings of an image classification task used to distinguish lymphoblastic leukaemia cells from non-lymphoblastic ones have been presented to support their performance review. The features were extracted using a convolutional neural network and they were fed to different classifiers. Performance analysis of each of the classification algorithm is shown below:

CLASSIFIER	ACCURACY	PRECISION	RECALL	ROC
SVM	85.68%	0.86	0.87	0.86
Decision Trees	84.61%	0.85	0.84	0.82
KNN	86.32%	0.86	0.86	0.88
ANN(for 100 epochs)	83.10%	0.88	0.87	0.88
CNN(for 300 epochs)	91.11%	0.93	0.89	0.97

Figure 4.6: Performance analysis for binary image classification by OpenGenus IQ

4.2.4.2 Classification using CNN

Image classification is the process of taking an input(like a picture) and outputting its class or probability that the input is a particular class. Neural networks are applied in the following steps:

i. One hot encode the data:
A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

ii. Define the model:
A model said in a very simplified form is nothing but a function that is used to take in certain input, perform certain operation to its beston the given input (learning and then predicting/classifying) and produce the suitable output.

iii. Compile the model:
The optimizer controls the learning rate. We have used 'rmsprop' as our optmizer. Rmsprop is generally a good optimizer to use for many cases. The rmsprop optimizer adjusts the learning rate throughout training as per the parameters. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

iv. Train the model:
Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

v. Test the model:
A convolutional neural network convolves learned featured with input data and uses 2D convolution layers.

4.2.4.3 Convolution Operation

In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other.

Convolution formula:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Here are the three elements that enter into the convolution operation:

- Input image.
- Feature detector.
- Feature map.

Steps to apply convolution layer:

- You place it over the input image beginning from the top-left corner within the borders you see demarcated above, and then you count the number of cells in which the feature detector matches the input image.

- The number of matching cells is then inserted in the top-left cell of the feature map
- You then move the feature detector one cell to the right and do the same thing. This movement is called a and since we are moving the feature detector one cell at time, that would be called a stride of one pixel.
- What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so you write "1" in the next cell in the feature map, and so on and so forth.
- After you have gone through the whole first row, you can then move it over to the next row and go through the same process. There are several uses that we gain from deriving a feature map.

  These are the most important of them: Reducing the size of the input image, and you should know that the larger your strides (the movements across pixels), the smaller your feature map.

4.2.4.4 Activation Function

Rectified Linear Unit (ReLU), unlike other activation functions, does not activate all neurons at the same time. Rectified linear unit is used to scale the parameters to non negative values.We get pixel values as negative values too . In this layer we make them as 0's. The purpose of applying the rectifier function is to increase the non-linearity in our images. The reason we want to do that is that images are naturally non-linear. The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation. What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).The essential difference between the non-rectified version of the image and the rectified one is the progression of colors. After we rectify the image, we fond the colors changing more abruptly. The gradual change is no longer there. That indicates that the linearity has been disposed of.

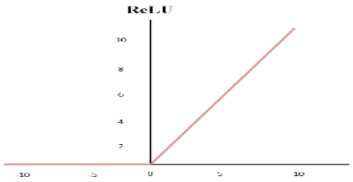


Figure 4.7: ReLU

4.2.4.5 Pooling

We have used Max pooling in our model. Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map. The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation.

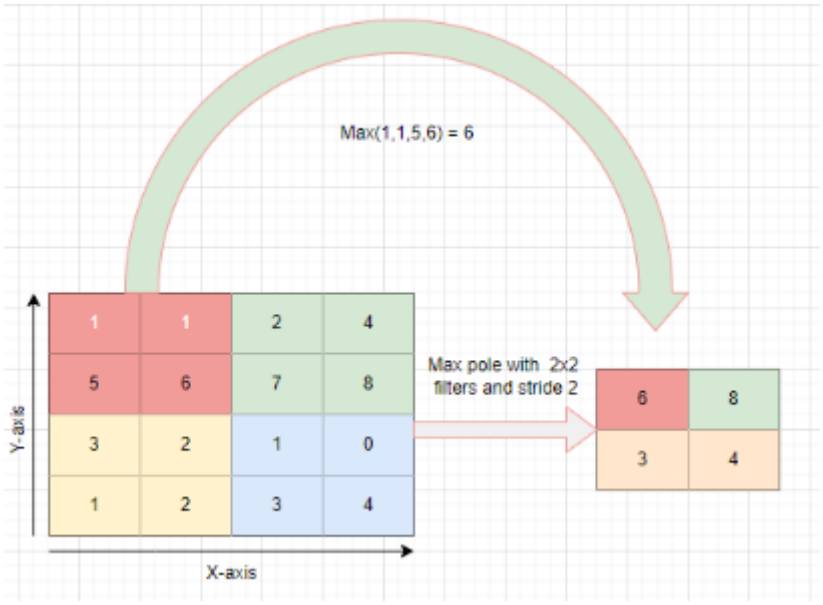


Figure 4.8: Max pooling

4.2.4.6 Optimizer

We have used Root Mean Square Prop (RMSprop) Optimizer in our classification model. This optimizer aims to reduce the aggressiveness of the learning rate by taking an exponential average of the gradients instead of the cumulative sum of squared gradients. RMS prop also takes away the need to adjust learning rate, and does it automatically. More so, RMSProp choses a different learning rate for each parameter.

4.2.4.7 Loss Function

Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only. Similarly, the block before the Target block must use the activation function Softmax.

4.2.4.8 Learning Rate

Learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a sub-optimal solution, whereas a learning rate that is too small can cause the process to get stuck.

```
model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
callbacks = [ ReduceLROnPlateau(monitor='loss', patience=2, cooldown=1),
              EarlyStopping(monitor='accuracy', min_delta=1e-4, patience=2)]
```

Figure 4.9: Parameters of CNN

4.2.4.9 Fully Connected Layer

The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network. It has neurons linked to each other ,and activates if it identifies patterns and sends signals to output layer .the output layer gives output class based on weight values, For now, all you need to know is that the loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness. That requires certain things to be altered in our network. These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features

and has to be reviewed multiple times for the sake of optimization.This full connection process practically works as follows:

- The neuron in the fully-connected layer detects a certain feature; say, a nose.
- It preserves its value.
- It communicates this value to the classes trained images.

4.2.4.10 Training CNN Model

In our model, we have added five convolution layers where two convolution layers have 64 filters, other two layers have 128 filters and the fifth layer has 256 filters. The convolution layers applies the rectified linear unit activation (ReLu) function. We also have dropout layers which acts as a mask to nullify the contributions of neurons towards the next layer. We have a dense layer with 37 units that applies softmax activation function. The output of convolution layer is a feature map which highlights the features of gesture. We have added two max pool layers of pool size (4, 4), which helps in reducing the size of input data to learning layers of model while ensuring that the details in image are not lost. This is done in order to reduce the computing time.

