Ministry of High Education

El Obour High Institute for

Management & Information

Sign Language Interpreter

By Machine learning

مشروع تخرج استيفاءاً لمتطلبات الحصول على درجة البكالوريوس

Prepared by "CS-07"

1-Radwan Kamal Mohamed

2-George Magdy Fayez

3-Mohamed Ahmed Mohamed Eliwa

4-Kerelus Moner Erian

Supervised by

Prof. Dr. Ahmed Said

Ass.Lec. Reham Elshamy

4th year Computer Science

Graduation Year:(2020/2021)

ACKNOWLEDGMENT

In the beginning, we would like to extend all the thanks that the pen cannot express to the administration of the Obour Institute, for what they have given us to complete this project in the best possible form.

We also extend our sincere thanks, appreciation and unlimited gratitude to

Prof. Dr. Ahmed Saeed and Ass.Lec. Reham Elshamy for supervise this project and help us in every possible way and to benefit from their experience and knowledge to complete this project in the best form.

Abstract:

We all have at least one person with special needs (special people).

This project concerns deaf and dumb as we find difficulty in dealing with them and understand their needs or translate their sign language.

So, we decided to make a simple program to convert sign language to our speech to ease our communication with them.

The user will use this program to insert a picture of sign language and the system will convert it to text to understand their need and ease our communication between us and we don't have to learn sign language. So, by making this program, we will help them and help us to understand each other and for them to avoid embarrassment.

Sign language is used by 70 million people around the world and there's a lack of communication between deaf and dump community and our community. The deaf and dump community face many problems while communicating. When it comes to the interactive part with people with special needs, this communication gap has exacerbated because our community has no idea about sign language.

Sign-to-text is a program that translates sign language to text by inserting the picture of sign language and there is a model that understands the sign and turns it to text.

Contents

Chapter 4 “Design & Implementation”

Chapter 5 “Conclusion and Future work”

Chapter 1

INTRODUCTION

1.1. What is sign language?

Sign language is a visual language that is used by deaf people as their mother language and there are many different dialects and different signs depending on the geography of humans .

Unlike acoustically conveyed sound patterns, sign language uses body language and manual communication to fluidly convey the thoughts of a person. It is achieved by simultaneously combining hand shapes, orientation and movement of the hands, arms or body, and facial expressions. It can be used by a person who has difficulties in speaking or by a person who can hear but could not speak and by normal people to communicate with hearing disabled people.

Sign languages (also known as signed languages) are languages that use the visual-manual modality to convey meaning.

Sign languages are expressed through manual articulations in combination with non-manual elements. Sign languages are full-fledged natural languages with their own grammar and lexicon.

Sign languages are not universal and they are not mutually intelligible with each other, although there are also striking similarities among sign languages.

Linguists consider both spoken and signed communication to be types of natural language, meaning that both emerged through an abstract, protracted aging process and evolved over time without meticulous planning.

Sign language should not be confused with body language, a type of nonverbal communication.

Wherever communities of deaf people exist, sign languages have developed as useful means of communication, and they form the core of local Deaf cultures. Although signing is used primarily by the deaf and hard of hearing.

It is also used by hearing individuals, such as those unable to physically speak, those who have trouble with spoken language due to a disability or condition (augmentative and alternative communication), or those with deaf family members, such as children of deaf adults.It is unclear how many sign languages currently exist worldwide.

Each country generally has its own native sign language, and some have more than one.

The 2021 edition of Ethnologue lists 150 sign languages, while the SIGN-HUB Atlas of Sign Language Structures lists over 200 of them and notes that there are more which have not been documented or discovered yet.

Some sign languages have obtained some form of legal recognition.

Linguists distinguish natural sign languages from other systems that are precursors to them or obtained from them, such as invented manual codes for spoken languages, home sign, "baby sign", and signs learned by non-human primates.

1.2. History of sign language:

Groups of deaf people have used sign languages throughout history. One of the earliest written records of a sign language is from the fifth century BC, in Plato's Cratylus, where Socrates says: "If we hadn't a voice or a tongue, and wanted to express things to one another, wouldn't we try to make signs by moving our hands, head, and the rest of our body, just as dumb people do at present?"

Until the 19th century, most of what is known about historical sign languages is limited to the manual alphabets (fingerspelling systems) that were invented to facilitate transfer of words from a spoken language to a sign language, rather than documentation of the language itself. Pedro Ponce de León (1520–1584) is said to have developed the first manual alphabet.

1.3. Deaf communities and deaf culture:

When Deaf people constitute a relatively small proportion of the general population, Deaf communities often develop that are distinct from the surrounding hearing community. These Deaf communities are very widespread in the world, associated especially with sign languages used in urban areas and throughout a nation, and the cultures they have developed are very rich.

1.4. Interpretation:

In order to facilitate communication between deaf and hearing people, sign language interpreters are often used. Such activities involve considerable effort on the part of the interpreter, since sign languages are distinct natural languages with their own syntax, different from any spoken language.

With recent developments in artificial intelligence in computer science, some recent deep learning based machine translation algorithms have been developed which automatically translate short videos containing sign language sentences (often simple sentences consisting of only one clause) directly to written language.

1.5. Problem Definition:

As we said, sign languages are the only medium through which the educated deaf and mutes communicate all over the world. The World Health Organization states that over 5% of the world's population are a part of this category. However, most of the people blessed with the ability to hear and speak are not well versed with this sign language, thus leading to communication gaps. This system aims to bridge this communication gap and aid the deaf and the mute to use technology to carry out their daily transactions by using a simple approach which is easily implementable.

without the middle person who generally acts as a medium of translation.

Language endangerment and extinction,As with any spoken language, sign languages are also vulnerable to becoming endangered.

For example, a sign language used by a small community may be endangered and even abandoned as users shift to a sign language used by a larger community, as has happened with Hawai'i Sign Language, which is almost extinct except for a few elderly signers.

Even nationally recognised sign languages can be endangered; for example, New Zealand Sign Language is losing users.

Methods are being developed to assess the language vitality of sign languages.

1.6. Motivation:

- 7.5 million people are deaf and dumb in Egypt .. more than 360 Million people (5%) of the world's population suffers from deafness
- 10 Million Egyption hear at 40 decibels (dB) or higher. "Normal" hearing ranges from 0 to 20 dB.
- more than 70 Million Deaf people around the world use sign language
- 98% of Deaf people in this world do not receive education in sign language.
- 1.1 billion young people (aged 12-35 years) are at risk of hearing loss due to noise pollution
- 72% of families do not sign with their Deaf children.
- 70% of Deaf people don't work or are underemployed.
- 1 in 4 Deaf people have quit a job due to discrimination

The importance of this program in motivating the next generation of young people, which motivates us to change and strive to develop and transform it from a mere to a glopal product that is available on the ground and Helping this group of people to obtain appropriate education and appropriate jobs without discrimination

1.7.objective:

We intend to achieve equality by disseminating information about deaf culture and breaking down barriers and misconceptions.

Another goal of this project is to promote Arabic as a second language for the deaf.

1. In this project we want to translate sign language into text. The framework provides a helping-hand for speech-impaired to communicate with the rest of the world using sign language.
2. This leads to the elimination of the middle person who generally acts as a medium of translation. This would contain a user-friendly environment for the user by providing text output for a sign gesture input.
3. Help the hearing and speaking impaired to communicate with people who do not understand sign language.
4. Helps the deaf and mute to carry out their daily transactions without relying on an interpreter.
5. Broaden the use of technology to solve social problems.
6. Helping deaf and dumb people to obtain appropriate education and appropriate work without discrimination or rejection.

Chapter 2

Background

2.1. Languages and technologies we use it :

2.1.1. python

What is python ?

Python is a high-level programming language that is simple to write and read, easy to learn, open source programming, and scalable. Python is a multi-purpose, multi-purpose language that is widely used in many fields, such as building standalone programs using known graphical interfaces and running web programs, using odoo programming, and using it as a scripting language to control the performance of some of the most popular programs or build Attached programs.
In general, Python can be used to program simple programs for beginners, and to accomplish large projects like any other programming language at the same time. Programming beginners are often advised to learn this language because it is among the fastest learning software languages.

2.1.2. keras & tensorflow

Keras is an open-source neural-network library written in Python. It can run on top of TensorFlow. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine.

It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

However, Keras is also a highly-flexible framework suitable to iterate on state-of-the-art research ideas. Keras follows the principle of progressive disclosure of complexity: it makes it easy to get started, yet it makes it possible to handle arbitrarily advanced use cases, only requiring incremental learning at each step.

2.1.2.1. What is TPU ?

A tensor processing unit (TPU) is an AI accelerator application specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning, particularly using Google's own TensorFlow software.

It is a symbolic math library which is used for machine learning applications such as neural networks. However, Google still uses CPUs and GPUs for other types of machine learning.

2.1.2.2. What is a GPU ?

A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing.

PU die.

2.1.2.3. GPU vs TPU ?

GPUs are extremely efficient at matrix multiplication, which basically forms the core of machine learning. The strength of GPU lies in data parallelization, which means that instead of relying on a single core, as CPUs did before, a GPU can have many small cores. A single GPU can have thousands of Arithmetic Logic Units or ALUs, each performing a parallel computation to return a higher throughput faster, thus running complex computations in a short span of time. This makes GPUs highly suitable for performing complex matrix multiplications in a short span of time.

