

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ  
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ  
ΜΗΧΑΝΙΚΩΝ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ  
ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ



<http://www.eee.uniwa.gr>  
<http://www.idpe.uniwa.gr>  
Θηβών 250, Αθήνα-Αιγάλεω 12241  
Τηλ: +30 210 538-1614

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών  
Τεχνητή Νοημοσύνη και Βαθιά Μάθηση  
<https://aidl.uniwa.gr/>

UNIVERSITY OF WEST ATTICA  
FACULTY OF ENGINEERING  
DEPARTMENT OF ELECTRICAL & ELECTRONICS  
ENGINEERING  
DEPARTMENT OF INDUSTRIAL DESIGN AND  
PRODUCTION ENGINEERING  
<http://www.eee.uniwa.gr>  
<http://www.idpe.uniwa.gr>

250, Thivon Str., Athens, GR-12241, Greece  
Tel: +30 210 538-1614

Master of Science in  
Artificial Intelligence and Deep Learning  
<https://aidl.uniwa.gr/>

## Master of Science Thesis

### Sign Language Detection

Greek Sign Language - The Alphabet



Student: Stamoulis Stefanos-Ioannis  
Registration Number: AIDL-0014

MSc Thesis Supervisor Lecturer  
Nikolaou Grigorios

ATHENS-EGALEO, September 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ  
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ  
ΜΗΧΑΝΙΚΩΝ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ  
ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ



<http://www.eee.uniwa.gr>

<http://www.idpe.uniwa.gr>

Θηβών 250, Αθήνα-Αιγάλεω 12241

Τηλ: +30 210 538-1614

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών

Τεχνητή Νοημοσύνη και Βαθιά Μάθηση

<https://aidl.uniwa.gr/>

UNIVERSITY OF WEST ATTICA  
FACULTY OF ENGINEERING  
DEPARTMENT OF ELECTRICAL & ELECTRONICS  
ENGINEERING  
DEPARTMENT OF INDUSTRIAL DESIGN AND  
PRODUCTION ENGINEERING

<http://www.eee.uniwa.gr>

<http://www.idpe.uniwa.gr>

250, Thivon Str., Athens, GR-12241, Greece

Tel: +30 210 538-1614

Master of Science in

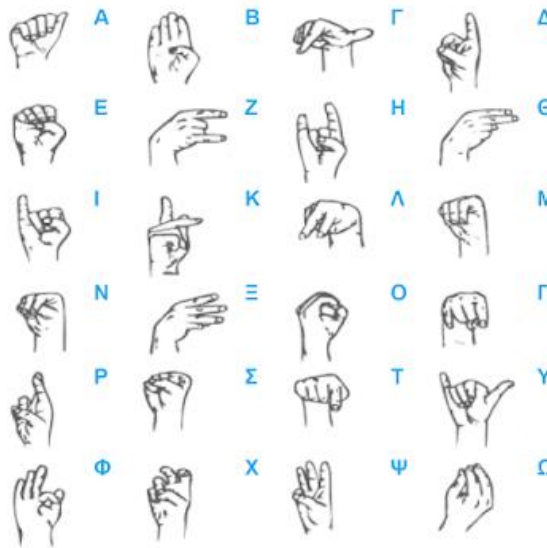
Artificial Intelligence and Deep Learning

<https://aidl.uniwa.gr/>

## Μεταπτυχιακή Διπλωματική Εργασία

### Αναγνώριση Νοηματικής Γλώσσας

Greek Sign Language - The Alphabet



Φοιτητής: Σταμούλης Στέφανος-Ιωάννης  
AM: AIDL-0014

Επιβλέπων Λέκτορας  
Νικολάου Γρηγόριος

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Σεπτέμβρης 2022

This MSc Thesis has been accepted, evaluated and graded by the following committee:

Supervisor	Member	Member
Nikolaou Grigorios	Patrikakis Charalampos	Feidakis Michael
Lecturer	Professor	Lecturer

**Copyright** © Με επιφύλαξη παντός δικαιώματος. All rights reserved.

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ και Σταμούλης Στέφανος-Ιωάννης,  
Σεπτέμβριος, 2022**

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον/την συγγραφέα του και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις θέσεις του επιβλέποντος, της επιτροπής εξέτασης ή τις επίσημες θέσεις του Τμήματος και του Ιδρύματος.

#### **ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ**

Ο/η κάτωθι υπογεγραμμένος/η Σταμούλης Στέφανος-Ιωάννης του Αθανασίου, με αριθμό μητρώου 0014 μεταπτυχιακός/ή φοιτητής/ήτρια του ΔΠΜΣ «Τεχνητή Νοημοσύνη και Βαθιά Μάθηση» του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών και του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής,

**δηλώνω υπεύθυνα ότι:**

«Είμαι συγγραφέας αυτής της μεταπτυχιακής διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων,

*MSc in Artificial Intelligence & Deep Learning, MSc Thesis*

*Stamoulis Stefanos-Ioannis AIDL-0014.*

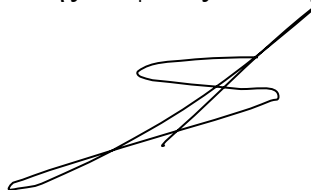
είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Η εργασία δεν έχει κατατεθεί στο πλαίσιο των απαιτήσεων για τη λήψη άλλου τίτλου σπουδών ή επαγγελματικής πιστοποίησης πλην του παρόντος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι ..... και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος/ουσας καθηγητή/ήτριας.»

Ο Δηλών

Σταμούλης Στέφανος-Ιωάννης



**Copyright ©** All rights reserved.

**University of West Attica and Stamoulis Stefanos-Ioannis**

**September 2022**

You may not copy, reproduce or distribute this work (or any part of it) for commercial purposes. Copying/reprinting, storage and distribution for any non-profit educational or research purposes are allowed under the conditions of referring to the original source and of reproducing the present copyright note. Any inquiries relevant to the use of this thesis for profit/commercial purposes must be addressed to the author.

The opinions and the conclusions included in this document express solely the author and do not express the opinion of the MSc thesis supervisor or the examination committee or the formal position of the Department(s) or the University of West Attica.

#### **Declaration of the author of this MSc thesis**

I, Stamoulis Stefanos-Ioannis with the following student registration number: 0014, postgraduate student of the MSc programme in “Artificial Intelligence and Deep Learning”, which is organized by the Department of Electrical and Electronic Engineering and the Department of Industrial Design and Production Engineering of the Faculty of Engineering of the University of West Attica, hereby declare that:

I am the author of this MSc thesis and any help I may have received is clearly mentioned in the thesis. Additionally, all the sources I have used (e.g., to extract data, ideas, words or phrases) are cited with full reference to the corresponding authors, the publishing house or the journal;

this also applies to the Internet sources that I have used. I also confirm that I have personally written this thesis and the intellectual property rights belong to myself and to the University of West Attica. This work has not been submitted for any other degree or professional qualification except as specified in it.

Any violations of my academic responsibilities, as stated above, constitutes substantial reason for the cancellation of the conferred MSc degree.

I wish to deny access to the full text of my MSc thesis until ....., following my application to the Library of UNIWA and the approval from my supervisor.

The author  
Stamoulis Stefanos-Ioannis

(Signature)

## **Abstract**

Communication is defined as the exchange of information, ideas and feelings. In order for two people to communicate with each other they must understand the same language. When a person is deaf they cannot hear and when they are mute they cannot speak. So in order to communicate with each other these groups of people use sign language. Being the community of the population this makes it difficult for them to communicate with the rest of the world. To overcome this barrier with the technologies that exist today we can create machine learning based models so that a deaf and dumb person with the help of this model can communicate with a normal person without knowing sign language. In this project we propose to create a dataset for Greek sign language phrase and word recognition so that real-time recognition can be performed. Real-time sign language recognition could solve many of the communication problems between deaf and non-deaf people. To achieve the desired result we train a model to create a real-time recognition system. The model reaches a high level of accuracy even though the data is not very accurate.

## **Keywords**

Greek Sign Language (GSL), Sign Language Recognition (SLR), Long-short-term memory(LSTM), Computer Vision, Machine Learning, Deep Learning.

## Περίληψη

Η επικοινωνία ορίζεται η ανταλλαγή πληροφοριών, ιδεών και συναισθημάτων. Για να μπορέσουν δυο άνθρωποι να επικοινωνήσουν μεταξύ τους θα πρέπει να κατανοούν την ίδια γλώσσα. Στην περίπτωση όμως των κωφάλαλων ο διάυλος επικοινωνίας είναι διαφορετικός. Όταν ένας άνθρωπος είναι κωφός αδυνατεί να ακούσει και όταν είναι μουγγός αδυνατεί να μιλήσει. Έτσι για να μπορέσουν να επικοινωνήσουν μεταξύ τους αυτές οι ομάδες ανθρώπων χρησιμοποιούν τη νοηματική γλώσσα. Όντας όμως η κοινότητα του πληθυσμού αυτό το καθιστά δύσκολο να επικοινωνήσουν με τον υπόλοιπο κόσμο καθώς ο υπόλοιπος κόσμος δεν κατανοεί την νοηματική γλώσσα οπότε αδυνατεί να υπάρξει διάυλος επικοινωνίας μεταξύ ενός κωφάλαλου και ενός φυσιολογικού ανθρώπου. Για να ξεπεραστεί αυτό το εμπόδιο με τις τεχνολογίες που υπάρχουν σήμερα μπορούμε να δημιουργήσουμε μοντέλα βασισμένα στη μηχανική μάθηση ώστε ένας κωφάλαλος με τη βοήθεια αυτού του μοντέλου να μπορεί να επικοινωνήσει με έναν φυσιολογικό άνθρωπο χωρίς να γνωρίζει νοηματική γλώσσα. Στην παρούσα εργασία προτείνεται να δημιουργηθεί ένα σύνολο δεδομένων για αναγνώριση φράσεων και λέξεων της Ελληνικής νοηματικής γλώσσας ώστε να μπορεί να γίνει αναγνώριση σε πραγματικό χρόνο. Η αναγνώριση της νοηματικής γλώσσας σε πραγματικό χρόνο θα μπορούσε να επιλύσει πολλά από τα προβλήματα επικοινωνίας μεταξύ κωφάλαλων ανθρώπων και μη. Για να επιτύχουμε το επιθυμητό αποτέλεσμα εκπαιδεύουμε ένα μοντέλο Tensorflow για να δημιουργήσουμε ένα σύστημα αναγνώρισης σε πραγματικό χρόνο. Το σύστημα επιτυγχάνει ένα καλό επίπεδο ακρίβειας παρόλο που τα δεδομένα που συλλέγουμε δεν είναι πολλά.