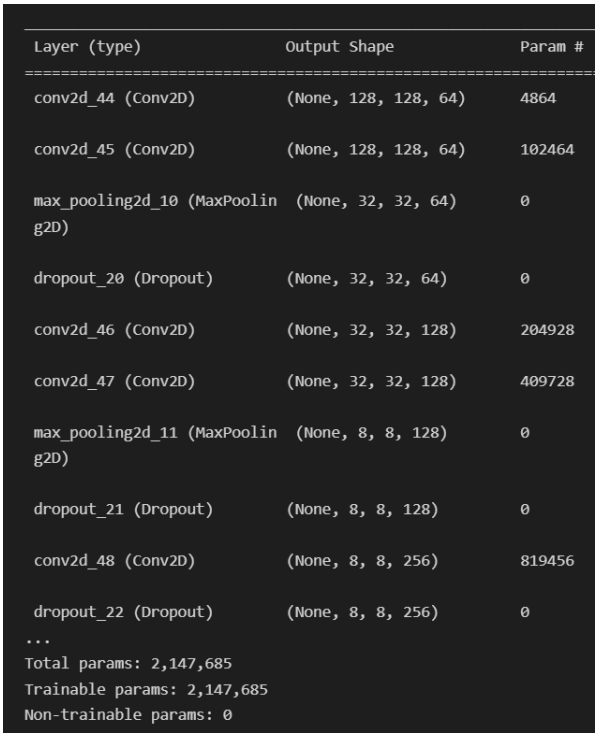
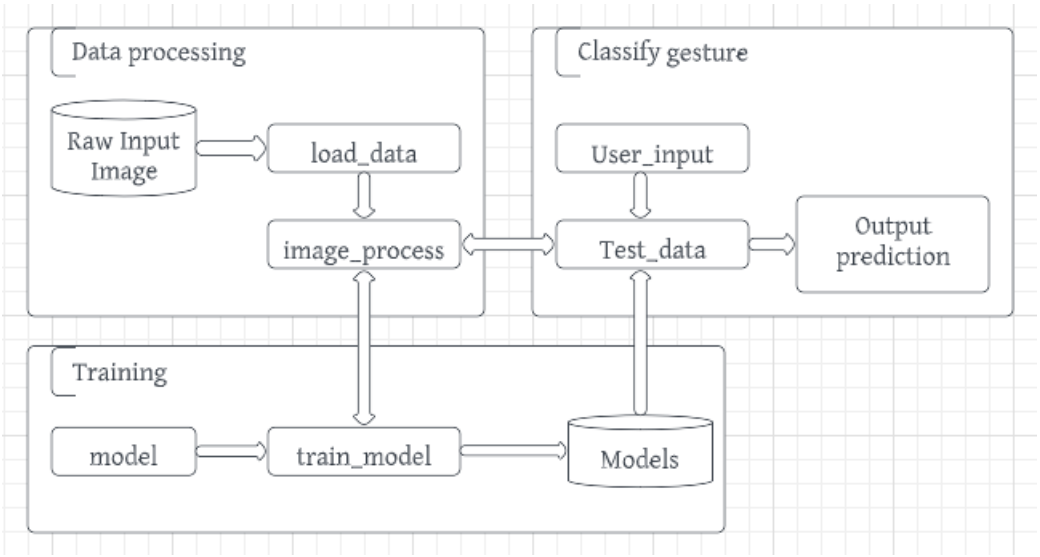


Figure 4.10: Model Summary

4.3 Overall System Structure



Source: https://ecasp.ece.iit.edu/publications

Figure 4.11: System Structure of Model

The overall process can be generalized as:

- Data Processing
  At first we collect the dataset in the form of Raw Image and save them in the file storage. Then we preprocess the images by resizing/rescaling, normalizing the color and grayscaling to enhance features. During training the processed image data is split into training, evaluation, and testing data and written to storage.

- Training
  We then train the model with hyperparameters that lists the learning rate, batch size, number of epochs, etc.. The model is then trained using RMSprop optimizer with Cross Entropy Loss. Further, the model is evaluated every epoch on the validation set and the model with best validation accuracy is saved to storage for further evaluation and use. Upon finishing training, the training and validation error and loss is saved to the disk, along with a plot of error and loss over training.

- Classify gesture
  After model is trained, it can be used to classify NSL gesture that is available as a file on the filesystem.

4.4 UML Diagrams

UML stands for Unified Modeling Language. Taking SRS document of analysis as input to the design phase drawn UML diagrams. The UML is only language so is just one part of the software development method. The UML is process independent, although optimally it should be used in a process that should be driven, architecture-centric, iterative, and incremental. The UML is language for visualizing, specifying, constructing, documenting the articles in a software- intensive system. It is based on diagrammatic representations of software components. A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of the system. A modeling language such as the UML is thus a standard language for software blueprints. The UML is a graphical language, which consists of all interesting systems. There are also different structures that can transcend what can be represented in a programming language.

There are various UML Diagrams which are:

4.4.1 Use Case Diagram

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the function provided by the system as a set of events that yield a visible result for the actor.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.

- **Flow of Events in Use Case Diagram**

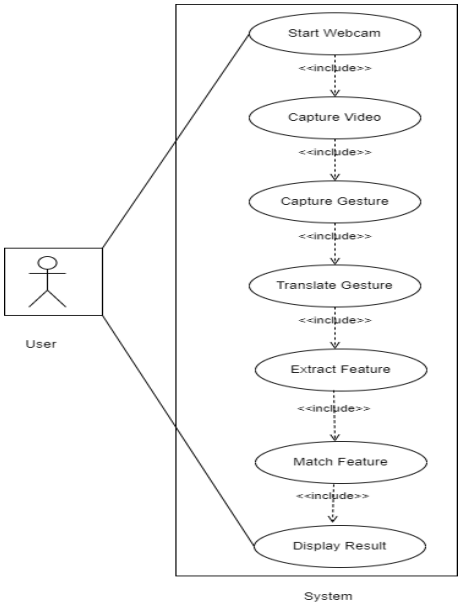


Figure 4.12: Flow of Events in Use Case Diagram

- **Use Case Scenario**

Use Case Name	Sign Language Recognition
Participating Actors	User, System
Flow of Events	Start the system(u) Capturing video(s) Capture gesture(s) Translate gesture(s) Extract feature(s) Match feature(s) Recognize gesture(s) Display result
Entry Condition	Run the code
Exit Condition	Displaying the label

Table 4.1: Use Case Scenario for NSL Recognition System

4.4.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. They're also called event diagrams. It depicts the objects involved in the scenario which in our system are user, webcam, system and dataset along with the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. A sequence diagram is a good way to visualize and validate various runtime scenarios.