Nvidia's most powerful Tesla V100 data center GPU for instance, features 640 tensor cores and 5,120 CUDA cores. This means that at the peak of its tensor throughput, it can produce a staggering 125 TFLOPS.

Thanks to that, it offers a dramatically high precision rate resulting in a tighter machine learning system with very high accuracy. Even if in your project, you might not necessarily use a Tesla V100, you now have a general idea of how robust and precise the GPU performance can be when it comes to analyzing extremely large datasets and accurately training your machine learning model.

In terms of memory and cache, the best GPUs offer a shared memory subsystem that can be shared across all different cores of the GPU. The L1 cache ensures swift performance, reducing latency and optimizing memory usage.

GPUs are optimized for the CUDA software technology that was specifically designed for parallel computing and independent multi-threading operations. This offers ample flexibility for the programmers and keeps things simple.

TPUs When it comes to machine learning and deep neural networks, Google is the leader of the pack.

The fact that Google Photos accurately identifies the photos of one person from the time they were just a day old to when they were playing soccer in the park, or how Gmail almost always has the perfect auto-responses to emails, is a testament to Google's deep learning prowess.

While Google itself used Nvidia GPUs on an Intel Xeon CPU for the longest time, they have now truly jumped into the hardware market with their custom-made tensor processing units or TPUs.

Google has been using these TPUs in-house on their applications like Google Photos, Google Translate, Google speech recognition, Gmail, Google assistant and many more for a few years now. They recently made it possible for other developers to use the TPU by renting it through the cloud. They are still not planning on selling the hardware itself.

2.1.3. Kaggle

Kaggle: is a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

The Kaggle community is growing fast. There are currently over one million Kaggle members (Kagglers). This data community has submitted above four million learning models to different competitions. Kaggle users have shared over one thousand datasets, more than 170,000 forum posts and over 250 kernels.

According to the founder, this incredibly fast growth can be attributed to high-quality content, data and code shared by Kagglers.

Kaggle's services:

1. Machine learning competitions: this was Kaggle's first product. Companies post problems and machine learners compete to build the best algorithm, typically with cash prizes.

2. Kaggle Kernels: a cloud-based workbench for data science and machine learning. allows data scientists to share code and analysis in Python, R and R Markdown.

3. Over 150K "kernels" (code snippets) have been shared on Kaggle covering everything from sentiment analysis to object detection.

Public datasets platform: community members share datasets with each other has datasets on everything from bone x-rays to results from boxing bouts.

4. Kaggle Learn: a platform for AI education in manageable chunks.

The courses are free and you can earn certificates.

Some of the courses:

(Python-Intro to Machine Learning - SQL - Natural Language Processing)

2.2.Create dataset :

Arabic letters in Sign language

There is a wide range of datasets for automatic image description search. The images in these datasets are related to text descriptions and differ from each other in such aspects as the size, the format of the descriptions, and how the descriptions are collected.

This dataset is a complete set of gestures that are used in sign language and can be used by other lay people to better understand the sign language gesture.

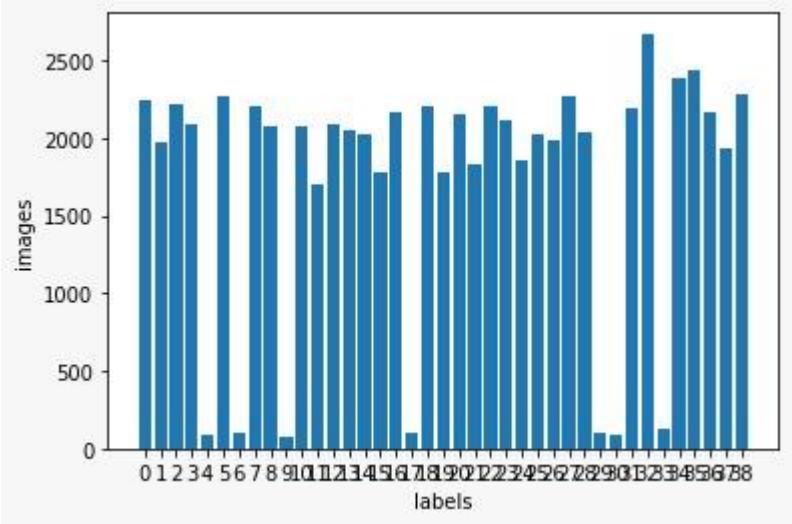
We are looking for the existing dataset, and did not find anything about the relevant sign language dataset for Arabic

We found some information about the dataset in English but we made a data set in Arabic according to Dr. Ahmed Saeed's instructions. Because we did not find a data set for the Arabic language on the internet,We bought a camera to make the operation faster and more accurate.

The dataset is useful for developing applications and devices in the field of assistive technology for the benefit of the deaf and hard of hearing. Examples of relevant data sets can be found in the references.

The dataset is unique in the sense that it is the first large comprehensive dataset for Arabic sign language.

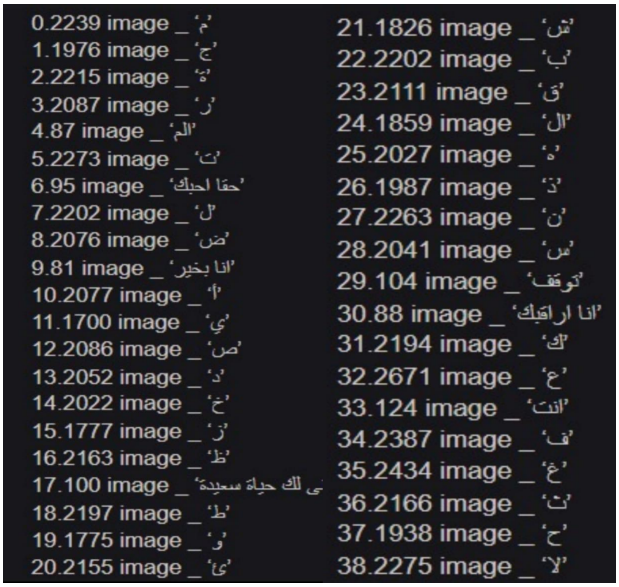
There is great potential for this data set to be used by researchers to increase classification and identification accuracy and to develop useful prototypes for the deaf community.



We captured 68,132 images.The dataset contains a single volume consisting of color images of hands of different gestures At a rate of 2000-2500 images per letter The dataset contains 32 Arabic characters,And 100-150 images per word,we have 7 words.

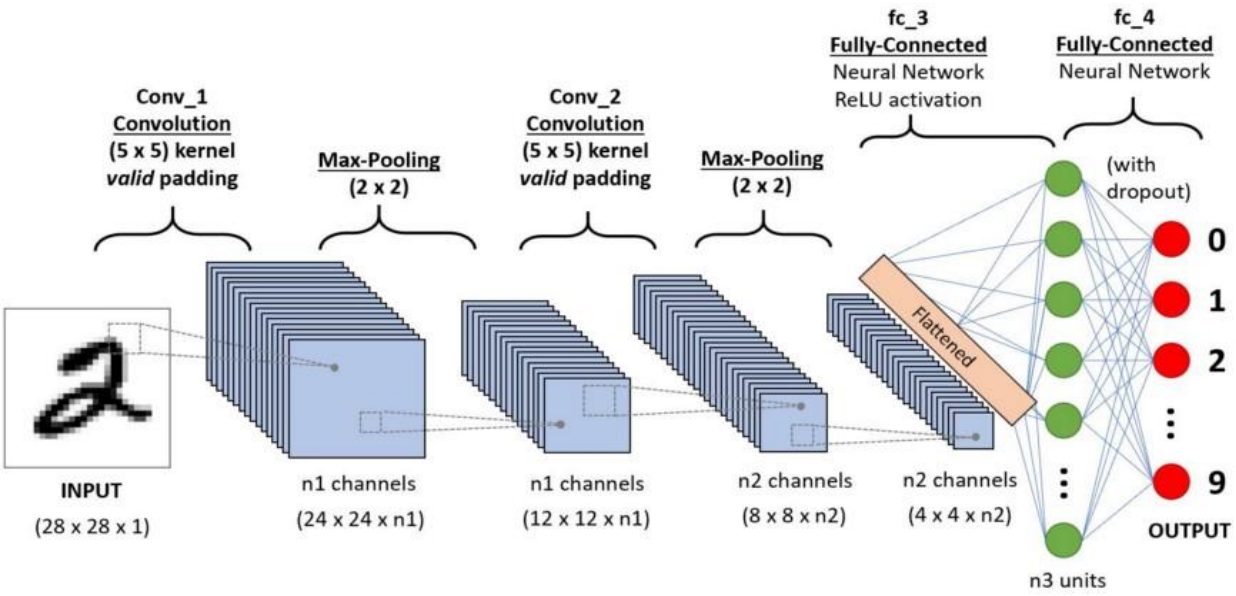
The background of the images and the hands used for pointing has been changed so that the data set is different so that we can extract a high resolution that is used in more than one position.