## Λέξεις – κλειδιά

Ελληνική Νοηματική Γλώσσα(GSL), Αναγνώριση νοηματικής γλώσσας(SLR), Μηχανική Μάθηση, Long-short-term memory(LSTM), Βαθιά Μάθηση

## Table of Contents

<b>INTRODUCTION</b>	<b>8</b>
<b>The subject of this thesis</b>	<b>8</b>
<b>Aim and objectives</b>	<b>11</b>
<b>Methodology</b>	<b>12</b>
<b>Structure</b>	<b>12</b>
<b>1 RELATED WORK</b>	<b>12</b>
<b>1.1 Methods of Collecting Data</b>	<b>13</b>
<b>1.2 Processing Methods have been used</b>	<b>14</b>
<b>1.3 Types of Sign Language Detection</b>	<b>15</b>
<b>2 METHODOLOGY</b>	<b>16</b>
<b>2.1 Introduction to OpenCV library</b>	<b>17</b>
<b>2.1.1 The advantage of OpenCV</b>	<b>18</b>
<b>2.1.2 Image Processing and Computer Vision</b>	<b>18</b>
<b>2.1.3 Modules of OpenCV library</b>	<b>18</b>
<b>2.2 MediaPipe / Holistic Landmarks</b>	<b>19</b>
<b>2.2.1 How the model will work</b>	<b>19</b>
<b>2.2.2 KeyPoints Extraction</b>	<b>19</b>
<b>2.2.3 POSE_LANDMARKS</b>	<b>20</b>
<b>2.2.4 FACE_LANDMARKS</b>	<b>20</b>
<b>2.2.5 LEFT_HAND_LANDMARKS &amp; RIGHT_HAND_LANDMARKS</b>	<b>21</b>
<b>3 SEQUENTIAL AND LONG-SHORT TERM MEMORY</b>	<b>22</b>
<b>3.1 Sequential and its use</b>	<b>22</b>
<b>3.1.1 Recurrent Neural Network</b>	<b>22</b>
<b>3.1.2 LSTM(long short-term memory)</b>	<b>24</b>
<b>3.1.3 What is Backpropagation</b>	<b>26</b>
<b>3.1.4 Mathematical representation of the model</b>	<b>28</b>
<b>4 DATA ACQUISITION</b>	<b>29</b>
<b>4.1 Dependencies for Data Collection</b>	<b>30</b>
<b>4.2 Landmarks acquisition</b>	<b>31</b>
<b>4.3 How data was obtained</b>	<b>31</b>
<b>5 BUILD THE MODEL</b>	<b>32</b>
<b>5.1.1 LSTM</b>	<b>32</b>
<b>5.1.2 LSTM Usage</b>	<b>34</b>
<b>5.1.3 LSTM Layer units usage</b>	<b>34</b>
<b>5.1.4 Dense Layers</b>	<b>35</b>
<b>5.2 Models created</b>	<b>37</b>
<b>5.3 Fit The Model</b>	<b>37</b>
<b>5.4 Model Performance</b>	<b>38</b>
<b>6 EVALUATION AND TEST IN REAL TIME</b>	<b>38</b>
<b>7 CONCLUSION</b>	<b>41</b>



**Bibliography – References – Online sources**

## INTRODUCTION

### The subject of this thesis

Sign languages use a different way of communication from what we are used to in our everyday life. Sign languages use the physical - visual pathway produced by the body and perceived by the eyes of the receiver. This of course does not apply to spoken languages. Spoken languages use the phonetic-acoustic route, meaning that the speaker articulates his words orally and they are perceived by the ear of the receiver.

Unfortunately sign languages are not international so all the deaf people in the world cannot communicate with each other having in common the knowledge of sign language. Sign languages develop when groups of Deaf people get together and communicate with each other. Each country develops its sign language by different methods which we will discuss below. [1].

In fact the signs used by the Deaf have a similar structure to the way spoken words are spoken. If we take as an example the English language and how many of its words are produced in the verbal speech we will find many correspondences in the creation of words in sign language. As spoken words are produced by a small number of different sounds, so are signs produced by several gestures. Thus, signs are not holistic gestures but analyzable gestures. They are not just a set of gestures, but a set of gestures that can be analyzed.

Sign languages include some features that are of course also valid for spoken languages. These are:

- The manual features, i.e. the speaker in both cases (either with signs or with speech) depending on his shape, his position, how he moves his hands and the orientation of the palm of his hand and the fingers
- Non-manual features, such as gaze, head position, head nodding, shoulder orientation and various types of facial features (facial expression, grimaces, eyebrow movement, mouth gestures, etc.)

The combinations mentioned above essentially represent a glossary, which is the basic element of the structure of a sign language. This represents the closest meaning of a signal [2]. Sign languages, like spoken languages, consist of many different grammatical rules governing manual and non-manual features. So when signers want to create sentences in sign language they use the method described above. Depending on the situation, a particular feature can decide the

*MSc in Artificial Intelligence & Deep Learning, MSc Thesis*  
*Stamoulis Stefanos-Ioannis AIDL-0014.*

whole meaning of the conversation. For example, it can completely alter the meaning of the verbs, as well as give spatial/temporal reference. In addition, these features can be used to distinguish one object from another.

Deaf people have difficulties communicating with people who are not sign language users. For example in Greece, the state doesn't encourage people to learn sign language as there are not enough courses for those who want to learn and practice. If people who do not suffer from a disability such as deafness or mute, had the opportunity to learn sign language, this would have the effect of adapting the disabled people of this category to integrate into society much more easily because they would have many more people to communicate with.

As a result, sign language recognition tools play a key role in the communication of people, regardless of whether they are disabled. A sign language recognition tool can recognize the signals carried out by the signer and translate them to the receiver who will be able to understand them without having any prior knowledge of sign language. Technology has evolved in the field of communication by making innovations such as automatic translation in any spoken language. In sign language, similar applications include sign language classification system which turns signs into words and thus into sentences.

## **Aim and objectives**

In automatic sign language recognition, there are some challenges in terms of making a system able to assist people with and without disabilities communicate successfully with each other through an automated communication tool.

- Deaf people often use a grammatical device known as "role reversal" when narrating an event or story with more than one character [3]. So, the position in space where they stand, and the content have a big impact on the interpretation of sign language. For example, there are no personal pronouns such as "he", "she", "it". If the signer wants to refer to someone, he points directly to him. Likewise, if he wants to reproduce the content of a conversation, pronouns are modeled by returning his/her shoulders or gaze. In addition, the deaf person utilizes the space in front of them to locate people or places [3].
- Many glosses are distinguished only by their constituent non-manual features are usually difficult to detect accurately, as even very small human movements can

impose different grammatical or semantic interpretations depending on the context. [3] .

- The speed of execution of a particular tongue marker may indicate a completely different meaning or attitude of the signer; for example, signers would not use two tongues to express "I run fast" but would speed up the execution of the involved meanings [4]
- Notationalists often discard an individual linguistic feature depending on the previously executed and evolving glosses.
- For most sign languages, very few formal typing activities have been implemented, to the extent that signers in the same country exhibit distinct differences during the execution of a particular gloss.

Before the development of deep learning, scientists were mainly working on individual languages and gestures since classification was only performed between different words and static gesture frames. With deep learning, it is now possible to train a system through animated images such as videos whereas in the past the only information that could be given to a system was in the form of an image[5][6]. Sign language recognition is directly related to computer vision and therefore, most approaches dealing with sign language recognition have been adapted in this direction.

## **Methodology**

In this project the user will be able to create his own dataset and store it in any desired storage space. He will be able to record movements that indicate words, phrases, or gestures in video format. Subsequently, all the stored frames will be utilized to train the model. The model will predict the outcomes it believes according to the gestures performed by the user. Some basic libraries such as google's mediapipe/holistic, tensorflow, cv2 for recording and matplotlib for visualization will be used to implement this.

## **Structure**

In this paper we present in order, similar works that have been done, models that have been trained to do sign language recognition. The results obtained and the accuracy score. Below we present how the data acquisition was done for this particular work where we want to achieve real time sign language recognition. Then the methodology with which the model was built, which layers were used, how many epochs, evaluation of the model and real time test. Finally, the reasons for using the technologies used, what could be done better and how such a model could eliminate a problem that exists in society.

## 1 RELATED WORK

In recent years, there has been increasing interest in sign language among researchers and it has become a dominant topic of study. Machine learning plays a key role in constructing tools for sign language recognition and researchers among many countries use it in an attempt to develop such tools. These techniques allow computer systems to make decisions based on data[4][7]. The classification algorithms that are the basic tools in such a system, most of the time need two datasets, training data and test data. Subsequently, by training the model with the training data it can be tested on the test data. When creating a dataset and we get to the point of training the model, we split the data approximately 80/20 and 70/30 depending on the data sets we have and the number of them and experiment to see in which case the model would give better results.