Sequence diagram for our system is shown below:

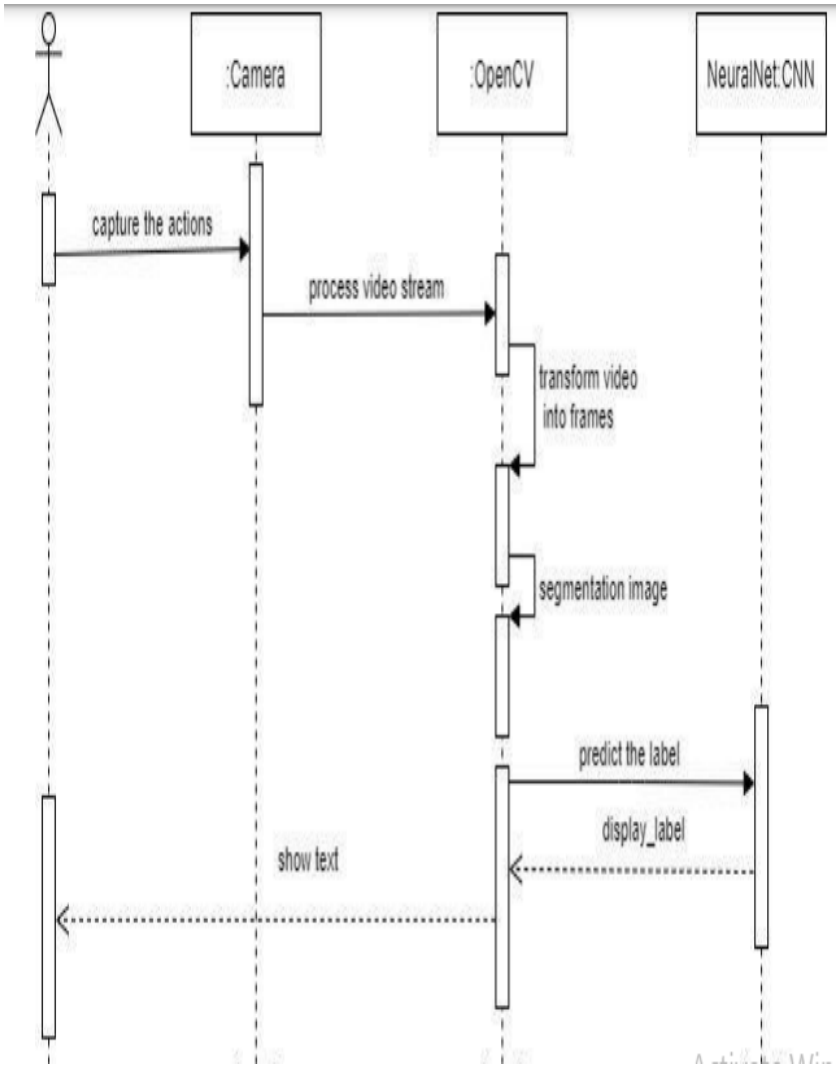


Figure 4.13: Sequence Diagram

4.5 Data Flow Diagram(DFD)

- **Context Level DFD / Level 0 DFD**

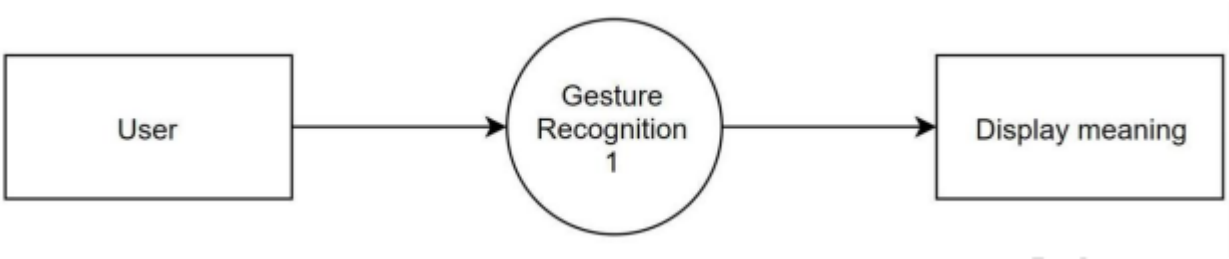


Figure 4.14: Level 0 DFD

In this level 0 data flow diagram, the whole system is represented with the help of input, processing and output. The input to the gesture recognition system will be the live feed from the camera which will contain the gestures performed by the user. The camera will provide the frames which can be mapped to their corresponding meanings.

- **Level 1 DFD**

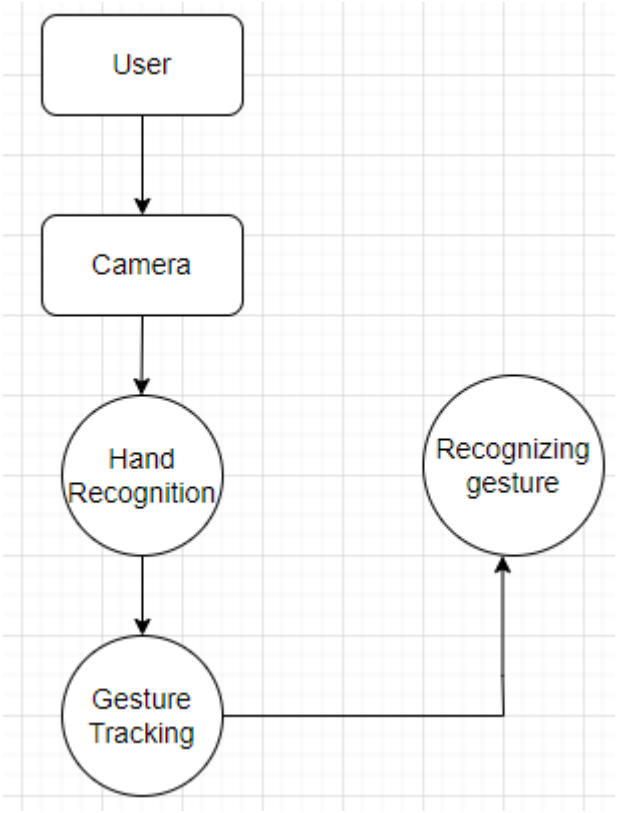


Figure 4.15: Level 1 DFD

In level 1 data flow diagram, the gesture recognition module is explained in further detail. The camera will provide live feed of the user actions. Operations will be performed to enhance the hand movements. The hand movement will be recognized and sent to the gesture tracking module. The gesture tracking module will check for the gesture in the pre-trained network of gestures and their respective meanings. The gesture control module will map the gesture to its meaning. Thus in these processes the gesture will be recognized and meaning will be displayed.