We pre-processed all the image by limited the size of the images to 64X64 in order to reduce the area of the images and save time and capabilities while training the model, and we also used the grayscale function to convert images from RGB to gray to reduce the colors in the background and focus the signal



2.3. Background of CNNs:

CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.



2.3.1 What exactly is a CNN?

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery.

CNNs are powerful image processing, artificial intelligence (AI) that uses deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition, along with recommender systems and natural language processing (NLP).

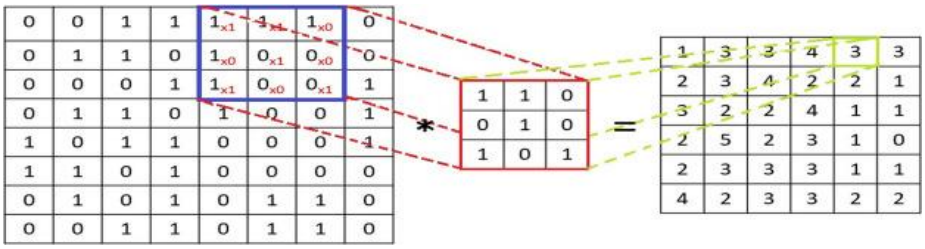
A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces.

CNNs 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.

2.3.2. Convolutional layer :

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map



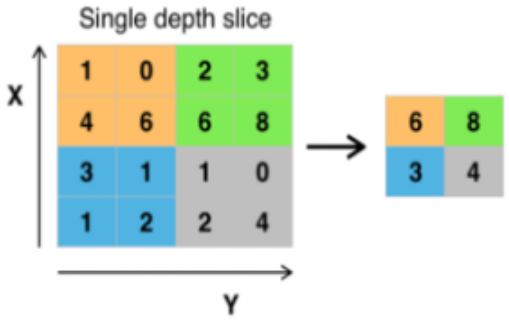
Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter.

This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

2.3.3. Pooling layers :

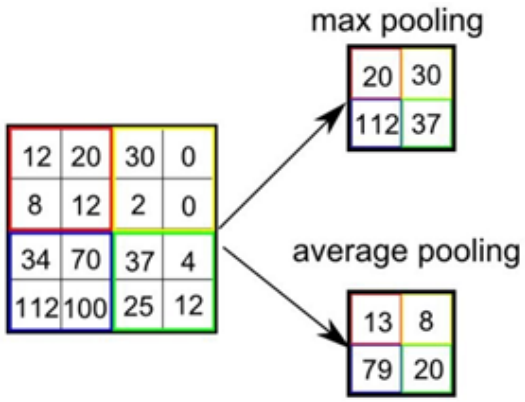
Pooling layers, also known as downsampling,conducts dimensionality reduction by combining the outputs of neuron clusters at one layer into a single neuron in the next layer, reducing the number of parameters in the input. Similar to the convolutional layer, Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. The pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights, populating the output array. There are two main types of pooling:



Max pooling with a 2x2 filter and stride = 2

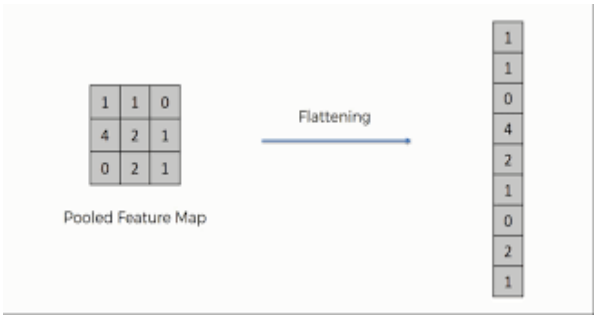
Max pooling: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.

Average pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.



While a lot of information is lost in the pooling layer, it also has a number of benefits to CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

2.3.4. Fully connected layer :



The output feature maps of the final convolution or pooling layer is typically flattened , i.e., transformed into a one-dimensional (1D) array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight.

Once the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks.

The final fully connected layer typically has the same number of output nodes as the number of classes.Each fully connected layer is followed by a nonlinear function, such as ReLU, as described above.

2.3.5. Neural Networks VS CNN:

Points of comparison	Neural Networks	Convolution Neural Networks
Structure and Definition	Works based on the structure and functions of a human brain. A neural network has interconnected artificial neurons that transmit data among each other are called nodes.	It is a subclass of neural networks which has at least one convolution layer.
Layers	A typical neural network consists of 3 layers: 1. Input layer: accepts inputs in different forms. 2. Hidden layer: transform the inputs and do several calculations and feature extractions using fully connected layers. 3. Output layer: produces the desired output.	Convolution operation forms the core of every convolution neural network: 1.Convolution layer. 2.Activation Function. 3.Pooling. 4.Fully Connected Layer.

Differences	Each neuron is connected to every other neuron.	Only the last layer of a CNN is fully connected.
Advantages	capable of learning any nonlinear function. They have the capacity to learn weights that map any input to the output.	• CNN learns the filters automatically without mentioning it explicitly. These filters help in extracting the right and relevant features from the input data • CNN captures the spatial features from an image. Spatial features refer to the arrangement of pixels and the relationship between them in an image.
disadvantages	• The number of trainable parameters increases drastically with an increase in the size of the image. • ANN loses the spatial features of an image. Spatial features refer to the arrangement of the pixels in an image.	CNN does not encode the position and orientation of object Lack of ability to be spatially invariant to the input data.

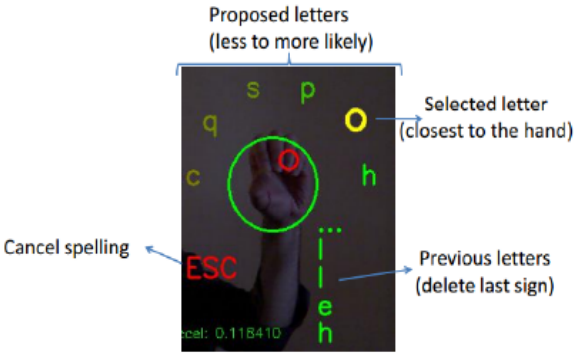
Chapter 3

Previous Work

Sign language recognition has seen a major breakthrough in the field of Computer Vision in recent years . A detailed review of sign language recognition models can be found in The challenges of developing sign language recognition models range from the image acquisition to the classification process. We present a brief review of some :related models of sign language

3.1.Sign Mobiles: Android Application

The sign mobile application makes use of three technologies- Video Relay Service (VRS), Outfit 7 and Mimix, inorder convert the sign language into textual and audible output.



It makes use of mobile gesture recognition and also supports remote text-to-sign conversion and vice versa. However it is an extremely complex application from the developer's point of view and is also platform dependent .

3.2. Sign Language Rings

Sign Language Ring is designed by a group of students from the Asian University and is the recipient of the Red Dot Design Award The Buddhist prayer beads are an inspiration for this device. Though conceptual, it shows a great future potential.

This device comprises 3 rings on each hand and a bracelet. The rings have motion sensors which are responsible to track the movement of the fingers of the person wearing them and for translating the sign language depicted to spoken language.

The word in the spoken language is then displayed on the LED Screen present on the bracelet and is also played by the speaker and amplified by the microphone. Both the speaker and the microphone are a part of the bracelet.

3.3. EnableTalk: Sensor Enabled Gloves

(a) Leap motion sensor.

(b) Hand glove for hand gesture recognition.

The EnableTalk Gloves, developed by a group of students in Ukraine, are fitted with a variety of sensors to detect the position of the fingers and palm in space.
This data is transmitted through the controller fitted on the back of the glove, via Bluetooth to a mobile device. The mobile device contains the Microsoft Speech API and Bing API in order to convert the sign language to text and speech.

A major disadvantage of this system is its cost. It also requires a variety of hardware components to bring about the translation.
Data gloves and Vision based methods are commonly used to interpret gestures for human computer interaction. The sensors attached to a glove finger flexion into electrical signals for determining the hand posture in the data gloves method .The vision based method reduces the difficulties in the glove based method.