### 1.1 Methods of Collecting Data

Based on the method of data collection, The previous works that have been done in this field are mainly divided into two categories. The first one, is methods that use recording systems and gloves that give data on hand movement which of course can be very accurate not only for finger tracking but also for the rest of the body, sensors etc[4]. These methods are called direct measurement methods. In addition, there are also approaches based purely on vision. Data is collected through photographs or videos and there is the option to extract spatial data for example the location of the subject's hands in relation to a specific reference point, usually the subject's head, and the movements they perform. The recognition of the hands' movement is mainly achieved because the pre-trained models, used for the recognition of such data, understand the skin color, the pose of the human figure and the place where each limb of the subject is located.[7][8]

To obtain data for any of the above-mentioned approaches, different devices must be used, however the main device used is the camera. Generally, there are many devices that can help the data collection as sign language relies on many factors to understand the signer. Some other devices worth mentioning are:

- Microsoft Kinect that provides color video streaming as well as depth video  
(As mentioned above when the perceiver wants to present a situation in

which a third person is located will try to present him in space. So, such a tool as Microsoft Kinect would be useful in recognition of such discussion).

- The sensor of the accelerometer is equally important as in sign language the speed with which the signs are carried out can completely change the meaning of the sentence.
- Tracing gloves give full details of where the signifier's hands are in space and precise details of the movement carried out by the fingers as well as the whole hand.
- Another system used for data acquisition is Leap Motion Controller (LMC)[9][10]. This is a non-contact controller, based in San Francisco and developed by the technology company "Leap Motion" now called "Ultraleap"[9]. Approximately, it can operate about 200 frames per second and detect and track hands, fingers, and objects that appear to resemble fingers. Most researchers collect the training dataset by recording themselves, as finding a dataset in sign language is a problem [12].

## **1.2 Processing Methods have been used**

Many sign language recognition systems have been created over the years. However, the use of different processing methods or application systems have given results of different accuracy. All these different models are listed below, and their accuracy is shown in parentheses.

- In SLR the most widely used model has been The Hidden Markov Model (HMM). There are plenty of variations of HMM that has been used as Multi Stream HMM (MSHMM)(86.7% accuracy) which based on HMMs, Light-HMM(83,6% accuracy), and Tied-Mixture DensityHMM[12].
- ANN[14]
- Naive Bayes Classifier(NBC)
- Multilayer Perceptron(MLP)[14]
- Self-Organizing Map(SOM)[14]
- Self Organizing Feature Map(SOFM)
- Simple Recurrent Network(SRN)[15]

- Support Vector Machine(SVM)(97.5% accuracy)[14]
- 3D Convolution residual Network(CRN) [15]
- Wavelet-based method(100% accuracy)[12]
- EigenValue Euclidean Distance(97% accuracy)[16]

### 1.3 Types of Sign Language Detection

There are two types of sign language recognition and clearly, they are both different with one being more difficult to detect than the other. In the first case which is also called isolated sign language recognition the system is trained to recognize only static data, for example only a gesture. The system can recognize a letter of the alphabet or a digit or a specific gesture which represents a word or a phrase. In the second case, in continuous sign language recognition the system can recognize and synthesize whole sentences [17].

Although much research has been carried out in the past, many deficiencies (weaknesses) need to be addressed in order to meet the challenges; some of these challenges are:

- Isolated SLR methods should be as accurate as possible in labelling words
- In continuous SLR methods, the program is mainly based on the isolated systems with time segmentation as preprocessing, which is non-trivial and this can cause many errors in the successive steps and eventually in the sentence synthesis[17]
- The devices that have to be used to create sign language recognition systems are expensive and it is difficult for the public to create a good model to commercialize sign language recognition systems.[17]
- The camera we use on our computer doesn't give the best quality of video, so the dataset gets collected might have degraded quality.[17]
- Acquiring data from sensors could be very helpful in creating such a technology but problems arise such as noise, poor human handling, poor ground connection etc.[17]
- Vision based methodologies may occur plenty of inaccuracies because hands and fingers are overlapping; this problem could be solved by some 3D visualization of the mentalist.[17]
- There are misconceptions around sign language such as for example:
  - Sign language is common all over the world
  - Sign language is based on spoken language



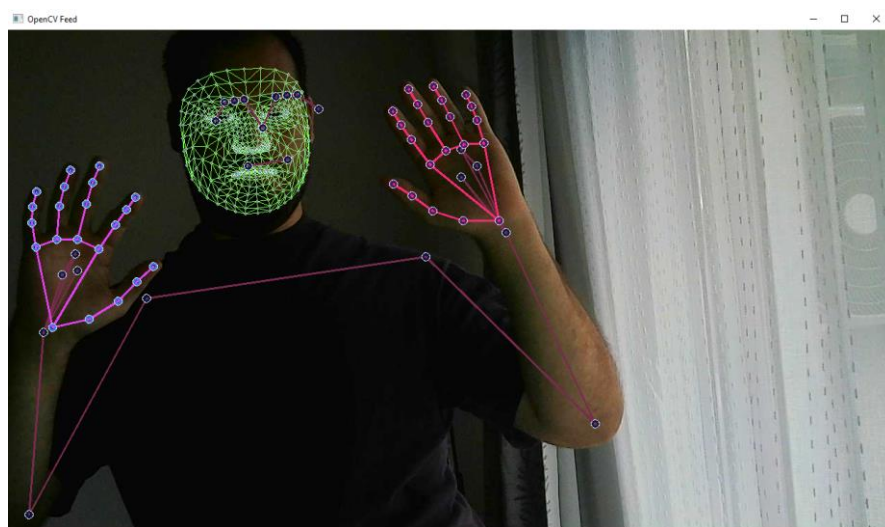
In this project a real-time sign language detection system is developed. Initially the user has the option to create his own dataset but there are also ready data for use and experimentation. The dataset is created using OpenCV and a web camera. The program is built in Python

## 2 METHODOLOGY

In this paper we used the google holistic library "mediapipe holistic" which incorporates separate models for pose, face and hand data each of which is optimized for its own domain. In section 2.2 we will see a brief description of the models used by mediapipe holistic which plays a key role in the creation of the dataset and in real-time sign language recognition. First of all they will be presented some features of the OpenCV library which plays the main role in the creation of the dataset that has been used to create this project as it allows us to take videos in the process of signal prediction and data collection. On this video produced by OpenCV we can see the key points generated by Mediapipe Holistic (Fig.1).

Then after collecting the data we are ready to build the model, using a library called Keras. Keras is an open source library that provides a Python interface for artificial neural networks[31]. It acts as an interface for the Tensorflow library. It is designed to allow quick experimentation with deep neural networks however its main advantage is that it is very user-friendly and scalable[31].

Below in this section we will explore what a Sequential model is, what an LSTM layer is and the meaning of a Recurrent Neural Network(RNN) and how they are used in our model.



**Fig. 1. Keypoints from Mediapipe Holistic**

## **2.1 Introduction to OpenCV library**

OpenCV (Open Source Computer Vision Library) is an Application Peripheral Interface (API) developed by Intel. It is used to process images and perform computer vision tasks on a range of hardware platforms. OpenCV was originally developed in 1999 as part of an Intel project aimed at accelerating CPU-intensive applications including video analysis tools and more.[22]

OpenCV is a collection of computer vision algorithms, interfaces to image processing tools, and classes that implement computer vision algorithms as well as image processing. OpenCV is written in C++, optimized for multicore processors and designed for real-time applications. It contains over 500 functions spanning many areas in vision, including factory product inspection, medical imaging, security, user interface and robotics[22]

The OpenCV project is a library of computer vision algorithms, simple to use and can be operated by an individual or a team of developers to build sophisticated applications quickly. Microsoft created the Direct3D API in order to create multimedia applications and games and OpenCV is using it as well . One of the goals of OpenCV is to provide a free and open source environment where every developer will be able to distribute and make the library as optimized as possible[22].

### **2.1.1 The advantage of OpenCV**

The OpenCV library is optimized for Intel processors. Through this library, if it is open source, the new community will be able to leverage the library and apply it to computer vision tasks on computers and mobile phones. OpenCV is supplied with C sources and has a platform independent interface.

### **2.1.2 Image Processing and Computer Vision**

When we talk about computer vision and image processing, we are usually referring to the same thing. But this is not the case, as when we talk about image processing, we are referring to low-level image and video processing, while when we talk about computer vision we are talking about high-level image and video processing.

OpenCV is therefore designed to provide the basic tools needed to solve computer vision problems[22]. Currently, the basic elements of the library are at a very good level to provide complete solutions to computer vision problems.

### **2.1.3 Modules of OpenCV library**

The OpenCV core module contains basic data structures and basic functions used by other modules. The image processing related functions in the IMGPROC module include linear and nonlinear image filtering and geometric transformations such as scaling, rotation, and translation. The motion estimation and object tracking algorithms in the VIDEO module can be used to track moving objects in videos. The machine-learning interfaces in ML are accompanied by a set of sample applications that make use of these interfaces. The HighGUI module provides multi-platform windowing capabilities and interfaces for accessing feature streams from cameras and for controlling display devices connected to computer systems via USB or serial ports.

## **2.2 MediaPipe / Holistic Landmarks**

Media Pipe is a framework for machine learning solutions developed by Google. It is an open source framework, which allows developers to create custom solutions quickly. Media Pipe allows the tracking of hands and fingers using ML algorithms in real-time on mobile phones as well as on low-end computers[16].Below we will look at the development of the models that have been used to extract the key points. Giving this visual aid will help us create the dataset in the first place as well as guide the signer about the signs he is performing on camera.