4.5.1 Implementation of Model

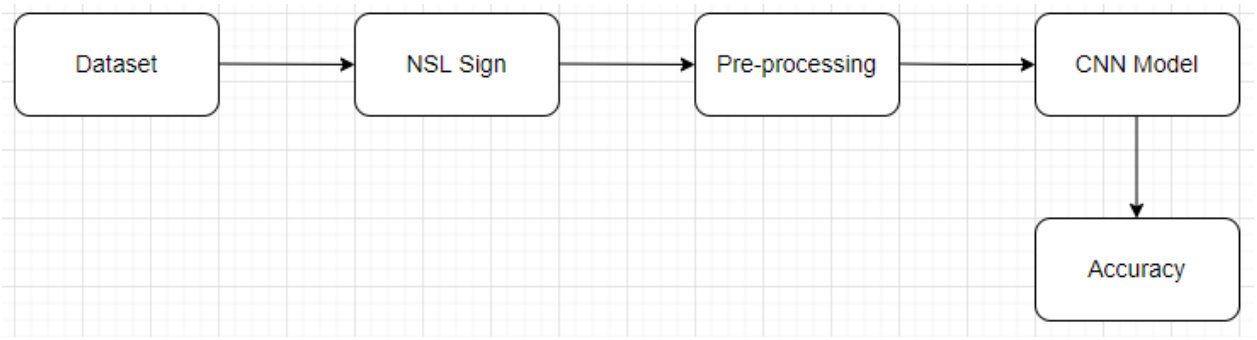


Figure 4.16: System architecture

Generalization of steps taken:

- At first, we dataset is created by capturing sample amount of images through a webcam. When the system starts up, it begins collecting gesture images from the previously generated dataset.
- Then the pre-processed of images by rescaling the images, normalizing them and converting them to gray-scale occurs.
- These images are then divided into three categories: Training, Validation, and Testing.
- After this, the model starts building by adding different layers to the image for the accuracy purpose of all the input images.
- Then, different layers are added to the Model. After the addition of the layers, the training of the CNN model begins.
- With the help of CNN, the model builds a network and creates a prediction model which will be used in the prediction of the result.
- After this, 'validation' and 'testing' of the model is performed to check whether the model is over-fitting or under-fitting.

4.5.2 Validation Criteria

Confusion Matrix

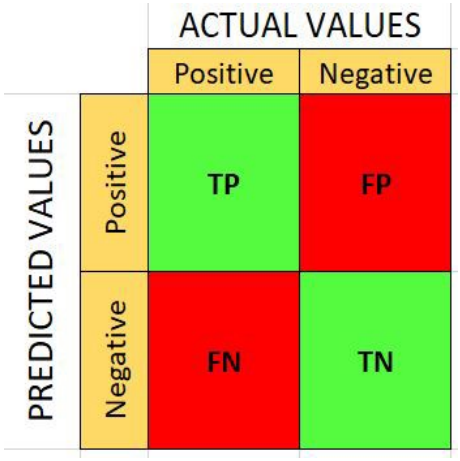


Figure 4.17: Confusion Matrix for Classification

Confusion Matrix is the visual representation of actual vs predicted value and it help to measure the performance of our Machine Learning model. There are 4 basic Terminology in confusion matrix which represent the different combinations of Actual and Predicted values for determining the metrics.

- True Positives (TP): when the actual value is Positive and predicted is also Positive.
- True negatives (TN): when the actual value is Negative and prediction is also Negative.
- False positives (FP): When the actual is negative but prediction is Positive. Also known as the Type 1 error.
- False negatives (FN): When the actual is Positive but the prediction is Negative. Also known as the Type 2 error.

other metrics associated with confusion matrix that help to understand the performance and analysis are

- Accuracy; It is the measure of how often the classifier makes the correct prediction. It is the ratio between the number of correct predictions and the total number of predictions. given by

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- Precision: Precision checks how many outcomes are actually positive outcomes out of the total positively predicted outcomes. It is a measure of correctness that is achieved in true prediction. Precision is defined as the ratio of the total number of correctly classified positive classes divided by the total number of predicted positive classes, which is given by

$$Precision = \frac{TP}{TP + FP}$$

- Recall: It is a measure of actual observations which are predicted correctly. It simply evaluates how many positive classes are actually predicted positively. It is the ratio of the total number of correctly classified positive classes divided by the total number of positive classes. which is given by

$$Recall = \frac{TP}{TP + FN}$$

- F1 score: F1 score is a number between 0 and 1 and is the harmonic mean of precision and recall.F1 score maintains the balance between Precision and Recall. that is if precision is low F1 score is low and again if the Recall is low F1 score is low. F1 score is simply calculated as

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

5. Result and Discussion

5.1 Testing Accuracy

Test Accuracy	Evaluation Accuracy
99.56 %	75.297%

Table 5.1: Testing and Evaluation Accuracy

In the above Table 5.1 it shows the fractions of predition our model got right against the test image and the evaluation image with the model.