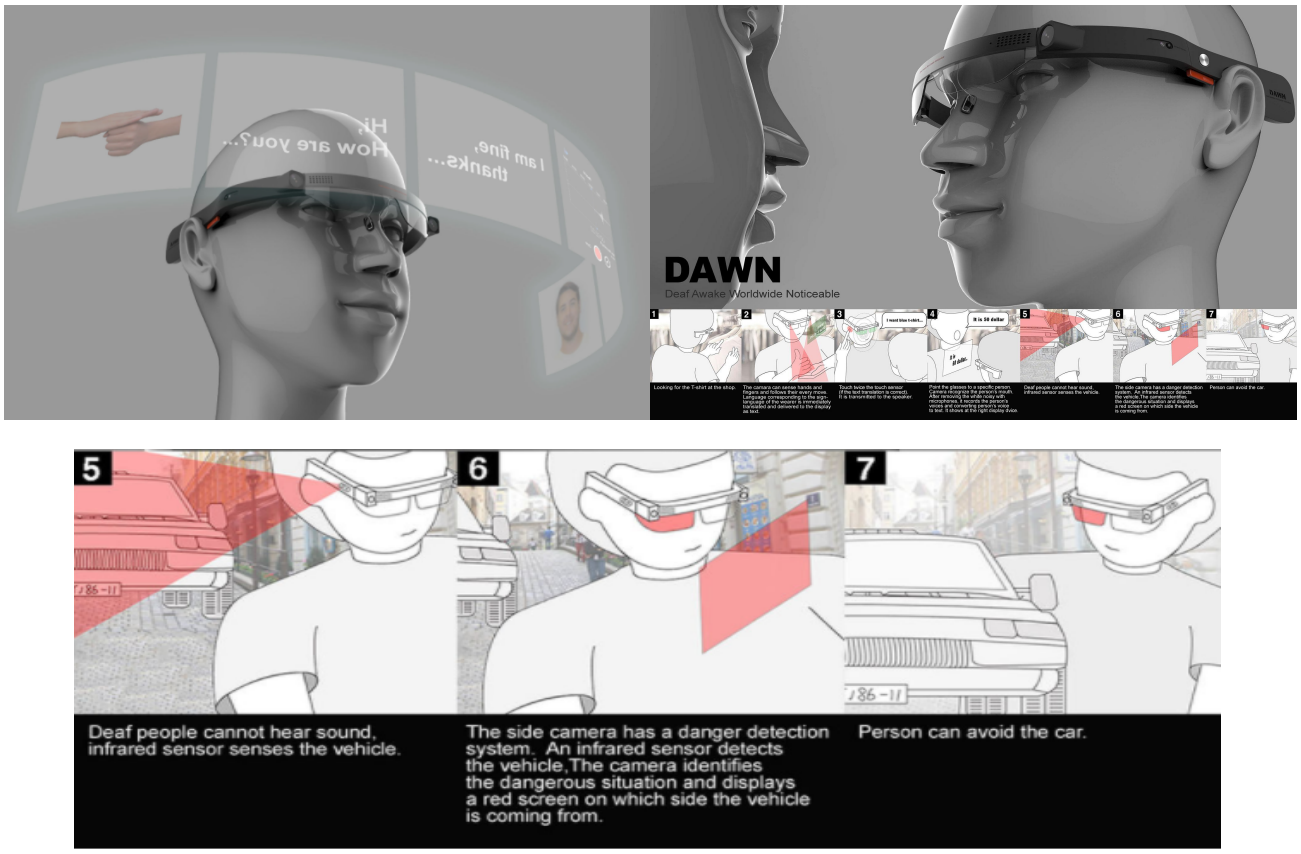
3.4.SensEar Glasses.

is glasses that can help deaf-mute people communicate with others. A screen appears in front of you on a transparent screen, appears in front of you on a screen that appears in expression by means of dialogue, virtual reality and start communication between deaf and dumb and the rest of the people.

The owner of the innovation is the Egyptian student “Omar Abdel Salam”, a student in the Department of Communications and Computers, Faculty of Engineering at Mansoura University.

They were behind the creation of these amazing glasses, through which people of determination can adapt to the environment around them.

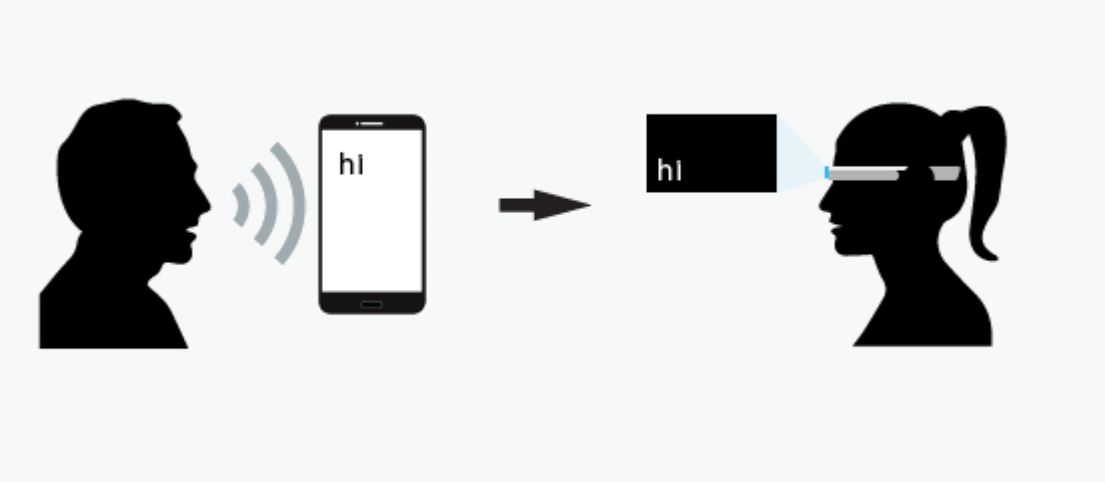
3.5. DAWN's glasses



DAWN's glasses can change the lifestyle of the deaf. It is a wearable device for the deaf and hard of hearing. and it has intelligence to help out the deaf. The glasses can help in communicating with people because they convert sign language into text and voice, and they can also record voice and convert it into text.

The glasses also contain a danger detection system that helps the deaf in dangerous situations, such as warning of a car heading towards it as a result of not hearing the sound of the car. The glasses warn him

3.6.google glasses for hearing impaired



Captioning on Glass (CoG) provides real-time captioning, allowing the deaf and hard of hearing to converse with others. Your conversational partner speaks into the phone; your speech is nearly instantly converted to text and displayed on your Glass.

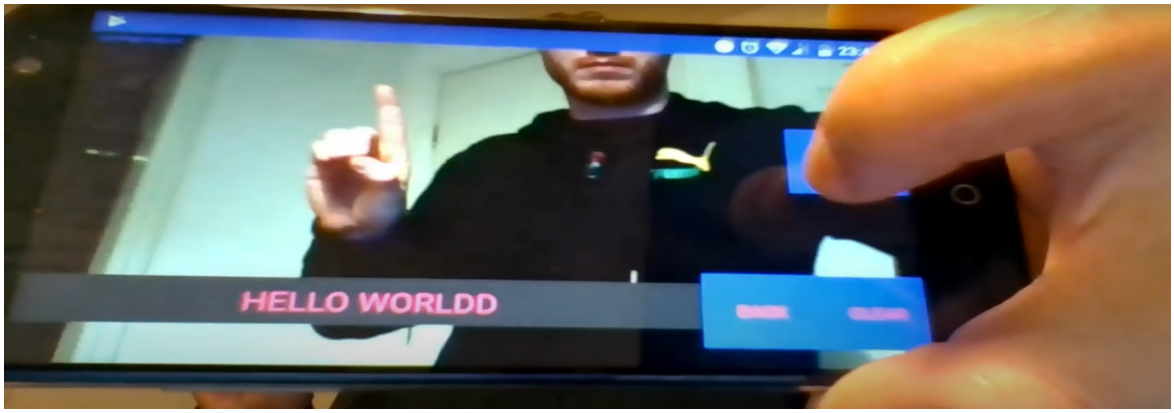
A hearing impaired person wearing Google Glass can download the app on his/her Glass from MyGlass. He/She can then download the app that helps pair Glass with nearby Android devices. The wearer can also choose to pair Glass with Android devices with which he/she has previously communicated. Once the devices are paired, the speech of the other person will appear in the form of text on Glass, when they speak directly into their smartphones.

Glass basically converts the voice into text and displays it on its screen. Once a device is paired with Glass, it will automatically detect the device when it is in the vicinity of the wearer.

3.7. ASL Sign Language Recognition App Live Demo (OpenCV):

The Sign Language app is an Android application which can translate static ASL and BSL signs, which uses feature extraction algorithms and machine learning (SVM) to recognise and translate static sign language gestures. such as the fingerspelling alphabet.

These translated signs can be displayed to the user whilst allowing for sentences to be constructed. This app is currently a proof of concept to illustrate low-cost, freely available and offline Sign Language recognition using purely visual data.
This project centred around developing an efficient, offline strategy for recognising visual sign language using an Android application. The end prototype created was capable of determining ASL and BSL alphabet phonemes at 90+% accuracy.