### **2.2.1 How the model will work**

In this work, we propose a pipeline that consists of multiple steps: the input video sequence, as a set of RGB frames, goes through a pose and hands tracker for key-point extraction, to the Transformer encoder for predicting the embedding of the sequence. The classification as the last step is necessary to perform few-shot learning.

## **2.2.2 KeyPoints Extraction**

Input videos could be sent directly to the model as a collection of pixels for each one of the image's color channels, such as Convolutional Neural Networks or Image Transformers. Complexity of a large quantity of pixels makes it necessary to have larger models, capable of creating good internal representations of the data in the context of the problem being solved. For this, more data is required in order to prevent common problems such as overfitting (when the model adjusts itself exactly according to its training data and fails to predict new results) or when it underfitting (when the model relies on a simple characteristic of the training data rather than learning to represent distributions). Additionally, given a limited training set and computer resources, the training process can have poor performance and become difficult to converge.

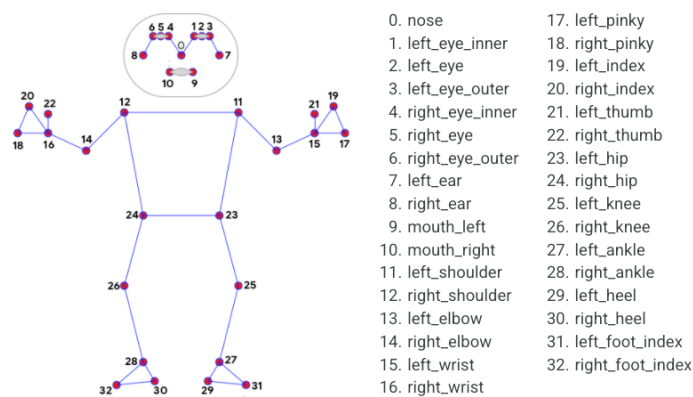
Sign Language Recognition requires only the information about the position of the body, hands and face. Other information contained in the pixels of the images, such as color, shadow and depth should not affect classification results[24]. To extract this useful information, it is a good practice to use a pre-trained model to predict key points from which we can determine articulations, fingers or regions of interest on faces.

## **2.2.3 POSE\_LANDMARKS**

First and foremost we will explore the model used to extract the landmarks from Mediapipe's pose landmark model. In this case the BlazePose GHUM 3D model is used.

In this model a detector-tracker setup is used which shows excellent real-time performance. The Pipeline consists of a body pose detector and a pose detection network[23]. The detector in this case predicts the coordinates of the keypoints, the presence of the face. In order to implement the prediction correctly the model focuses on detecting a rigid part of the body such as the face or the torso. In this way the model can more easily predict the space where the person is located. This face detector predicts additional person specific alignment parameters: the middle point between the person's hips, the size of the circle circumscribing the whole person, and incline (the angle between the lines connecting the two mid-shoulder and mid-hip points)[23].

The BlazePose GHUM 3D model in MediaPipe Pose predicts the location of 33 pose landmarks(Fig.2)



**Fig. 2. all the Keypoints from Pose\_Landmarks**

## 2.2.4 FACE\_LANDMARKS

Regarding the face\_landmarks as mentioned above the head landmarks are important to find the pose performed by the person in the video under consideration. For the 3D facial landmarks, transfer learning was performed[24]. The model is therefore able to predict real-time data such as those conducted in this paper.[24]

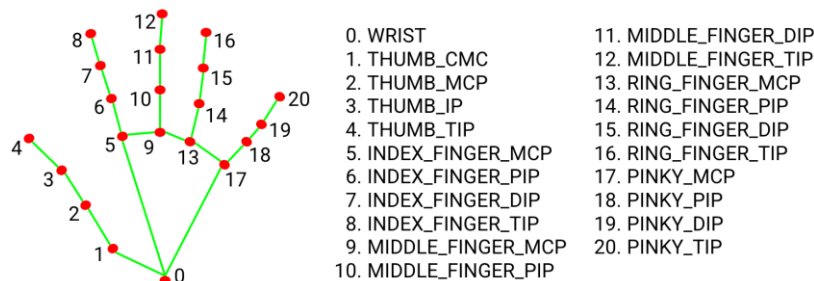
So for input the model accepts a cropped video frame without any depth input. Thus the model extracts the positions of the 3D points as it can sense if there is the presence of a face in the frame.

## 2.2.5 LEFT\_HAND\_LANDMARKS & RIGHT\_HAND\_LANDMARKS

In the case of hands, two models are used. Initially the same rules apply to both hands and in both cases we have a list of 21 hand landmarks[25]. These two models are the palm detection model and the hand landmark model.

The first task of the palm detector model is to find the starting position of each hand. This task is not particularly easy though. Although the face has high-contrast depths and patterns such as around the eyes and lips, the hands do not have such features[25]. So to initialize the model to find the starting point it usually relies on the pose of the person, the point of the arm, the posture of the body, etc. Despite the difficulties, the palm detection model achieves an accuracy of 95.7%. This is possible because firstly in this model a palm detector is trained and not a hand detector. As mentioned in the pose estimator model, rigid body points are much easier to detect compared to flexible ones. So it is easier to detect a palm that is not flexible than to find a hand that also contains fingers and creates a strong complexity in the training of the detection model.[25]

After detecting the palm, what remains is the detection of the hand. The model learns a consistent internal representation of the hand pose and is robust even to partially visible hands. In Fig.3 shows all the landmarks that can be detected by the model



**Fig. 3. all the Keypoints from Hand\_Landmarks**

### 3 SEQUENTIAL AND LONG-SHORT TERM MEMORY

#### 3.1 Sequential and its use

A sequential model is suitable for a simple stack of layers where each layer has exactly one input tensor and one output tensor. Sequential data includes text streams, audio clips, video clips, time series data, etc. In this paper, video data is presented. As mentioned above, openCV is used to capture video in real time and in collaboration with sequential we will be able to use frames from the movement to get data and lead to a result.

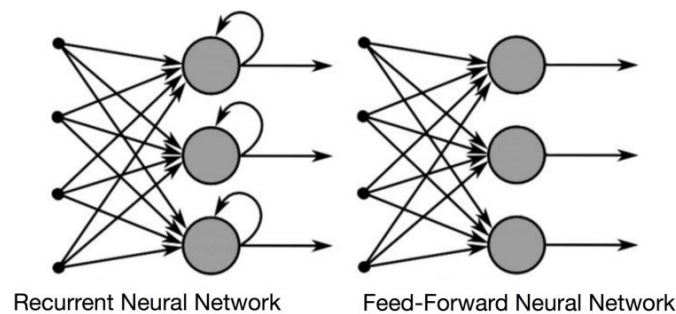
##### 3.1.1 Recurrent Neural Network

Recurrent neural networks is a modern algorithm for sequential data. It is used by Google's Siri for example and other well-known applications [26]. The above machine learning algorithm contains an internal memory that enables it to remember its input. This is essentially why it is the ideal algorithm for machine learning problems that deal with sequential data like speech, text, time series and much more. In this document the user will perform the gesture and accordingly the memory will save every previous frame. Through the mediapipe library and openCV the program will be able to precisely track the user's head, hands and body pose to save a set of frames and be able to do a prediction of the gesture the user performs.

So in a recurrent neural network (RNN), each time a frame is read, the information being processed is used to update the information transferred to the feedback links. This process continues until all the elements of the model have been read and the processed

information is passed on at each step. With each step the RNN produces an output that serves as a prediction of the model. When constructing an RNN the purpose is to ensure that all this processed information after all these steps does not decay. It is also important to emphasize that the error correction information must be able to propagate backwards through the same paths without degradation.

To understand a little better what recurrent neural networks are, we need to see how they work. Ideally a comparison between feed-forward and recurrent neural networks will clarify the situation a bit. As the name feed-forward neural network says, it is capable of feeding forward and moves only in one direction; from the input stroke i.e. to the output layer. These neural networks have no input memory and do not do so well at predicting what will come next. It also pays no attention to the time order of things happening. It is only capable of remembering its training. But unlike the recurrent neural network, all the information is recycled. When it makes a prediction it takes into account both the current input and what it has learned from previous inputs.



Let's see an example for better understanding. Let's say we want to generate a word and the neural returns the completed word before we even complete it. If we use a feedforward neural network, if we give it the word "Love" as input (ie get all the frames it needs to "build" the word), it will not be able to predict which coordinates will follow to form this gesture. Unlike a recurrent neural network, however, because as mentioned above, it includes internal memory. It is able to predict, and very accurately indeed, what comes next.

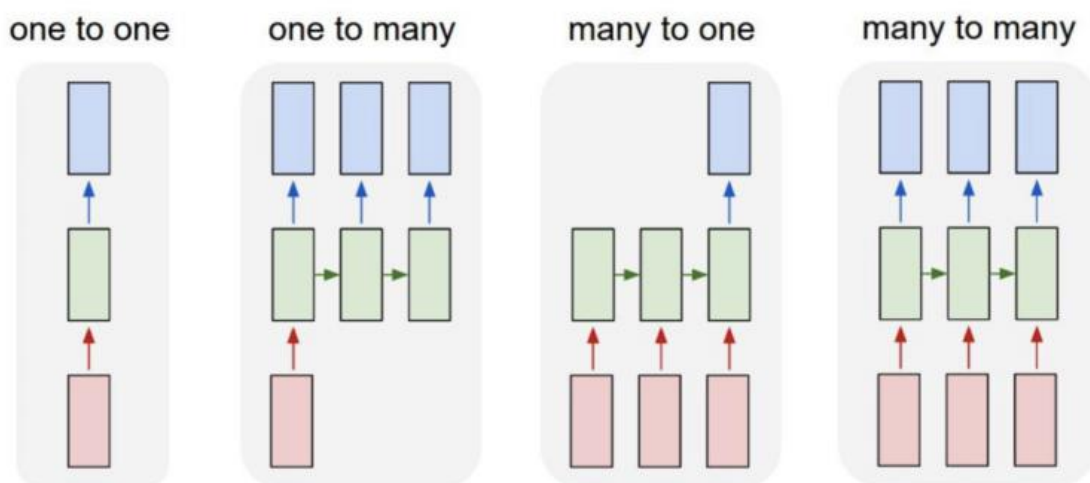
Therefore, an RNN has got two inputs: the present and the recent past. This is crucial because the data sequence contains critical information about what's next, and for this reason an RNN is capable of doing things that other algorithms cannot.