5.2 Confusion Matrix

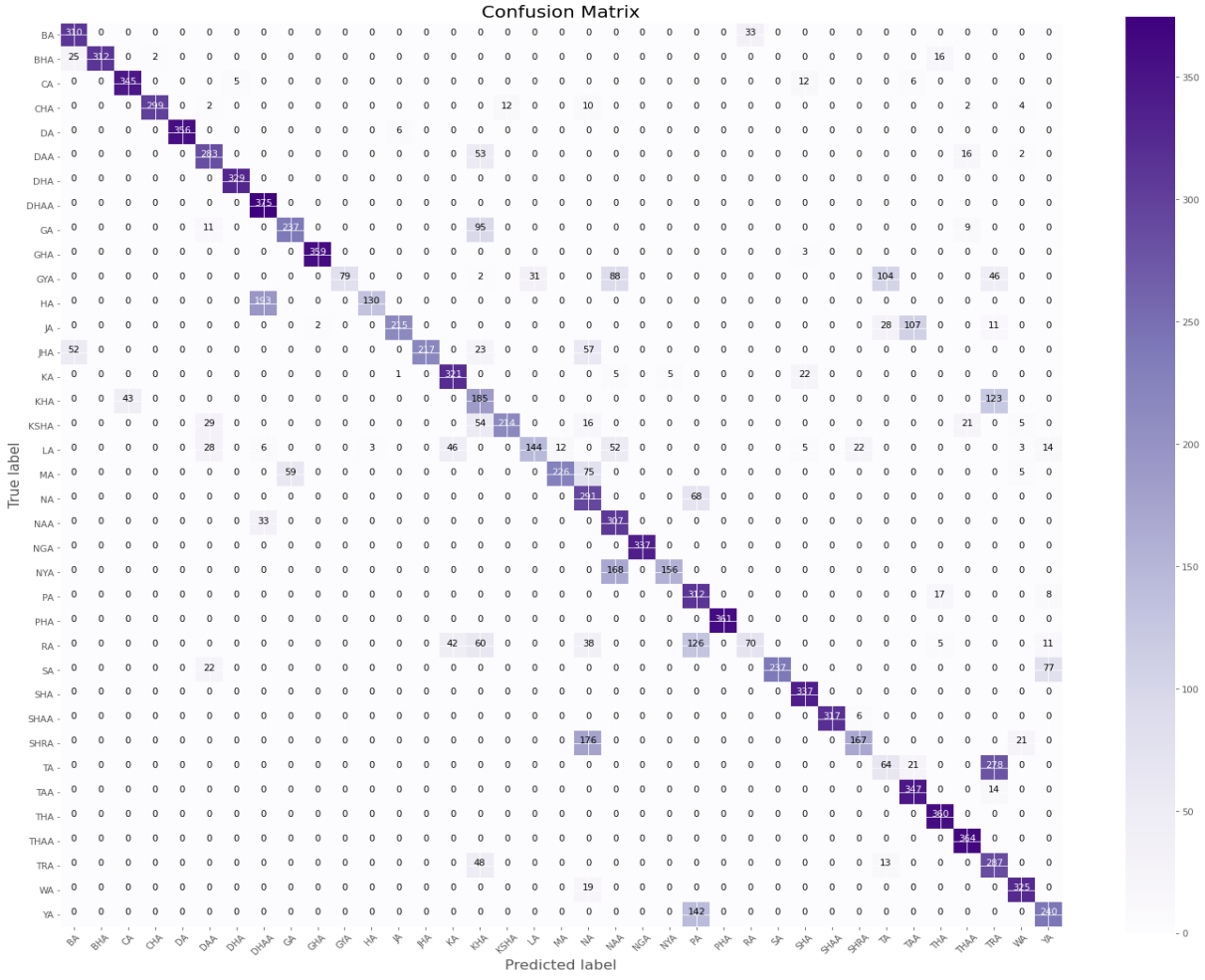


Figure 5.1: Confusion Matrix for evaluation data

A confusion matrix gives the summary of prediction results on a classification problem. Each row corresponds to an actual class and every column of the matrix corresponds to a predicted class. It is desirable that a diagonal is obtained across the matrix, which means that classes have been correctly predicted.

In the prediction we are getting the anomalies in some classes:SA , SHA, NYA, GYA, GA, SHRA, TA, THA, RA, WA, TRA, YA these classes were predicted wrongly.

Performance Metrics

Accuracy	Precision	Recall	F1 Score
Macro avg	82%	76%	75%
Weighted avg	81%	76%	75%

Table 5.2: Performance Metrics

5.3 Training Loss and Accuracy curve

The graph for training loss and validation accuracy is shown in Figure 5.3

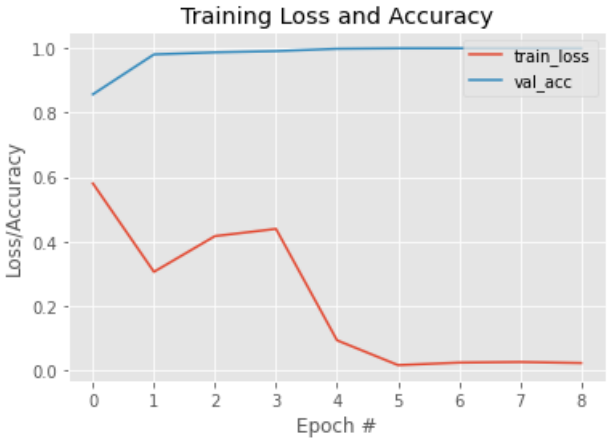


Figure 5.2: loss and accuracy curve

When visualizing the training loss and accuracy curves, w observe that due to the small batch size and low learning rate, the training loss and accuracy curve appears smooth and without jitter. When the loss and accuracy are compared, the loss is relatively low, which is preferable for the model. The accuracy is very high, which may be due to the similarity of gestures, and there is certainly less noise in the data.

5.4 Experiment with optimizers

While implementing our model, we have worked with multiple optimizers. The results obtained are as shown below.

Comparison Table

Optimizer	Test Accuracy	Evaluation Accuracy
Adam_5	100%	78.761%
Adam_10	100%	78.716%
RMSprop Epoch 5	99.982%	78.122%
RMSprop Epoch 10	100%	78.341%

Table 5.3: Comparision between Optimizers

5.5 Discussion

Processing

As we go on with the processing of the dataset and feeding into our model we've seen a slight issue that we get while performing that. Inspite of getting 75% accuracy our model still is not be able to recognize some of the alphabet gesture. This may be due to the effect of low resolution and the size which is 64x64 that compresses the image that may cause the loss of data. This may be enhanced by escalating the resolution to 128x128,because the image gets clearer and the attributes that are being used in CNN would be greatly improvised which may lead to good performance.

Testing

In testing process we've got accuracy of 100% in testing which is doubtful. This may be due to our dataset and the background. We've initially use the normal background and the size of the dataset is also minimum which is dreadful to the model. Hence we increase the size of the dataset and the position of hand . That slightly helps in acceptable the testing accuracy. Which conclude that our doubt was true, due to low size of dataset and it's resolution along with the position and structure of hand the testing accuracy is too high and we normalize it to it's satisfied state.

When we used RMSprop optimizer in our dataset of NSL, we obtained testing accuracy as 99.982% and evaluation accuracy as 78.122%. To verify our model architecture, we further tested and evaluated with ASL standard dataset. We found testing accuracy as 99.43% and evaluation accuracy 80%. The previous model for ASL recognition had

evaluation accuracy of about 98%. This showed that our architecture had some problems. Hence we modified our architecture. In our modified model, we were able to obtain 97.24% accuracy in evaluation and 97.72% accuracy in testing using the standard ASL dataset. After that we tested and evaluated in our own dataset in which we obtained 98% accuracy in testing and 82% accuracy in evaluation.