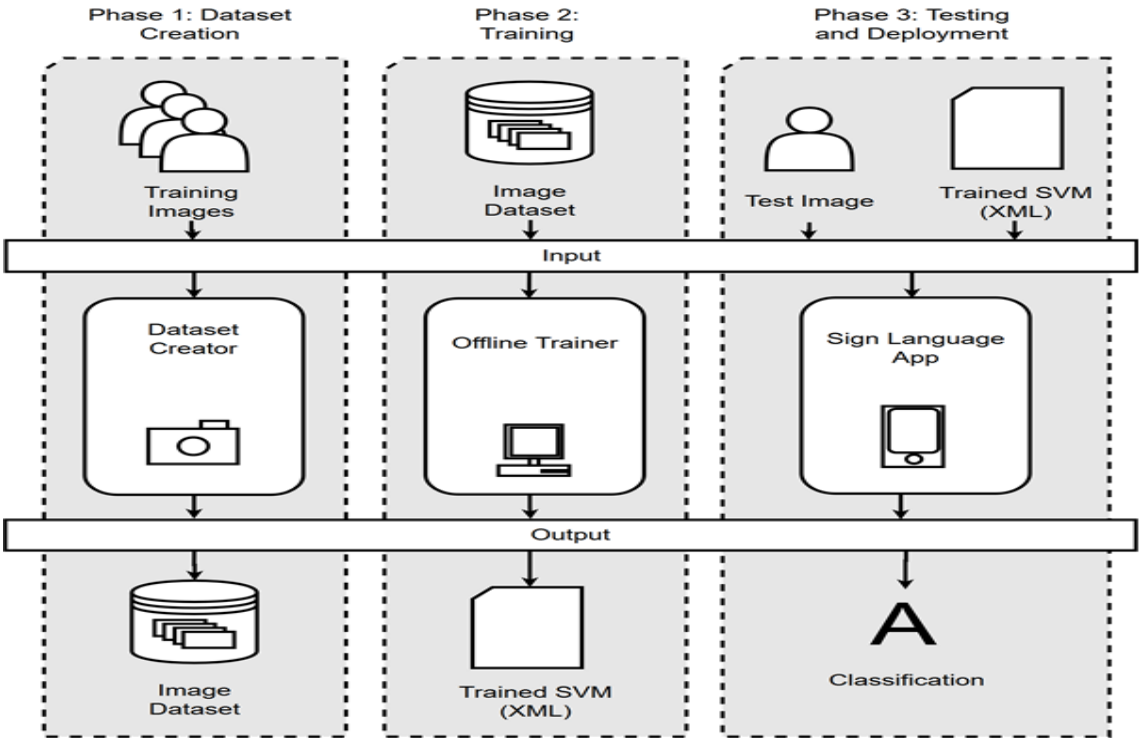
Built With: languages >c++ 65.2% \ Java 26.8% \ CMake 4.4% \ C 3.3% \ HTML 0.1% \ Objective-C 0.1% \ Makefile 0.1%

3.7.1. Dataset Creator:

Simple python script for creator Machine Learning dataset images, either manually or automatically, from your webcam. This script was created for use with the below repositories, but can be utilised for generic ML projects.

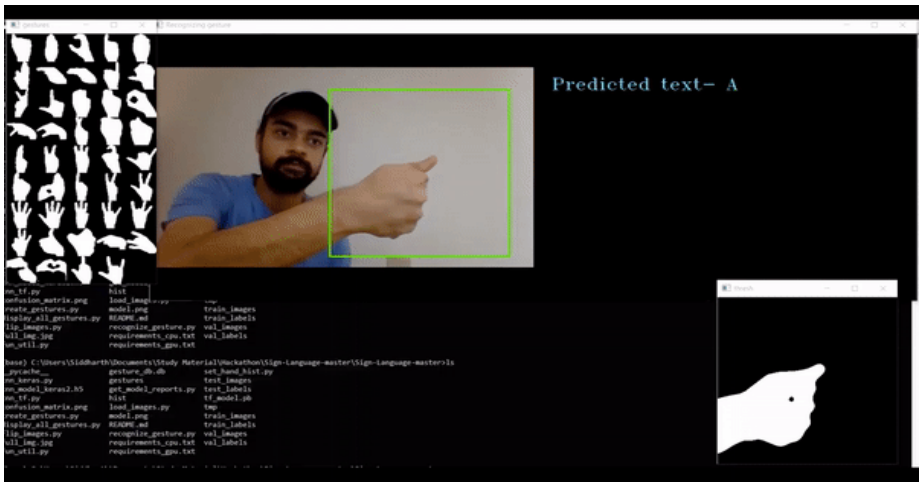
3.7.2.Offline Trainer:

This project is an offline training and testing module for OpenCV Support Vector Machines. Input image datasets can be classified and compiled into output SVM model files for use with the Sign Language app.



3.8. Sign Language Interpreter using Deep Learning:

Using live video feed from the camera, use technology to improve accessibility by finding a creative solution to benefit the lives of those with a disability.our model was able to predict the 44 characters in the ASL with a prediction accuracy >95%.



Technologies and Tools:

Python

TensorFlow

Keras

OpenCV

3.9. Sign Language Identification:

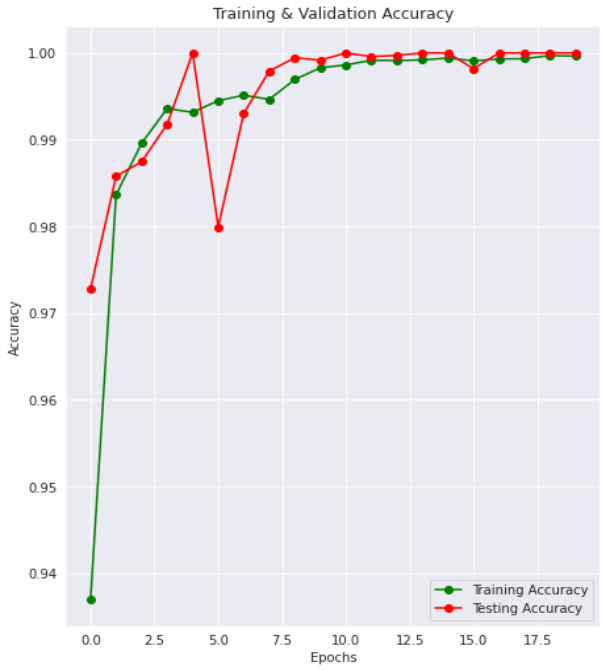
Data information > are given 2 comma separated file(.csv) each on them contains some rows and 785 columns:

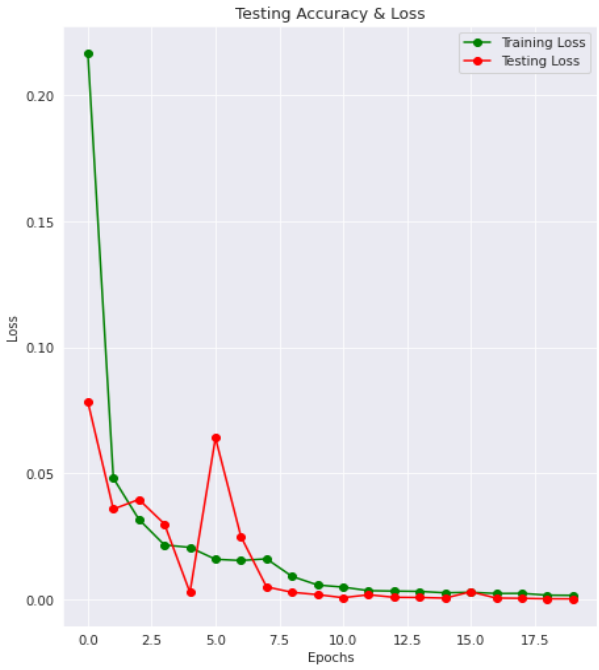
1.from 2nd column onwards each column represents the pixel values associated , representing a 28x28 grayscale image.

2.first column in each row represents a label with the image.

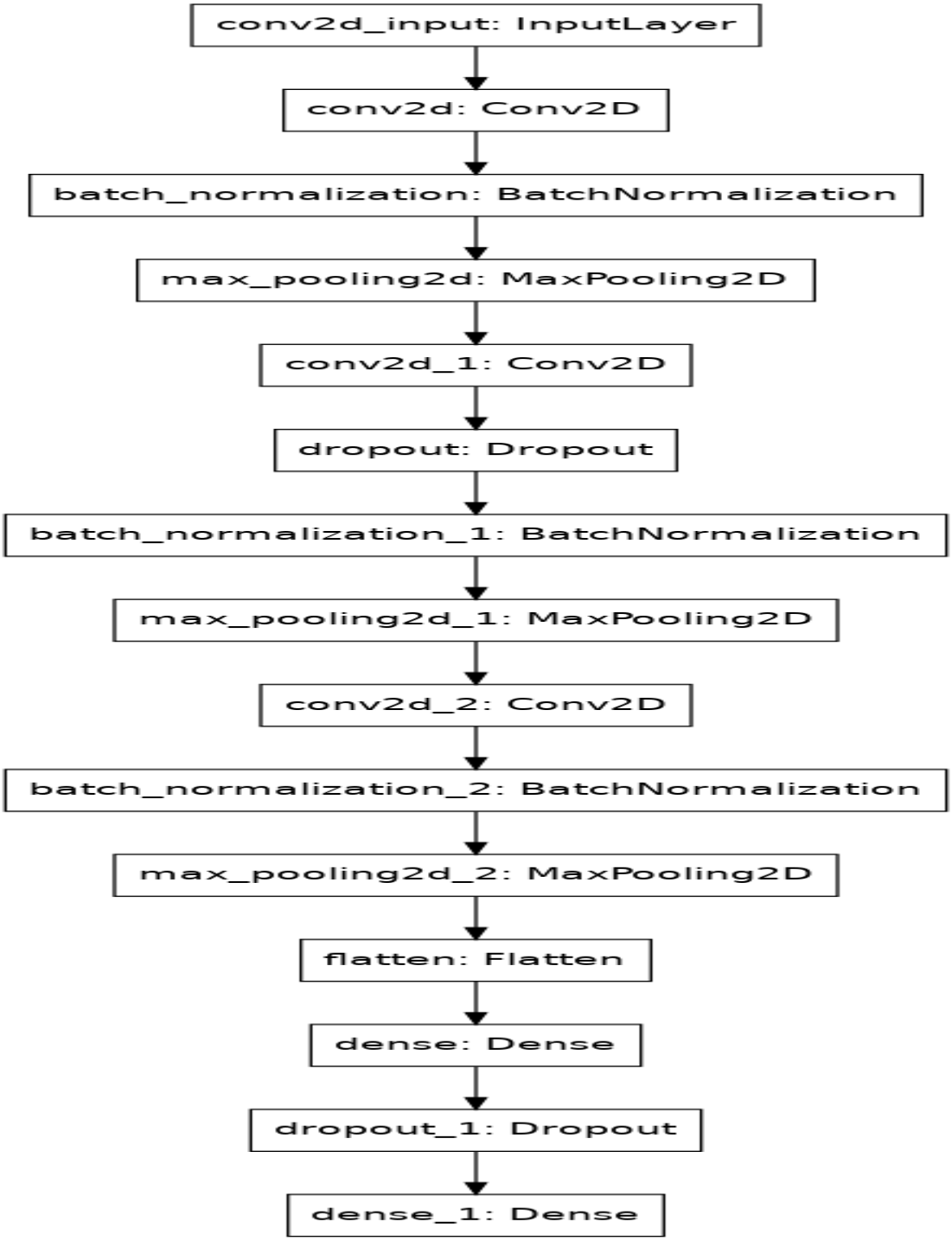
3.There are a total 24 labels (in american sign language) A-I,K-Y means A-Z except J and Z.