An RNN applies its weights to each current input as well as the previous one unlike a feed-forward which assigns a weight table to the beginning of the input and finally produces

the output. Regarding gradient descent and backpropagation through time an RNN will adjust its weights accordingly, of course it is important to note that there are 4 types of RNN[26] and these are:

- One to One
- One to Many
- Many to One
- Many to Many

In the figure(5) below we see that RNNs can map an input to an output or multiple inputs to an output or even multiple inputs to multiple outputs as opposed to feed-forwards which can only map an input to an output.



**Fig. 5. RNN inputs/outputs**

Usually an RNN has a short-term memory; but to avoid this it is combined with the LSTM and so they manage to have a long-term memory.

### **3.1.2 LSTM(long short-term memory)**

Long-term short-term memory (LSTM) is a special kind of artificial neural network and is found in the fields of artificial intelligence, deep learning and computational neuroscience[5]. LSTM is not a common neural network. As discussed above, among the categories of neural networks, feed-forward and recurrent neural networks, LSTM has



feedback connections. So we are talking about an RNN which beyond single data points(images) can process whole data sequences(speech or video).[5]In this paper the LSTM model is used and its use will be presented in the following sections. Besides, as it has been mentioned above, the prediction we want to achieve in this paper is about a sequence and more specifically real-time video.

Some examples of application of the LSTM are:

- Non-segmental linked handwriting recognition where each letter must be recognized as an individual symbol and not as part of a word[27]
- Machine translation speech recognition where each word must be transcribed independently[27]
- Robot control systems that must respond to events over time and not just once when an event occurs; and[27]
- Video games such as those played on handheld devices where player inputs are not directly mapped to motion commands for in-game characters; and finally[27]
- Health applications such as detecting heart arrhythmias from electrocardiograms (ECGs)[27]

If we analyze the name of the LSTM model we understand that a traditional RNN has both long-term memory and short-term memory. The connection weights in a traditional RNN change once per episode of training, analogous to how synaptic changes in the brain store long-term memories. The activation patterns in a traditional RNN change once per time-step, analogous to how electric firing patterns in the brain store short-term memories. The Long Short-Term Memory architecture tries to provide a short-term memory for RNN that lasts thousands of timesteps, thus "long short-term memory".[5]

In a time series in which some processes are performed there could be delays of unknown duration. This is where long short-term memory networks make the difference. They are responsible for classification, processing and making predictions based on time series data. LSTMs were first developed because a traditional RNN often faces a vanishing gradient problem during its training and LSTM has an advantage over simple RNNs[30].

Recurrent neural networks are a class of artificial neural networks that use a dynamic memory in the form of a state variable which is updated through the dynamic computational processes. This update is made possible by an additional layer which is called a "forget" gate and which allows the update, or non-update, of the state variable depending on whether it has

been updated before in that loop iteration or not[30].Below on the fig.6 the architecture of a standard LSTM is represented.

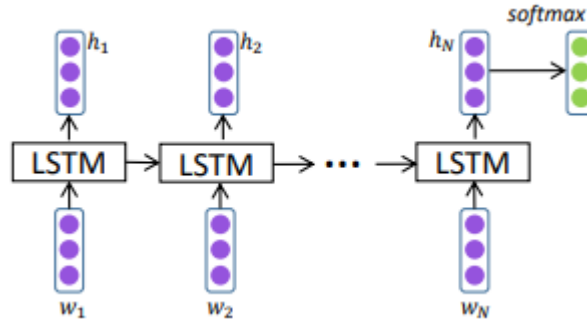


Fig.6

Essentially, every cell in the LSTM can be calculated as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \quad (1)$$

$$f_t = \sigma(W_f \cdot X + b_f) \quad (2)$$

$$i_t = \sigma(W_i \cdot X + b_i) \quad (3)$$

$$o_t = \sigma(W_o \cdot X + b_o) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot X + b_c) \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

In the model, we define the parameters  $W_f$ ,  $W_o$  and  $b_i$ ,  $b_f$ ,  $b_o$  to represent different transformations of the input and output gates with sigmoidal function  $\sigma$  and elementary multiplication  $\odot$ , respectively. The vector  $x_t$  contains the cell module inputs of LSTM, which are the embedding word vectors  $w_t$ . The vector representing the hidden layer of the LSTM cell is  $h_t$  [30]. We consider the last hidden vector  $h_N$  as the representation of a sentence and place it in the softmax layer for classification.

As mentioned above in an LSTM there are three gates. In the following fig(7) it's an illustration of the three gates. These gates play a pivotal role in how the neural network will handle the information. There are three cases. Whether to let new input in, whether to delete the information for understanding that it is not important or whether to let it affect the output in the current step. Notice in the figure that the gates are analogous in sigmoid form. This allows them to do backpropagation.

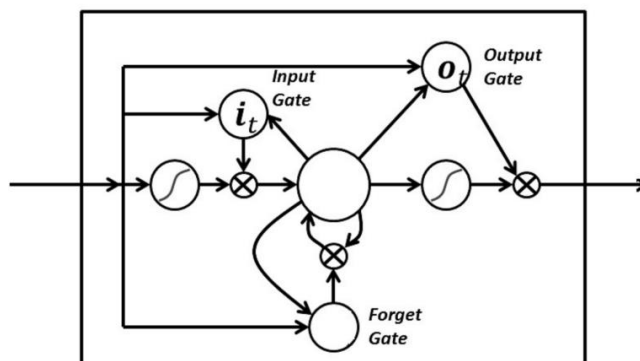


Fig.7

### 3.1.3 What is Backpropagation

The backpropagation algorithm is well known in machine learning because it is often used in cases such as the one discussed below. It is mainly used to compute the error gradient of a function with respect to the weights of a neural network. The purpose of the algorithm is to work backwards, hence the name, in order to find the partial derivative of the errors with respect to the weights.

The most common path that occurs in neural networks is forward propagation to obtain the output of the neural network model and check the correctness of this output. In this case, back propagation prevails in which we move backwards in the neural network to find the derivatives of the error with respect to the weights.

According to the aforementioned, the derivatives are then used by the gradient descent. The gradient descent is an algorithm that iteratively minimizes a given function; it then adjusts the weights up or down depending on which one reduces the error. This is how a neural network learns during the training process. The purpose of backpropagation is therefore to adjust the weights of the model during training.

### 3.1.4 Mathematical representation of the model

As mentioned above in the definition of the model and in the function, below is the mathematical presentation of the model.

Since the model is recursive, it is correspondingly defined for the whole length of the sequence. The initial state of the model is  $h_0$  and is mandatory for the first processing step of the model,  $t=1$ . It is then emphasized that the initial state must resemble the output from all subsequent processing steps which [26]. Finally with the result of  $h_0$  in terms of  $s_0$  this result is obtained and it is necessary to adapt it to the data.

$$h_0 = f(s_0)$$

So having the activation function of the model which is  $f$  we see that the parameters  $s_0$  are passed through it and produce the output  $h_t$  or which output of course represents the average for each passed step. The equation of the model is given below

$$h_t = f\left(\frac{\sum_{i=1}^t z(x_i, h_{i-1}) \circ e^{a(x_i, h_{i-1})}}{\sum_{j=1}^t e^{a(x_j, h_{j-1})}}\right)$$

Between the two models  $z$  and  $a$ , observe how the weighted average term is used in a genetic sequence model based on RNN. For each symbol in the sequence, the features  $x_i$  are encoded by the model  $z$ , while its recurrence relations provide the necessary context for encoding a sequence and are represented by  $h_{i-1}$ [26]. Thus, since the relative contribution of  $z$  is determined at each processing step,  $a$  serves as the attentional model; however, to generate the appropriate weighted average, the exponential terms of model  $a$  are normalized to the denominator.