Issue in Real Time Prediction

When we test the model with entire image, we had issues to detect the hand gestures of NSL. But, we found the prediction to be drastically different when we tested in real-time using cropped image with background. While using cropped images, the model detected the hand gestures with certain level of precision. However, while using the entire image, due to the variation in background, the prediction was not so good. We believe this is likely due to complexity in our model caused by inclusion of images with variation in background. So we plan further to use the concept of region of interest (ROI) of managing ROI to predict signs from hand gestures in images with background.

6. Limitations and Future Enhancement

6.1 Limitations

Sign Language Recognition can be a challenging task due to the high number of attributes that should be considered for the accurate prediction. The major step in the prediction process is collection and preprocessing of the data which has already been fulfilled. We have also successfully tested and validated our model. However, our aim for the future research is increasing the prediction accuracy of our model successfully with various data sets.

Despite showing high accuracy while validation and testing, the model faces difficulties in predicting some of the alphabets. This is due to the more difficult nature of our classification with the inclusion of background in the images and the lower resolution, causing training to be more difficult.

We have also observed that model trained on pre-processed images performs much better than the model trained on the original images. Currently we have used 64*64 images for our model. We will now focus on improving our model by testing and validating it using images of size 128*128. We will also reconstruct our dataset by using red-glove approach.

6.2 Future enhancement

For the future enhancement of model we may use the high resolution images and the fine size of dataset and the background and diversity in the hand position and the red glove approach that will clearly distinguish the sign alphabet which can be recognized by the model. The main issue with our model that processing of data requires higher performance and capacity device.Hence for the further enhancement high GPU and RAM capcity devices may be use for processing that can resolve the high resoultion and large dataset processing issue.

7. Conclusion

In this project, we learned that how sometimes basic approaches work better than complicated approaches. Despite trying to use a complex approach we simply use the normal CNN to classify images .We also noticed the time constraints and difficulties of creating a dataset from scratch. Looking back,it would have nice to had the standard dataset that could greatly worked off. Some letters like SA , SHA, NYA, GYA, GA, SHRA, TA, THA, RA, WA, TRA, YA were harder to classify in our demo since due to the similarity in their gesture.Although our system work well,there is still bunch of scope for possible future work.

Hence, we conclude that Convolution Neural Networks can be used in image classification for sign language recogniton.However pre-training has to be performed with good resolution images and large dataset to accquire higher accuracy. We were able to accomplish 75% accuracy with 77770 dataset and having resoulution of 64x64 and the dataset splitted at 70% for training and 30% for testing,which is more than the accuracy acquired by previous literatures.

References

[1] G. Vasquez. Demand for deaf interpreters poses opportunities to learn asl. 2021.

[2] Sushila Sipai Drish Mali, Rubash Mali and Sanjeeb Prasad Pandey (PhD). Nepali sign language translation using convolution neural network. 2018.

[3] Acharya K. and Sharma D. Nepali sign language dictionary. page 209, 2003.

[4] Asl recognition using hand gestures. https://pdfcoffee.com/asl-recognition-using-hand-gestures–pdf-free.html. Accessed: 2022-05-16.

[5] Jhuma Sunuwar and Ratika Pradhan. Hand gesture recognition for nepali sign language using shape information. 3, 2015.

[6] Mujahid Abdullah, Awan Mazhar Javed, Yasin Awais, Mohammed Mazin Abed, Robertas Damaševičius, Rytis Maskeliūnas, and Karrar Hameed Abdulkareem. Real-time hand gesture recognition based on deep learning yolov3 model. *Applied Sciences*, 11(9):4164, 2021.

[7] Rasha Amer Kadhim and Muntadher Khamees. *A real-time american sign language recognition system using convolutional neural network for real datasets*, volume 9. UIKTEN-Association for Information Communication Technology Education and . . . , 2020.

[8] Abdulwahab A Abdulhussein and Firas A Raheem. Hand gesture recognition of static letters american sign language (asl) using deep learning. *Engineering and Technology Journal*, 38(6):926–937, 2020.

[9] Feng Wen, Zixuan Zhang, Tianyiyi He, and Chengkuo Lee. Ai enabled sign language recognition and vr space bidirectional communication using triboelectric smart glove. *Nature communications*, 12(1):1–13, 2021.

Appendices

Screenshots

i. Image collection process

3. Setup Folders

```
In [3]: IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')

In [4]: if not os.path.exists(IMAGES_PATH):
            if os.name == 'posix':
                !mkdir -p {IMAGES_PATH}
            if os.name == 'nt':
                !mkdir {IMAGES_PATH}
        for label in labels:
            path = os.path.join(IMAGES_PATH, label)
            if not os.path.exists(path):
                !mkdir {path}
```

4. Capture Images

```
In [5]: IMAGES_PATH
Out[5]: 'Tensorflow\\workspace\\images\\collectedimages'

In [7]: labels=['SHRA']
        number_imgs = 200

In [8]: for label in labels:
            cap = cv2.VideoCapture(0)
            print('Collecting images for {}'.format(label))
            time.sleep(0)
            for imgnum in range(number_imgs):
                print('Collecting image {}'.format(imgnum))
                ret, frame = cap.read()
                imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
                cv2.imwrite(imgname, frame)
                cv2.imshow('frame', frame)
                time.sleep(0)

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
        cap.release()
        cv2.destroyAllWindows()
```