3.9.1. Accuracy results





3.9.2. plot the model architecture



Technology has always been the best and fastest means of communication in recent years. There was already a significant amount of work done in the text area to sign the conversion. Several modern apps and devices are available that achieve this conversion. However, almost all of these technologies make extensive use of special hardware components which leads to a very high increase in the total price of the device and therefore it is difficult for many categories to use it due to its high cost

Chapter 4

Design & Implementation

4.1. Visual studio code :

4.1.1. Main.css file :

Here, using css, we format pages, format and design websites (colors - fonts - buttons ....)

```css
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}

.upload-label:hover{
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

4.1.2. Index.html file :

The div tag is a block-level HTML element. It is used to split or divide other HTML tags into meaningful groups.
Select one or multiple files to upload, this file is displayed and name the prediction button, display the result
h2, h3 HTML titles are shown.

index.html ●

C: > Users > kh > Downloads > index.html > div

```
{% extends "base.html" %} {% block content %}

<h2>Sing Langauge Interpreter</h2>

<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <label for="images" class="upload-label">
            Choose Image...
        </label>
        <input type="file" multiple name="images" id="images" accept=".png, .jpg, .jpeg">
    </form>
    <div>
        <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
    </div>
    <div class="loader" style="display:none;"></div>

    <h3 id="result">
        <span> </span>
    </h3>


    <h3 id="result2">
        <span> </span>
    </h3>
    <div class="image-section" style="display:none;" id="imagesParent">




    </div>

</div>

{% endblock %}
```

4.1.3. Base.html file :

The head of an HTML document is the part that is not displayed in the web browser when the page is loaded. It contains information such as the page <title>, links to CSS , links to js , and other metadata (data about the HTML, such as important keywords that describe the document.)

the contents of the <body> element (which are displayed on the page when loaded in a browser). Instead, the head's job is to contain metadata about the document.

A <footer> typically contains links to related documents.

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Sing Langauge Interpreter Demo</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
</head>

<body>
    <nav class="navbar navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="#">Sing Langauge Interpreter Demo</a>

        </div>
    </nav>
    <div class="container">
        <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div>
    </div>
</body>

<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
</footer>

</html>
```

4.1.4. Main.js file :

JavaScript is a enables you to create dynamically updating content, control multimedia, previewed image in function readURL(input) that review the image that selected and predict image that selected by click function on button named ‘predict’

```
$(document).ready(function () {
    // Init
    $('#btn-predict').hide();
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            // console.log(input.files.length);
            for (index = input.files.length-1;index>= 0; index --)
            {
            var reader = new FileReader();
            reader.onload = function (e) {

                    document.getElementById('imagesParent').innerHTML += `
                    <div class="img-preview" id="imagePreviewParent" style="display: inline-block;">
                    <div class="imagePreview" style="background-image: url(${e.target.result})"/>
        </div>`;

                    $('#imagePreview').hide();
                    $('#imagePreview').fadeIn(650);
                }
                 reader.readAsDataURL(input.files[index]);
            }
        }
    }
    $("#images").change(function () {
        document.getElementById('imagesParent').innerHTML ='';
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });
    // Predict
   $('#btn-predict').click(function () {
            let form_data = new FormData();
        // Read selected files
        let totalfiles = document.getElementById('images').files.length;
        for (let index = 0; index < totalfiles; index++) {
           form_data.append("images[]", document.getElementById('images').files[index]);
        }
        // Show loading animation
        $(this).hide();
        $('.loader').show();
            // Make prediction by calling api /predict
        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function (data) {
                // Get and display the result
               $('.loader').hide();
                $('#result').fadeIn(600);
                $('#result').text(' Result:  ' + data);
                // console.log('Success!');
            },
        });
    });
});
```

4.1.5. App.py file :

Definition of libraries and classes

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import re
import cv2
import numpy as np
import time
```

A function is called from keras to decode the predictions

```
# Keras
#from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras.preprocessing import image
```

Flask is a web framework written in Python and built on Werkzeug tools

```
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
```

To define a flask app

```
app = Flask(__name__)
```

Summon the model and determine the labels

```
# Model saved with Keras model.save()
MODEL_PATH = 'bestmodel.hdfs'
dict_labels = {'ل',6:'احبك جدا',5:'ت',4:'ألم',3:'ر',2:'ة',1:'ج',0:'م':
7,'اتمني لك حياه سعيدة',16:'ظ',15:'ز',14:'خ',13:'د',12:'ص',11:'ي',10:'أ',9:'بخير',8:'ض':
17,'س',27:'ن',26:'ذ',25:'ه',24:'ال',23:'ق',22:'ب',21:'ش',20:'ئ',19:'و',18:'ط':
28,'38:'لا',37:'ح',36:'ث',35:'غ',34:'ف',33:'انت',32:'ع',31:'ك',30:'اراقبك',29:'توقف'}
```

Load the model

```
model = load_model(MODEL_PATH)
```

here we ask to get a photo to predict it
then resize it with 64X64 then predict the image

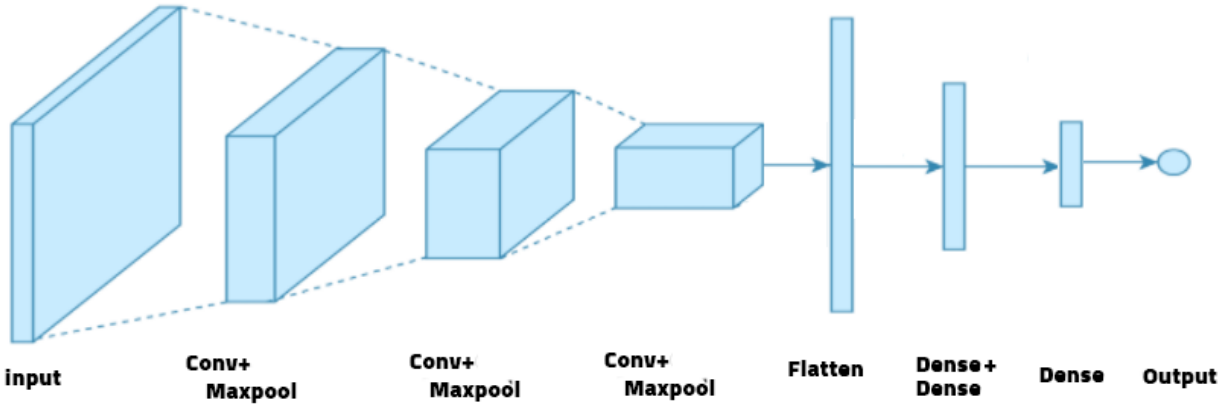
```python
def model_predict(img_path, model):
    img = cv2.imread(img_path)
    x = cv2.resize(img, (64, 64))
    x = np.array(x)
    x = x.astype('float32')/255.0
    x = np.expand_dims(x, axis=0)

    # Be careful how your trained model deals with the input
    # otherwise, it won't make correct prediction!
    preds = model.predict(x)
    pred = preds.argmax()
    return pred
```

Here first we get the file from post request
Then save the file to ./uploads
Then make prediction
Then process your result for human

app.py > ...