In this example, only one model is considered, as  $z$  represents the characteristics of a dataset; it will be ruled by  $x_i$  not  $h_{i-1}$  if  $z$  is split into an unconstrained component containing only  $x_i$  and a constrained component containing the repeated terms  $h_{i-1}$ .

$$z(x_i, h_{i-1}) = u(x_i) \circ \tanh g(x_i, h_{i-1})$$

The features are encoded by the model of  $u$  which contains only  $x_i$ ; similarly the model for  $g$  contains the recurrent relations and is constrained between  $[-1, 1]$  by the function  $\tanh$ . The absolute magnitude of  $z$  is not changed, but it has the ability to control the sign of  $z$  which is carried out at each processing step by a separate model. Its purpose is to ensure that the information encoded by  $u$  does not simply accumulate but can negate the information encoded by previous processing steps.[26]

$$\begin{aligned} u(x_i) &= W_u \cdot x_i + b_u \\ g(x_i, h_{i-1}) &= W_g \cdot [x_i, h_{i-1}] + b_g \\ a(x_i, h_{i-1}) &= W_a \cdot [x_i, h_{i-1}] \end{aligned}$$

Taking the output from each processing step after a sequence is executed, it is passed through a fully connected neural network layer to predict the label. To adjust the model parameters with the minimum possible error between the actual and predicted label, gradient descent methods can be used.

$$\begin{aligned} n_t &= \sum_{i=1}^t z(x_i, h_{i-1}) \circ e^{a(x_i, h_{i-1})} \\ d_t &= \sum_{j=1}^t e^{a(x_j, h_{j-1})} \end{aligned}$$

Because the sum for  $n_t$  and  $d_t$  can be recalculated as an iteration relation, it is easy to define them iteratively. Let  $n_0 = 0$  and  $d_0 = 0$ . [26]

$$\begin{aligned} n_t &= n_{t-1} + z(x_t, h_{t-1}) \circ e^{a(x_t, h_{t-1})} \\ d_t &= d_{t-1} + e^{a(x_t, h_{t-1})} \end{aligned}$$

To avoid having to redo large amounts of work during each step, the values  $n_{t-1}$  and  $d_{t-1}$  can be saved, so that they may be used in generating the next step's numerator,  $n_t$ . This can be accomplished by using equation (2), where  $h_t$  is equal to the output of equation .

$$h_t = f\left(\frac{n_t}{d_t}\right)$$

We can implement the RWA model using equations (1) and (3)–(6), as follows:

$$\begin{aligned}
h_0 &= f(s_0), \quad n_0 = 0, \quad d_0 = 0 \\
u(x_t) &= W_u \cdot x_t + b_u \\
g(x_t, h_{t-1}) &= W_g \cdot [x_t, h_{t-1}] + b_g \\
a(x_t, h_{t-1}) &= W_a \cdot [x_t, h_{t-1}] \\
z(x_t, h_{t-1}) &= u(x_t) \circ \tanh g(x_t, h_{t-1}) \\
n_t &= n_{t-1} + z(x_t, h_{t-1}) \circ e^{a(x_t, h_{t-1})} \\
d_t &= d_{t-1} + e^{a(x_t, h_{t-1})} \\
h_t &= f\left(\frac{n_t}{d_t}\right)
\end{aligned}$$

For a better understanding of the mathematical operation of Sequential it is necessary to know that the model is run recursively on an entire sequence; in the above picture we see that for each symbol its attribute is contained in  $x_t$  and the parameters  $s_0$ ,  $W_u$ ,  $b_u$ ,  $W_g$ ,  $b_g$  and  $W_a$  are determined by fitting the model to a training data set [26]. To avoid increasing or decreasing the exponential terms the numerator and denominator must be rescaled;

## 4 DATA ACQUISITION

The system that has been developed on this project will be able to detect in real time the signals carried out by the signer. The user is provided with the possibility to create his own dataset within the system using the OpenCV libraries as mentioned above its functions, and in Python language. Using OpenCV in combination with Medipipe Holistic we will be able to create a dataset to then use it in training the model.

### 4.1 Dependencies for Data Collection

By combining the OpenCV library with the Google Mediapipe Holistic library I managed to achieve a result where by opening the web camera that I have connected to the computer I can display on the screen graphic elements that connect all the keypoints of my body. The openCV library gives us the ability to have the camera open and facing the signifier and the mediapipe library understands what the important points are to identify and set the keypoints needed along the way so that we can train our model depending on the coordinates where the signifier stands. At the same time through the camera and all these keypoints in each of my movements I can see in which coordinates each point of my body is located at any given moment. Thus training my model, in addition to the complete movement of the gesture will help

me and the distances that have the limbs of my body to make a better prediction of the gesture I carry out.

For the data acquisition dependencies such as cv2,os,time have been used. The dependency os was used to help working with the file paths as you will see in 3.3. With dependency time in Python, time can be represented in many ways in the code, such as objects, numbers and strings. In this project it is used to add breaks between capturing images to provide the right amount of time to hand movements to get the data more correctly.

## **4.2 Landmarks acquisition**

Before starting the data collection we need to test if the head, hand and pose landmarks are displayed correctly when the camera is activated and return the correct data.

Then we record a frame and we can actually see in this frame which landmarks and if the mediapipe model works correctly.

To see this we keep the frame mentioned above and from the mediapipe we need to fetch the results with `extract_keypoints(results)`. So this will return an array with all the points within the frame

After everything is working properly, before we can start collecting the data we need to create the classes that we will build so that the data distribution is done correctly and the frames get the right labels.

## **4.3 How data was obtained**

In this paper we've created 3 main datasets. These datasets are the 12 months of the year in one set. The next one is 3 main verbs of greek language. These are the words:

- Love
- Understand
- Study

And lastly I've created the signal of danger which could be used in various situations to prevent to some extent the suspicion of a crime.

For each word we want to create data, 30 frames are taken. The frames are captured every 2 seconds providing time for recording the gestures with a small difference each time and a 2-second interval is given between two gestures. That is, to change the sign of one month to the

sign of a different month a two-second interval is given. The recorded images are stored in the corresponding folders.

On the present project we used 12 classes for the months. In order to distribute correctly all the files using the os dependency it will automatically create 30 folders and each folder will contain 30 frames. Same applies for all kinds of dataset creation using the present project code.

For the creation of the dataset, videos posted by the Greek Sign Language Center on their website were used. As the system we are building in this paper allows us to create our own dataset, so as a reference these videos were used and the dataset creator reproduces the words as correctly as possible to be accurate then at the point of prediction. In this website, users can find many videos containing either single words or whole sentences or conversations. It is mainly used for the education of students learning Greek Sign Language but it is also aimed at the rest of the public. So in case we want to collect more data, the above website will guide us to find out how they carry out the signal.

## **5 BUILD THE MODEL**

Above we have seen a brief description of what a Sequential is, what an LSTM is and where RNNs are used.

The model is using LSTM layers with different numbers of units and return\_sequences settings, followed by fully connected layers to process sequential data, the model is trained using Adam optimizer and categorical cross-entropy loss function and evaluated using categorical accuracy. The LSTM layers are used to process sequential data and the fully connected layers are used to make the final predictions. The activation function used is ReLU which introduces non-linearity in the model.

### **5.1.1 LSTM**

The model used in this essay is a type of Recurrent Neural Network (RNN) called a Long Short-Term Memory (LSTM) network. The model is created using the Sequential API from the Keras library, which allows for easy building of neural networks by stacking layers on top of each other.



The first layer in the model is an LSTM layer with 64 units, which is the number of memory cells in the LSTM layer. The input shape of this layer is defined as (30, 1662), which means that the input data is a 3D array with 30 time steps and 1662 features. The `return_sequences` parameter is set to `True`, which means that the output of this layer will be a 3D array with the same shape as the input, and can be used as input for the next layer. The activation function used is 'relu' which stands for rectified linear unit, it is used to introduce non-linearity in the model.

The second layer is also an LSTM layer with 128 units and `return_sequences=True`, this layer will process the output from the first LSTM layer and produce a new output.

The third layer is also an LSTM layer with 64 units and `return_sequences=False`, this layer will process the output from the second LSTM layer and produce a 2D array as output.

The fourth and fifth layers are fully connected layers (also known as dense layers) with 64 and 32 units respectively. The activation function used in these layers is 'relu' which introduces non-linearity in the model.

The final layer is a dense layer with the number of units equal to the number of actions, and the activation function is set to 'softmax', this layer will produce the final output of the model.

The model is then compiled using the Adam optimizer, which is a variant of stochastic gradient descent that adapts the learning rate for each parameter. The loss function used is 'categorical\_crossentropy', which is used for multi-class classification problems, and the metrics used to evaluate the model is 'categorical\_accuracy'

Finally, the model is trained using the `fit` function with the input data, output data, number of epochs and a callback function. The callback function is Tensorboard, which is a visualization tool for monitoring the training progress and performance of the model.

### **5.1.2 LSTM Usage**

LSTM layers are a type of recurrent neural network (RNN) that are particularly well-suited to processing sequential data, such as time series data or natural language data. LSTMs

have the ability to maintain information in memory over a longer period of time, which allows them to better capture long-term dependencies in the data.

The first LSTM layer with 64 units is used as the first step to process the input data, it takes the input data and produces a new output which is then passed to the second LSTM layer.

The second LSTM layer with 128 units, processes the output from the first LSTM layer and produces a new output which is then passed to the third LSTM layer.

The third LSTM layer with 64 units, processes the output from the second LSTM layer, and produces the final output of the model.

The different number of units in each LSTM layer can be seen as different levels of abstraction, the first layer captures low-level features, the second layer captures more complex features, and the final layer captures the most complex features. This architecture allows the model to learn hierarchical representations of the data, which can improve the ability of the model to generalize to new data.

Moreover, the `return_sequence` parameter is set to `True` for the first two layers and set to `False` for the last layer. This is because the first two layers are used to process the input data and produce a new output and the final layer is used to produce the final output of the model.

In summary, using multiple LSTM layers allows the model to learn hierarchical representations of the data, which can improve the ability of the model to generalize to new data. The different number of units in each LSTM layer and the different `return_sequence` settings are used to process the input data and produce the final output of the model.

### **5.1.3 LSTM Layer units usage**

The number of units in an LSTM layer, also known as the number of memory cells, determines the capacity of the layer to store and process information. A larger number of units in an LSTM layer generally corresponds to a larger capacity to store and process information. However, a larger number of units also increases the number of parameters in the model, which can increase the risk of overfitting if the model is not sufficiently regularized.

The first LSTM layer has 64 units. This number of units was chosen because it strikes a balance between the model's capacity to store and process information and the risk of

overfitting. 64 units is a relatively small number of units, which reduces the risk of overfitting, but it is still large enough to allow the model to capture important patterns in the data.

The second LSTM layer has 128 units, which is twice the number of units in the first LSTM layer. This is done because the second LSTM layer is processing the output from the first LSTM layer and therefore needs to have more capacity to store and process information. This increased capacity allows the second LSTM layer to capture more complex features in the data.