Figure 7.1: Image collection process

ii. Sample Dataset

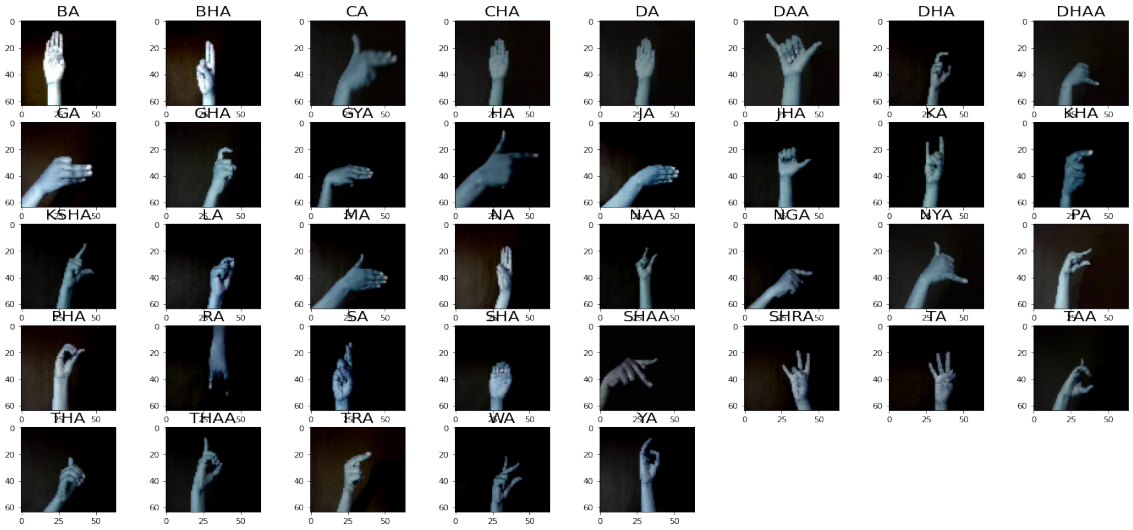


Figure 7.2: Evaluation Images

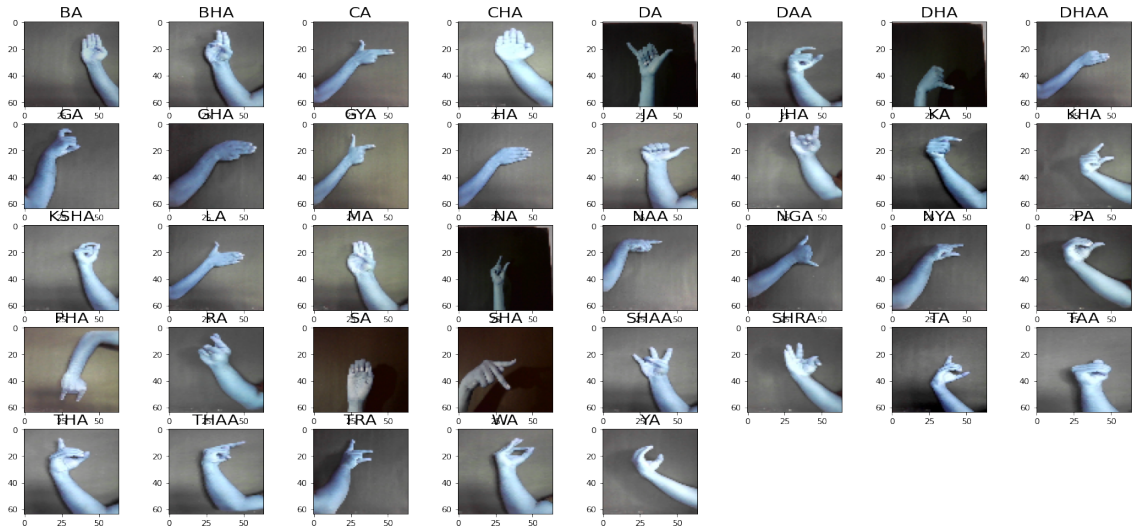


Figure 7.3: Training images

iii. Real Time Prediction

9. Prediction

```
# class_names=['KA','KHA','GA','GHA','NGA','CA','CHA','JA','JHA','NYA','TA','THA','DA','DHA','NA','TAA','THAA','DAA','DHAA','NAA
class_names=['BA','BHA','CA','CHA','DA','DAA','DHA','DHAA','GA','GHA','GYA','HA','JA','JHA','KA','KHA','KSHA','LA','MA','NA','NA
from keras.preprocessing import image
# predicting images
img = keras.utils.load_img('C:/Users/atulp/OneDrive/Desktop/DATA/Tensorflow/workspace/images/collectedimages/DA/DA.3a01d899-ff9a-
x = keras.utils.img_to_array(img)
x=x.astype('float32')/255.0
x = np.expand_dims(img, axis=0)
```

```
x.shape
```

```
(1, 64, 64, 3)
```

```
classes = model.predict(x, batch_size=64)
print(classes)
class_names[np.argmax(classes)]
```

```
1/1 [==============================] - 0s 17ms/step
[[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

```
'DA'
```

Figure 7.4: Process to predict in real time

iv. Confusion Matrix

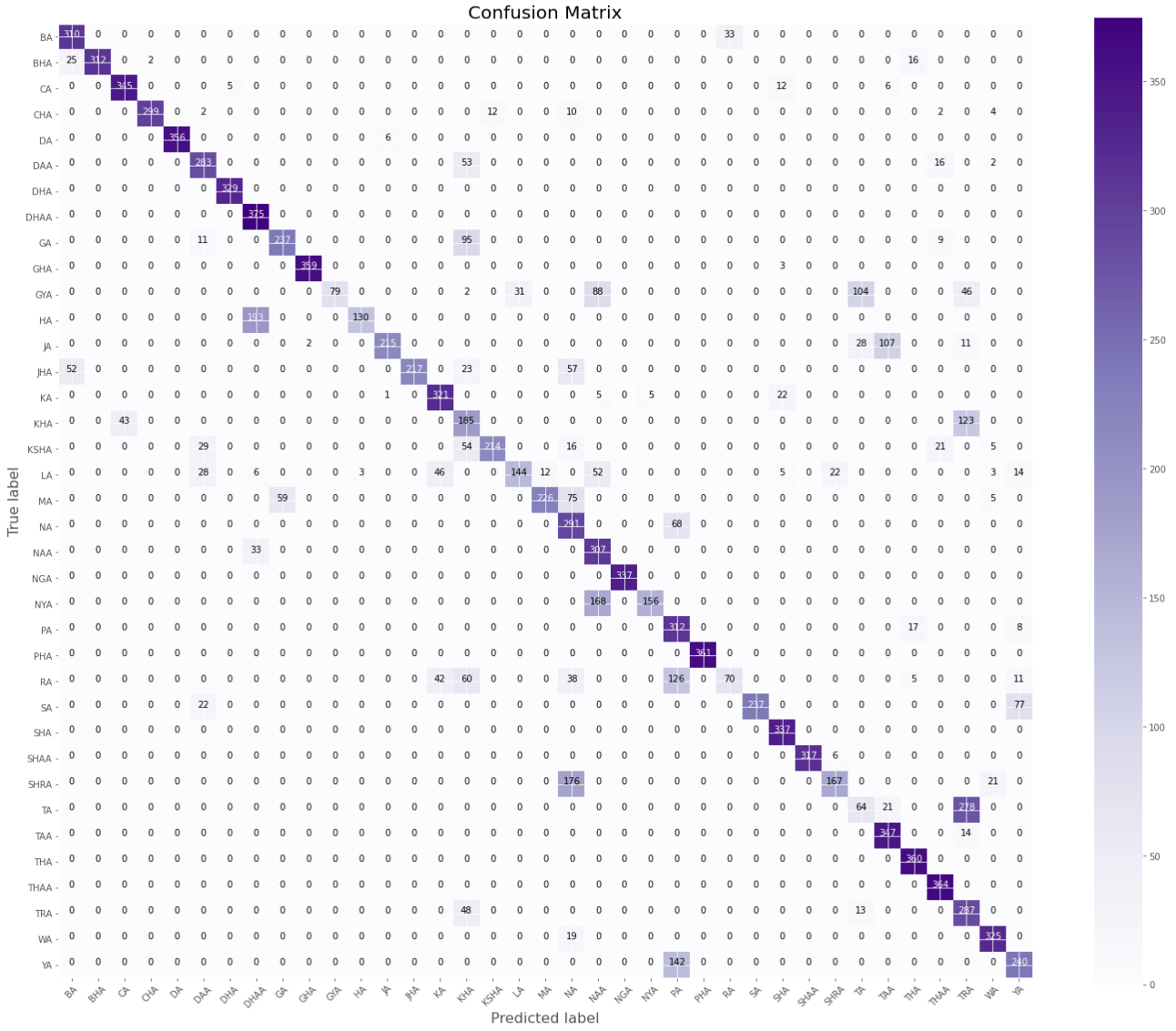


Figure 7.5: Confusion Matrix for Evaluation

v. Testing Accuracy