```python

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        #f = request.files['images']
        files = request.files.getlist("images[]")
        uploadedFile = []
        for file in files:
            # Save the file to ./uploads
            basepath = os.path.dirname(__file__)
            file_path = os.path.join(
                basepath, 'uploads', secure_filename(file.filename))
            file.save(file_path)
            uploadedFile.append(file_path)

        preds = []
        # Make prediction
        for loaded in uploadedFile:
            preds.append(model_predict(loaded, model))
        pred_class = ''
        # Process your result for human
        for pred in preds:
            for ins in dict_labels:

                if pred == dict_labels[ins]:
                    pred_class += ins + ' '
                    break
        result = " " + pred_class            # Convert to string
        return result

    return None

```

4.2. Creating a model



Input is an image of pixel size — 64 x 64 .Let's understand the architecture of the model. The model contained 10 layers excluding the input layer, let's go layer by layer:

```
model = Sequential([Conv2D(filters=32,  kernel_size=(3,3), activation="relu", input_shape=(64,6
4,3)),
                    MaxPool2D(2,2, padding='same'),

                    Conv2D(filters=128,  kernel_size=(3,3), activation="relu"),
                    MaxPool2D(2,2, padding='same'),

                    Conv2D(filters=512, kernel_size=(3,3), activation="relu"),
                    MaxPool2D(2,2, padding='same'),

                    Flatten(),

                    Dense(units=1024, activation="relu"),
                    Dense(units=256, activation="relu"),
                    Dropout(0.5),
                    Dense(units=39, activation="softmax")
])
model.summary()
```

4.2.1. model output :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 128)       36992
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 512)       590336
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              25691136
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 39)                10023
=================================================================
Total params: 26,591,783
Trainable params: 26,591,783
Non-trainable params: 0
_________________________________________________________________
```

1. Layer 1: A convolutional layer with kernel size of 3x3, it has 32 filters, stride of 1×1,activation type is relu . So, the input image of size 1x64x64x3 gives an output of 1x62x62x3. Total params in layer = 32 *3 * 3*3+32= 896 (bias terms) .

2. Layer 2 : This function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network,
The input values in the receptive were summed up and then were multiplied to a trainable parameter (1 per filter), the result was finally added to a trainable bias (1 per filter). Finally, relu activation was applied to the output. So, the input from the previous layer of size 1x62x62x3 gets sub-sampled to 1x31x31x3.

3: Similar to Layer 1, this layer is a convolutional layer with the same configuration except it has 128 filters. So, the input from the previous layer of size 1x31x31x3 gives an output of 1x29x29x128. Total params in layer = 128 * 3* 3* 32 + 128 =36992.

4. Layer 4: Again, similar to Layer 2, this layer is a pooling layer with the same configuration. Remember, the outputs are passed through the relu activation function. The input of size 1x29x29x128 from the previous layer gets sub-sampled to 1x15x15x128.

5: This time around we have a convolutional layer with 3×3 kernel size and 512 filters. There is no need to even consider strides as the input size is 1x15x15x128 so we will get an output of 1x13x13x512. Total params in layer = 512 * 3 * 3 *128+512 = 590336.

6.Layer 6: Again, similar to Layer 4, this layer is a pooling layer with the same configuration. Remember, the outputs are passed through the relu activation function. The input of size 1x13x13x512 from the previous layer gets sub-sampled to 1x7x7x512.

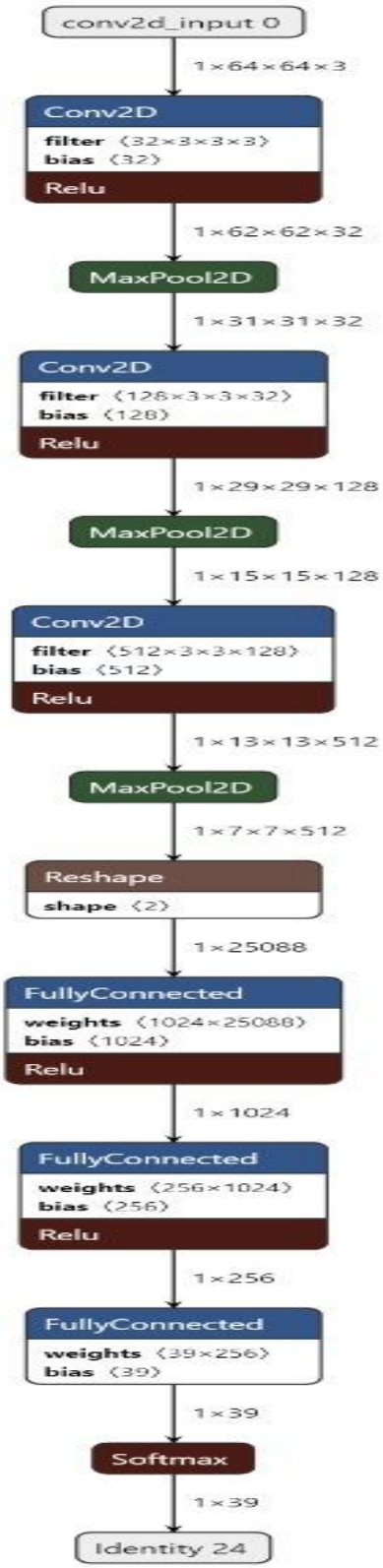
7. Layer 7:This is a fully connected layer , flatten is converting the data into a 1 dimensional array for inputting it to the next layer.The input of size 1x7x7x512 from the previous layer gets sub-sampled to 1x25088.

8. Layer 8: This is a dense layer with 25088 parameters. a neuron in the dense layer receives input from all neurons of its previous layer. So, the input of 1024 units is converted to 25088 units. Total params = 25088 * 1024 +25088 = 25715200.

9. Layer 9: Again, similar to Layer 8, This is a dense layer with 256 parameters. So, the input of 1024 units is converted to 256 units. Total params = 256 * 1024 +256 = 262400.

10. Output Layer: Finally, Again, similar to Layer 9, a dense layer with 39 units is used. And activation type is softmax Total params = 39 * 256 + 39 =10023.

4.2.2. model architecture:



4.3. Adam optimization

4.3.1. What is the Adam optimization algorithm?

1- Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data.
2- The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.
3- Adam is a popular algorithm in the field of deep learning because it achieves good results fast. It can efficiently solve practical deep learning problems.

The algorithm is called Adam. It is not an acronym and is not written as “ADAM”.that name Adam is derived from (adaptive moment estimation).

the benefits of using Adam on non-convex optimization problems, as follows:

Straightforward to implement.
Computationally efficient.
Little memory requirements.
Invariant to diagonal rescale of the gradients.
Well suited for problems that are large in terms of data and/or parameters.
Appropriate for non-stationary objectives.
Appropriate for problems with very noisy/or sparse gradients.
Hyper-parameters have intuitive interpretation and typically require little tuning.

In [27]:
```
callbacks_list =[reduceOnPlateau,checkpoint]
adam=keras.optimizers.Adam(lr=0.001)
```

In [28]:
```
model.compile(loss='categorical_crossentropy',
                  optimizer=adam,
                  metrics=['accuracy'])
```

4.3.2. How Does Adam Work?

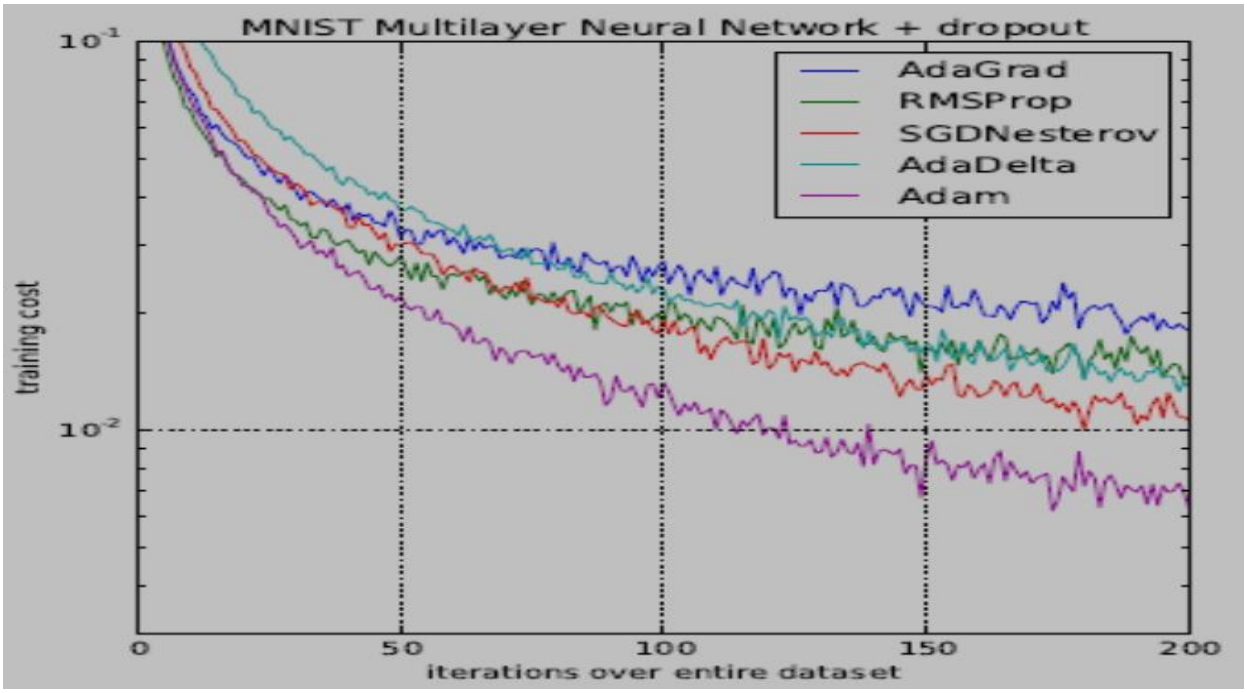
Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training.

A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).
Adam realizes the benefits of both AdaGrad and RMSProp.