The last LSTM layer has 64 units, the same number of units as the first LSTM layer. This is because the last LSTM layer is used to produce the final output of the model, and it does not need to have as much capacity to store and process information as the previous layers.

In summary, the number of units in each LSTM layer is chosen based on the complexity of the problem and the complexity of the data. The second LSTM layer has twice the number of units as the first one because it needs to process more complex features, and the last LSTM layer has the same number of units as the first one because it doesn't need to have as much capacity to store and process information, as it is producing the final output of the model.

#### **5.1.4 Dense Layers**

In this essay we are using three sets of dense layers after the LSTM layers. Dense layers, also known as fully connected layers, are used to process the output of the LSTM layers and make predictions. Each dense layer contains a certain number of neurons, which are used to make computations on the input data.

The first dense layer contains 64 neurons, the second dense layer contains 32 neurons and the last dense layer contains the number of neurons equal to the number of actions in the dataset. The use of multiple dense layers allows the model to learn more complex representations of the data, by combining the output of the previous layers.

Each dense layer is connected to the previous one, and the output of each dense layer is passed as input to the next one. In this way, the output from the previous layers are used to

refine the predictions by the next layers. As the output of the last LSTM layer is fed into the first dense layer, the first dense layer is able to use the output of the LSTM layers to make more accurate predictions.

In summary, the use of multiple dense layers allows the model to learn more complex representations of the data by combining the output of the previous layers. The number of neurons in each dense layer is chosen based on the complexity of the problem and the complexity of the data, and each dense layer is connected to the previous one, and the output of each dense layer is passed as input to the next one.

The activation function used in the first three dense layers is ReLU (Rectified Linear Unit), which is a popular choice for dense layers because it is computationally efficient and has been shown to work well in many cases. The ReLU function is defined as  $f(x) = \max(0, x)$ , where  $x$  is the input to the function. ReLU has the property of being non-linear, which makes it suitable for deep neural networks.

However, on the last dense layer, the activation function used is softmax. Softmax function is commonly used in the output layer of a classification model. The softmax function is used to convert the output of the last dense layer into a probability distribution over the possible classes. The output of the softmax function is a probability distribution over the classes, where each element represents the probability that the input belongs to that class.

The reason why softmax is used on the last dense layer is that we are trying to classify the input into different classes. Softmax function is used to calculate the probability of each class, and the class with the highest probability is considered as the prediction of the model.

## **5.2 Models created**

In the present work and since it is possible to construct our own dataset, 3 different datasets were created. The datasets were.

- The months in Greek sign language.
- 3 Basic verbs in Greek sign language (Love, Understand, Study)

So having this data and having already built the basic structure of the model, we fed the model with all 2 different datasets to see in which cases we would have better results.

After creating these models, these could be used for many purposes, such as:

- Training system for people who want to learn sign language and how well they perform the gestures according to the model that would obviously be trained by an experienced sign language teacher.
- Automatic notification of a system in case of detection of the gesture of danger.

### **5.3 Fit The Model**

The optimizer is used to update the model's parameters during training. The Adam optimizer is a popular choice for deep learning models, it is computationally efficient and has been shown to work well in many cases. After experimentation with different optimizers Adam was the best one to use.

The loss function is used to measure the difference between the predicted output and the true output during training. The loss function is used to guide the optimizer in adjusting the model's parameters to minimize the difference between the predicted output and the true output. The loss function used is 'categorical\_crossentropy', this loss function is commonly used in multi-class classification problems, where the goal is to predict one of several possible outcomes.

The metrics are used to evaluate the performance of the model during training and testing. The metrics used are 'categorical\_accuracy', this metric is used to evaluate the performance of the model on multi-class classification problems, it calculates the percentage of correct predictions.

The `model.fit()` function is used to train the model. It takes three main arguments: the training data (`X_train`), the true labels (`y_train`) and the number of training epochs.

The training data (`X_train`) is the input that the model will use to learn the relationships between the input and the output. The true labels (`y_train`) are the correct outputs that the model should produce for the given inputs. The model will use these labels during training to adjust its parameters to minimize the difference between its predictions and the true labels.

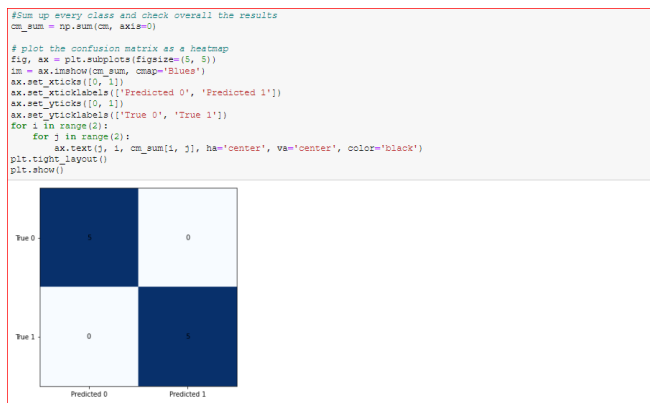
The number of training epochs is the number of times the model will see the entire training dataset. An epoch is a single pass through the entire training dataset. During each epoch,

the model will make predictions for each input in the training dataset, and the predictions will be used to update the model's parameters.

The callbacks parameter is used to specify a list of callbacks that will be executed during training. The only callback is `tb_callback`, which is a callback for TensorBoard, a tool for visualizing training progress and performance. This callback allows to log the model's performance during training and visualize it in TensorBoard, which can be useful for monitoring the training progress, identifying overfitting and underfitting, and debugging the model.

## 5.4 Model Performance

The 2 models we created performed very well and we will see this below in the tables listed in which we see the confusion matrix of the model with verbs in figure 8 and its metrics in figure 9.



**Fig. 8. 3 verbs model confusion matrix**

We observe for the confusion matrix of the first model that the results are 5 true positives and 5 false negatives so we have 100% successful predictions.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Assuming y_true and y_pred are your true and predicted labels
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

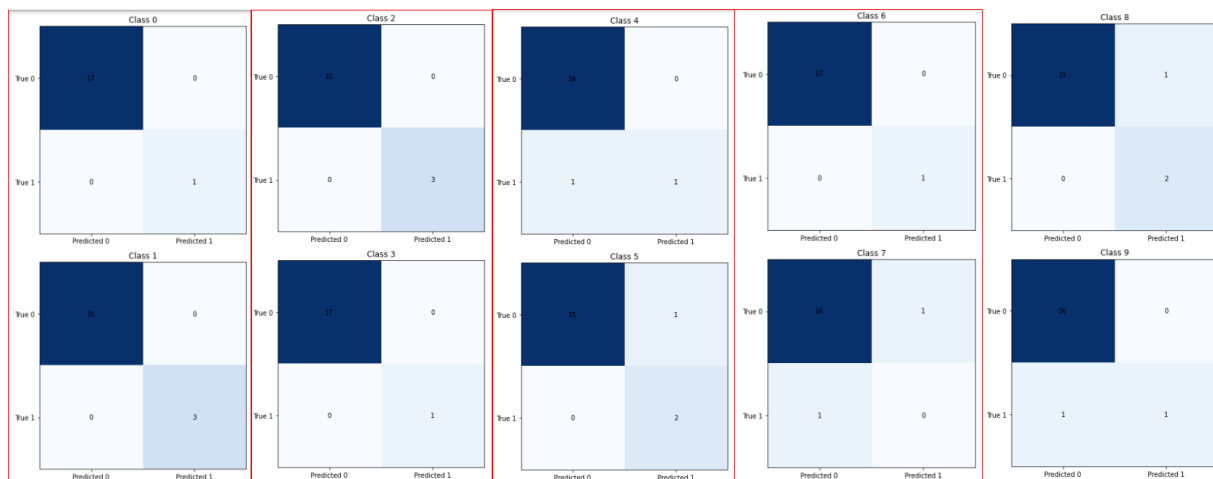
**Fig. 9. 3 verbs model metrics**

Next we see in figure 10 the confusion matrix of the model with the months.



**Fig. 10. months confusion matrix(overall)**

Here we see that again the predictions were quite successful. Below is a table showing specifically for each class.



**Fig. 11. months confusion matrix(one by one)**

Finally, in the metrics (figure 12) we observe that the model performed well also in the case of months although it had to be trained in more classes than the verbs which were only three.

```

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Assuming y_true and y_pred are your true and predicted labels
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

Accuracy: 0.83
Precision: 0.82
Recall: 0.83
F1 Score: 0.80

```

**Fig. 12. months metrics**

## 6 EVALUATION AND TEST IN REAL TIME

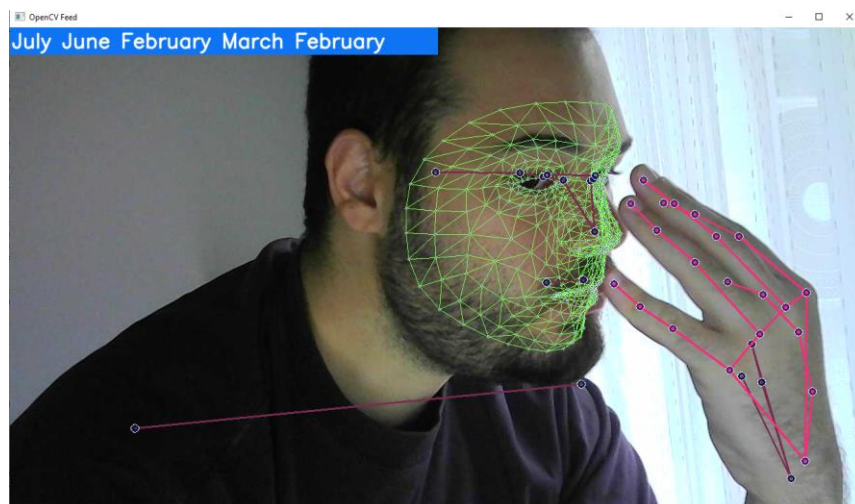
Finally to evaluate our model to see how well it works we import two main evaluation metrics. These are `multilabel_confusion_matrix` and `accuracy_score`. So after the evaluation we create a confusion matrix and we see in the table depending on how many classes we have used in the model we can distinguish the true positives, false positives, false negatives, true negatives etc. Then we see the `accuracy_score`, which the less classes we have used the better performing our model. In case we have 12 classes like the example with the months we will see that our model in the accuracy score will be close to 0.7, while in case we have 2-3 classes our model will have an accuracy score of 0.85-0.9. Surely many factors play a role such as how the dataset was obtained, what tools were used, whether the creator of the dataset was familiar with the Greek sign language, etc. Surely all these factors would give a better dataset which in turn would give even better results.