```
score = model.evaluate(X_test, y_test, verbose = 0)
print('Accuracy for test images:', round(score[1]*100, 3), '%')
```

[16]

```
Accuracy for test images: 99.561 %
```

+ Code + Markdown

Figure 7.6: Testing accuracy

vi. Evaluation Accuracy

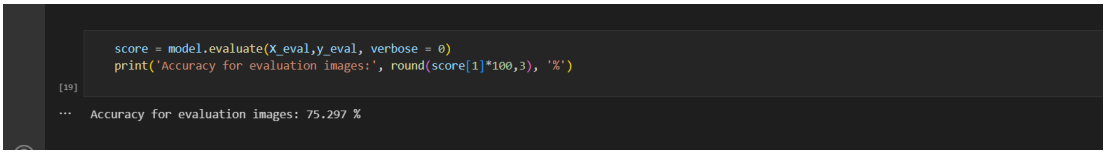
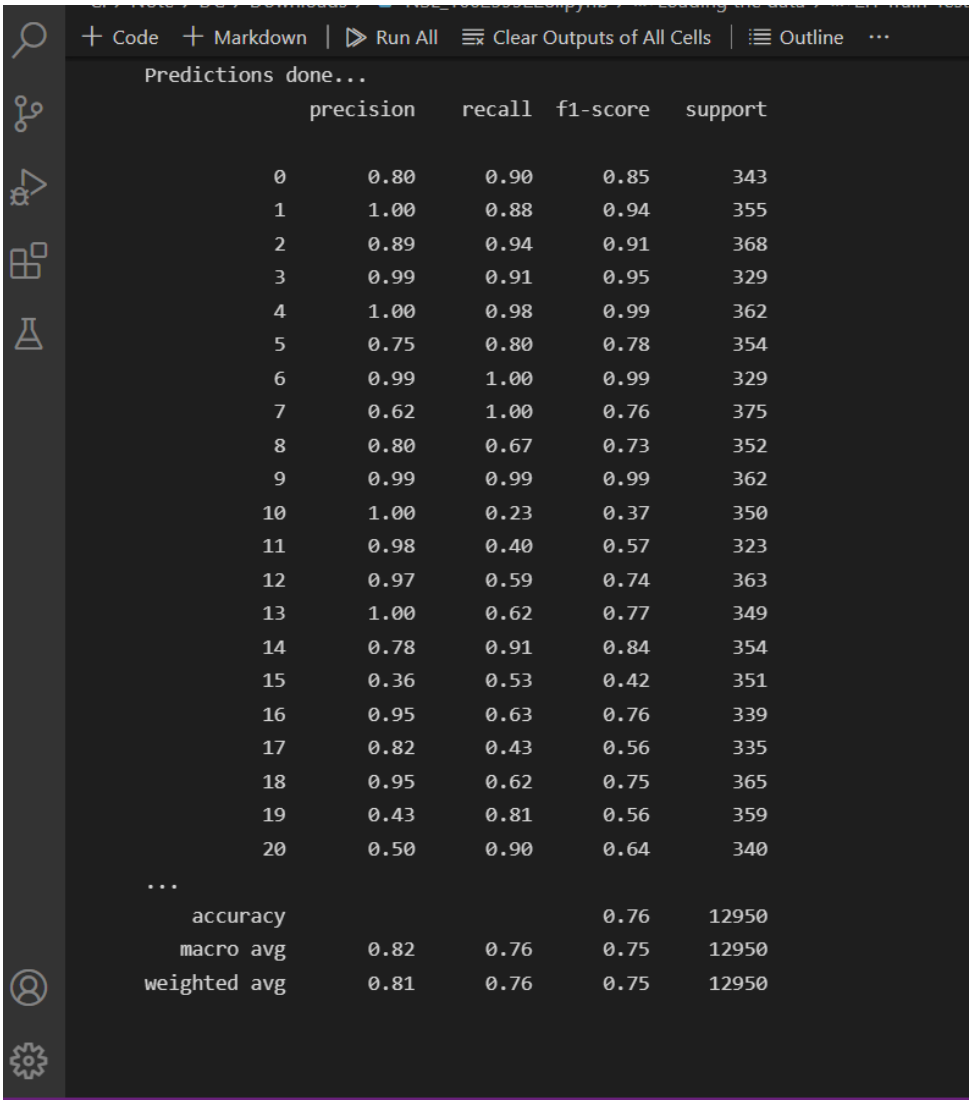


Figure 7.7: Evaluation accuracy

vii. Precision Recall and F1 Score



	precision	recall	f1-score	support
0	0.80	0.90	0.85	343
1	1.00	0.88	0.94	355
2	0.89	0.94	0.91	368
3	0.99	0.91	0.95	329
4	1.00	0.98	0.99	362
5	0.75	0.80	0.78	354
6	0.99	1.00	0.99	329
7	0.62	1.00	0.76	375
8	0.80	0.67	0.73	352
9	0.99	0.99	0.99	362
10	1.00	0.23	0.37	350
11	0.98	0.40	0.57	323
12	0.97	0.59	0.74	363
13	1.00	0.62	0.77	349
14	0.78	0.91	0.84	354
15	0.36	0.53	0.42	351
16	0.95	0.63	0.76	339
17	0.82	0.43	0.56	335
18	0.95	0.62	0.75	365
19	0.43	0.81	0.56	359
20	0.50	0.90	0.64	340
...				
accuracy			0.76	12950
macro avg	0.82	0.76	0.75	12950
weighted avg	0.81	0.76	0.75	12950

Figure 7.8: Precision Recall and F1 Score