4.3.3.When to choose which algorithm?



As you can observe, the training cost in the case of Adam is the least.

1.It is observed that the SGD algorithm (red) is stuck at a saddle point. So the SGD algorithm can only be used for shallow networks.
2.All the other algorithms except SGD finally converge one after the other, AdaDelta being the fastest followed by momentum algorithms.
3.AdaGrad and AdaDelta algorithm can be used for sparse data.
4.Momentum and NAG work well for most cases but are slower.
5. Adam from the plot above it is observed that it is the fastest algorithm to converge to minima.
6.Adam is considered the best algorithm amongst all the algorithms discussed above.

4.4. Results:

Fitting model :

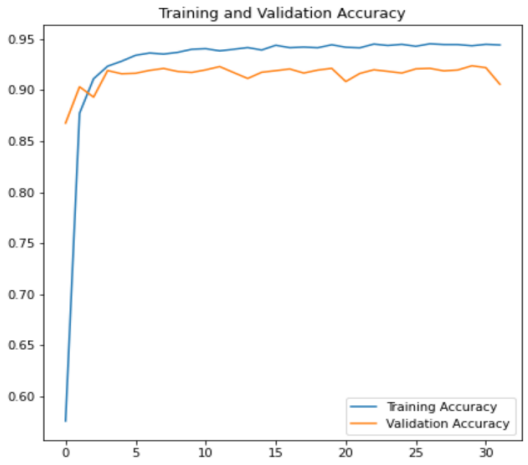
```
In [29]: history=model.fit(X_train,Y_train, batch_size=32, steps_per_epoch=len(X_train) //32,epochs=32,v
         erbose=1,callbacks=callbacks_list,validation_data=(X_val, Y_val))
```

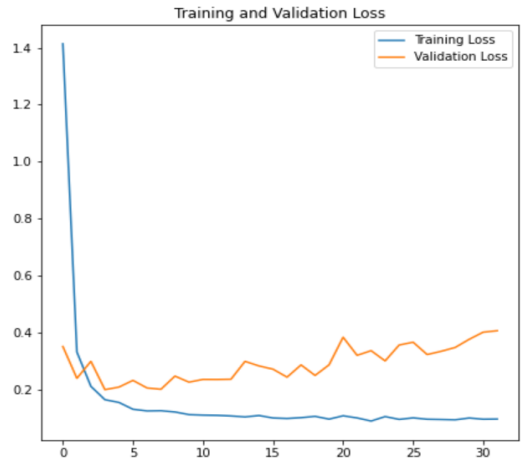
```
Epoch 1/32
1490/1490 [==============================] - 76s 48ms/step - loss: 2.5887 - accuracy: 0.2776
- val_loss: 0.4167 - val_accuracy: 0.8521
Epoch 2/32
1490/1490 [==============================] - 71s 47ms/step - loss: 0.4538 - accuracy: 0.8464
- val_loss: 0.2510 - val_accuracy: 0.8985
Epoch 3/32
1490/1490 [==============================] - 71s 47ms/step - loss: 0.2302 - accuracy: 0.9049
- val_loss: 0.2336 - val_accuracy: 0.9099
Epoch 4/32
1490/1490 [==============================] - 71s 48ms/step - loss: 0.1696 - accuracy: 0.9188
- val_loss: 0.1862 - val_accuracy: 0.9181
```

This technique gives accuracy 92% on validation data, And we predict on (X_test,Y_test).

```
In :  model.evaluate(X_val, Y_val)
```

```
320/320 [==============================] - 11s 35ms/step - loss: 0.3197 - accuracy: 0.9205

Out:  [0.3196914494037628, 0.9204500913619995]
```





Predictions in folder has 34 photo(each photo has one sign from classes in sign language), result is right you can have a look this picture:

Predict on all X_test and Y_test:

[39]:
```
Y_pred = model.predict(X_test)
```

In [41]:
```
Y_pred.round()
```

Out[41]:
```
array([[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.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

In [42]:
```
accuracy_score(Y_test, Y_pred.round())*100
```

Out[42]:
```
91.37964774951077
```

In [44]:
```
print(classification_report(Y_pred.round(), Y_test))
```

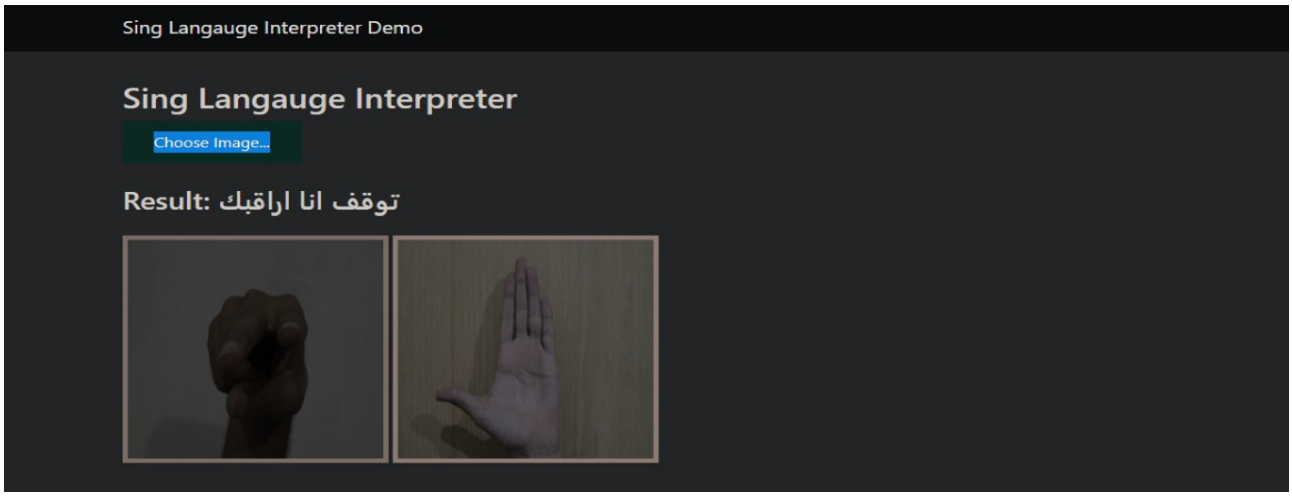
	precision	recall	f1-score	support
0	0.98	0.99	0.99	329
1	0.96	0.98	0.97	290
2	0.97	0.96	0.97	332
3	0.22	0.72	0.34	98
4	1.00	0.92	0.96	13
5	0.98	0.96	0.97	354
6	0.82	1.00	0.90	9
7	0.24	0.74	0.36	109
8	0.97	0.99	0.98	302
9	0.62	0.89	0.73	9
10	0.96	0.97	0.97	313
11	1.00	0.98	0.99	239
12	0.91	0.48	0.63	565
13	0.96	0.98	0.97	313
14	0.95	0.98	0.96	274
15	0.92	0.97	0.94	236
16	0.96	0.99	0.98	295
17	0.84	0.84	0.84	19
18	0.97	0.98	0.97	349
19	0.96	0.98	0.97	278
20	0.99	0.98	0.98	317
21	0.98	0.99	0.99	273
22	0.96	0.96	0.96	389
23	0.93	0.98	0.95	382
24	0.98	0.98	0.98	264
25	0.98	0.94	0.96	340
26	0.96	0.99	0.97	284
27	0.98	0.96	0.97	340
28	0.99	0.97	0.98	325
29	0.83	0.91	0.87	22
30	0.90	0.90	0.90	10
31	0.96	1.00	0.98	356
32	0.97	0.96	0.97	417
33	0.64	0.88	0.74	8
34	0.96	0.95	0.95	349
35	0.96	0.97	0.97	382
36	0.94	0.96	0.95	382
37	0.97	0.95	0.96	310
38	0.90	0.54	0.68	543
micro avg	0.91	0.92	0.92	10179
macro avg	0.90	0.93	0.90	10179
weighted avg	0.94	0.92	0.92	10179
samples avg	0.91	0.91	0.91	10179

4.5. User interface :

Step.1 -> Chose image

Step.2 ->image viewed then Click predict

Step.3 -> Result viewed



Chapter 5

Conclusion and Future work

5.1. Conclusion :

1- We create our dataset from A-z that includes nearly 65,000 images ,this dataset includes 39 classes 32 for letters and 7 for words, each class contains a specific letter or word.

2- A model convolutional neural network was made to organize and train the images that have 10 layers and we use adam algorithm.

5.2. Future work :

1- Create an android app for our project.
2- Detect the sign and convert it to text and sound.
3- Choose any language you want to convert it to sign language and vice versa.
4- Create a video calling app and translate between deaf people and normal people.

References:

1- https://en.wikipedia.org/wiki/Sign_language

2- https://en.wikipedia.org/wiki/Convolutional_neural_network

3- https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

4- https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/

5-https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

6-https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

7-https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

8-https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

9-https://www.kaggle.com/

10-https://stackoverflow.com/questions/56390917/convert-a-list-of-images-to-grayscale-using-opencv

11-https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

12-https://keras.io/api/models/model_training_apis/

13-https://ruder.io/optimizing-gradient-descent/index.html#adam

14-https://github.com/noumannahmad/Flask/tree/master/Image-Classification-Webapp