To be able to make a prediction we first create an empty array and in it for each new frame created in the camera we add it to the array we created.

Then in order for the user to be able to see in real time the results of the gestures he performs, we make a logic in which the program prints the word that it considers closest to the gesture he makes.

The model prints from the right-hand side each new prediction it makes and keeps the previous one in memory with a maximum number of 5 words. At each user movement the model will make a prediction according to the sequence of movements as shown in figure 13





**Fig. 13. Sign Detection of Month February**

As for the model with the months. We have an accuracy score of 0.83 and through the predictions we make it turns out that the results are not perfect but they are quite satisfactory. It seems that in the months where the signals are not at all similar to other months the prediction is much easier and more targeted, while if we take for example the summer months of June-July which differ only slightly in their signal we will see that the model has difficulties in predicting the correct month.

This can of course also be due to the data collection carried out and how correctly the signals were conducted. We also see that the distance at which the signals are conducted plays an important role; the model has difficulty identifying the signal if the user is far enough away from the camera.

## 7 CONCLUSION

Sign languages are types of visual languages, i.e. they use body movements, facial expressions and rely on detail to get the message out of the sender and into the hands of the receiver. Sign languages are important for people with disabilities because they are their primary means of communication. It is the only way they can share their feelings with others. The problem in modern society is that not everyone has the knowledge to communicate in sign languages. Above we have discussed how this limitation can be overcome by using automated sign language recognition systems that will be able to translate the gestures of deaf and mute people into a commonly spoken language. In this paper this was done with the object detection

API of tensorflow. The system was trained on the Greek sign language dataset for some phrases/words/emotions. For data acquisition the images were obtained using OpenCV and Python. The model shows a success rate close to 80% which always depends on how many classes are collected in the data and how successfully the data collection is done. The system could be used for all sign languages since, as mentioned above, each country has its own sign language.

Mediapipe Holistic with LSTM layers was used for 3 main reasons:

1. We needed much less data to train the model as mediapipe holistic did enough work for us
2. Since we had all the keypoints it was also much faster to train the model.
3. Since the dataset was not too big and the layers we made after the preprocess did not exceed the number of layers we have much faster detections(model.summary() -> we can see how many parameters we have)

```

Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
lstm (LSTM)                  (None, 30, 64)             442112
-----
lstm_1 (LSTM)                (None, 30, 128)           98816
-----
lstm_2 (LSTM)                (None, 64)                 49408
-----
dense (Dense)                (None, 64)                 4160
-----
dense_1 (Dense)              (None, 32)                 2080
-----
dense_2 (Dense)              (None, 3)                  99
-----
Total params: 596,675
Trainable params: 596,675
Non-trainable params: 0
    
```

## Bibliography – References – Online sources

{Bibliography and references should be given in one of the following forms:

1. W. Sandler, D. Lillo-Martin, “Sign language and linguistic universals”. Cambridge University Press, 2006
2. Z. Yang, Z. Shi, X. Shen, and Y.-W. Tai, “Sf-net: Structured feature network for continuous sign language recognition,” *arXiv preprint arXiv:1908.01341*, 2019
3. C. Padden, “Verbs and role-shifting in american sign language,” in *Proceedings of the fourth national symposium on sign language research and teaching, vol. 44. National Association of the Deaf Silver Spring, MD, 1986, p. 57*

4. H. Cooper, B. Holt, and R. Bowden, "Sign language recognition," in *Visual Analysis of Humans*. Springer, 2011, pp. 539–562.
5. J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.
6. C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1933–1941.
7. Rautaray, S.S.: *A Real Time Hand Tracking System for Interactive Applications*. *Int. J. Comput. Appl.* 18, 975–8887 (2011).
8. Zhang, Z., Huang, F.: *Hand tracking algorithm based on super-pixels feature*. *Proc. - 2013 Int. Conf. Inf. Sci. Cloud Comput. Companion, ISCC-C 2013*. 629–634 (2014). <https://doi.org/10.1109/ISCC-C.2013.77>
9. Mohandes, M., Aliyu, S., Deriche, M.: *Arabic sign language recognition using the leap motion controller*. *IEEE Int. Symp. Ind. Electron.* 960–965 (2014). <https://doi.org/10.1109/ISIE.2014.6864742>.
10. Enikeev, D.G., Mustafina, S.A.: *Sign language recognition through Leap Motion controller and input prediction algorithm*. *J. Phys. Conf. Ser.* 1715, 012008 (2021). <https://doi.org/10.1088/1742-6596/1715/1/012008>.
11. Gaus, Y.F.A., Wong, F.: *Hidden Markov Model - Based gesture recognition with overlapping hand-head/hand-hand estimated using Kalman Filter*. *Proc. - 3rd Int. Conf. Intell. Syst. Model. Simulation, ISMS 2012*. 262–267 (2012). <https://doi.org/10.1109/ISMS.2012.67>.
12. Suharjito, Anderson, R., Wiryana, F., Ariesta, M.C., Kusuma, G.P.: *Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on InputProcess-Output*. *Procedia Comput. Sci.* 116, 441–448 (2017). <https://doi.org/10.1016/J.PROCS.2017.10.028>.
13. Cui, R., Liu, H., Zhang, C.: *A Deep Neural Framework for Continuous Sign Language Recognition by Iterative Training*. *IEEE Trans. Multimed.* 21, 1880–1891 (2019). <https://doi.org/10.1109/TMM.2018.2889563>.
14. Bantupalli, K., Xie, Y.: *American Sign Language Recognition using Deep Learning and Computer Vision*. *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*. 4896–4899 13 (2019). <https://doi.org/10.1109/BIGDATA.2018.8622141>.

15. Hore, S., Chatterjee, S., Santhi, V., Dey, N., Ashour, A.S., Balas, V.E., Shi, F.: *Indian Sign Language Recognition Using Optimized Neural Networks*. *Adv. Intell. Syst. Comput.* 455, 553–563 (2017). [https://doi.org/10.1007/978-3-319-38771-0\\_54](https://doi.org/10.1007/978-3-319-38771-0_54).
16. . Kumar, P., Roy, P.P., Dogra, D.P.: *Independent Bayesian classifier combination based sign language recognition using facial expression*. *Inf. Sci. (Ny)*. 428, 30–48 (2018). <https://doi.org/10.1016/J.INS.2017.10.046>.
17. Huang, J., Zhou, W., Zhang, Q., Li, H., Li, W.: *Video-based Sign Language Recognition without Temporal Segmentation*
18. D. Mehta, H. Rhodin, D. Casas, O. Sotnychenko, W. Xu, and C. Theobalt. *Monocular 3d Human Pose Estimation Using Transfer Learning and Improved CNN Supervision*. *arXiv preprint arXiv:1611.09813*, 2016.
19. I. Oikonomidis, N. Kyriazis, and A. A. Argyros. *Efficient model-based 3d tracking of hand articulations using Kinect*. In *British Machine Vision Conference (BMVC)*, volume 1, page 3, 2011.
20. G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. *Coarse-to-Fine Volumetric Prediction for Single-Image 3d Human Pose*. *arXiv preprint arXiv:1611.07828*, 2016.
21. B.Garcia, S. Viesca, “*Real-time American Sign Language Recognition with Convolutional Neural Networks*” 2016.
22. Ammar Anuar, Khairul Muzzammil Saipullah, Nurul Atiqah Ismail, Yewguan Soo “*OpenCV Based Real-Time Video Processing Using Android Smartphone*”
23. Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, Matthias Grundmann “*BlazePose: On-device Real-time Body Pose tracking*”
24. Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, Matthias Grundmann “*Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs*”
25. Wei Liu , Dragomir Anguelov , Dumitru Erhan , Christian Szegedy , Scott Reed , Cheng-Yang Fu , Alexander C. Berg “*SSD: Single Shot MultiBox Detector*”
26. Jared Ostmeyer, Lindsay Cowell “*Machine Learning on Sequential Data Using a Recurrent Weighted Average*”
27. Vasantha Kumar Velu, Sendhilkumar Selvaraju. “*Developing a Conceptual Framework for Short Text Categorization using Hybrid CNN- LSTM based Caledonian Crow Optimization*” , *Expert Systems with Applications*, 2022

28. Luis Quesada, Gustavo López, Luis Guerrero. "Automatic recognition of the American sign language fingerspelling alphabet to assist people living with speech or hearing impairments" , *Journal of Ambient Intelligence and Humanized Computing*, 2017
29. "Intelligent Data Engineering and Automated Learning – IDEAL 2018" , Springer Science and Business Media LLC, 2018
30. Jun Xie, Bo Chen, Xinglong Gu, Fengmei Liang, Xinying Xu. "Self-Attention-Based BiLSTM Model for Short Text Fine-Grained Sentiment Classification" , *IEEE Access*, 2019
31. Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, Yoshua Bengio "Blocks and Fuel: Frameworks for deep learning"