NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Dynamic sign language recognition using deep learning

Stylianos K. Psara

Supervisor: Panagiotis Stamatopoulos, Assistant Professor

ATHENS

MAY 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση δυναμικής νοηματικής γλώσσας με χρήση βαθιάς μάθησης

Στυλιανός Κ. Ψαρά

Επιβλέπων: **Παναγιώτης Σταματόπουλος,** Επίκουρος καθηγητής

ΑΘΗΝΑ

ΜΑΙΟΣ 2023

BSc THESIS

Dynamic sign language recognition using deep learning

Stylianos K. Psara
S.N.: 1115201800226

SUPERVISOR: **Panagiotis Stamatopoulos, Assistant Professor**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση δυναμικής νοηματικής γλώσσας με χρήση βαθιάς μάθησης

Στυλιανός Κ. Ψαρά
Α.Μ.: 1115201800226

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Παναγιώτης Σταματόπουλος,** Επίκουρος Καθηγητής

ABSTRACT

Improving communication for hearing-impaired communities proves to be an extremely challenging task for scientists and researchers nowadays. Deaf people have to use an alternative to verbal speech communication such as sign language in order to interact with the world around them. In addition, the existence of over 200 sign languages around the world makes communication for people with hearing loss much harder. Thus, the need to develop an automatic dynamic sign language translator through AI becomes a necessity.

Vision-based sign language recognition aims to improve the accessibility and inclusivity of those who communicate through hand gestures. However, the recognition of complicated gestures that combine facial expression, hand gestures and body movements, in addition of an environment with background noises can be proved very challenging.

In this thesis we were able to experiment with multiple deep learning methods on the public DSL10_Dataset. Specifically, we applied the pretrained model VGG_16 for feature extraction based on pattern recognition and the MediaPipe framework for feature extraction based on the key-points of hands and pose. Finally, we evaluate both approaches using the proposed LSTM and GRU models for classification.

Our results show that the combination of the pretrained model VGG_16 for feature extraction and the proposed LSTM for classification achieved a recognition accuracy of 96.44% in comparison with others.

SUBJECT AREA: Deep learning

KEYWORDS: Artificial Intelligence, American sign language (ASL), VGG-16, MediaPipe, Long short-term memory (LSTM), Gated recurrent unit (GRU)

ΠΕΡΙΛΗΨΗ

Η βελτίωση της επικοινωνίας για άτομα με προβλήματα ακοής αποτελεί ένα εξαιρετικά δύσκολο έργο για τους επιστήμονες στις μέρες μας. Οι κωφοί αναγκάζονται να χρησιμοποιήσουν εναλλακτικούς τρόπους επικοινωνίας, όπως η νοηματική γλώσσα, προκειμένου να αλληλοεπιδράσουν με τους ανθρώπους γύρω τους. Εντούτοις, το γεγονός ότι υπάρχουν πάνω από 200 γλώσσες νοηματικής σε όλο τον κόσμο, κάνει την επικοινωνία για άτομα με απώλεια ακοής πολύ πιο δύσκολη. Συνεπώς, αυξάνεται η ανάγκη για ανάπτυξη ενός αυτόματου μεταφραστή νοηματικής γλώσσας μέσω της τεχνητής νοημοσύνης.

Η αναγνώριση νοηματικής γλώσσας μέσω της μηχανικής μάθησης στοχεύει στη βελτίωση της ποιότητας ζωής των ανθρώπων με ακουστική δυσλειτουργία. Ωστόσο, η αναγνώριση περίπλοκων νοημάτων τα όποια ερμηνεύονται μέσω του συνδυασμού της έκφρασης του προσώπου, της κίνησης των χεριών και της στάσης του σώματος, σε συνδυασμό με ένα περιβάλλον το οποίο μπορεί να μπερδέψει τον μεταφραστή αποτελεί μεγάλη πρόκληση για τους ερευνητές.

Σε αυτή την εργασία είχαμε την δυνατότητα να πειραματιστούμε με ποικίλες μεθόδους μηχανικής μάθησης χρησιμοποιώντας τη δημόσια βάση δεδομένων DSL10_Dataset. Συγκεκριμένα, εφαρμόσαμε το προ-εκπαιδευμένο μοντέλο VGG-16 για εξαγωγή χαρακτηριστικών προερχόμενων από την αναγνώριση των μοτίβων και το MediaPipe για εξαγωγή χαρακτηριστικών προερχόμενων από τα key-points των χεριών και του σώματος. Τέλος, αξιολογούμε και τις δύο προσεγγίσεις χρησιμοποιώντας τα προτεινόμενα μοντέλα LSTM και GRU για ταξινόμηση.

Τα αποτελέσματά μας δείχνουν ότι ο συνδυασμός του προ-εκπαιδευμένου μοντέλου VGG_16 για εξαγωγή χαρακτηριστικών και του προτεινόμενου μοντέλου LSTM για ταξινόμηση πέτυχε ακρίβεια 96,44% η οποία ξεπέρασε τους άλλους συνδυασμούς που εφαρμοστήκαν σε αυτή την εργασία.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Βαθιά μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Τεχνητή νοημοσύνη, Αμερικάνικη νοηματική γλώσσα, VGG-16, MediaPipe, LSTM, GRU

AKNOWLEDGMENTS

I would like to express my gratitude to Mr. Stamatopoulos, my supervisor, for his constant supervision and direction throughout this process, as well as his prompt responsiveness.

Moreover, I would like to thank my family and friends for their support through all the years of my undergraduate studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis was written as a part of the BSc program of studies at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens.

1. INTRODUCTION

The communication of hearing-impaired individuals proves a major issue nowadays. According to the World Health Organization (WHO), almost 446 million people worldwide have impaired hearing loss and studies reveal that in the future this number might increase rapidly [1]. Deaf people have to use an alternative to verbal speech communication, such as Sign language (SL), for example, however this only allows them to communicate with signers. According to World Federation, there are over 200 SLs in the world, from which Indian, American and British are the most common, meaning communication between signers is already difficult, not to mention with the hearing majority that don't understand sign language [2] [1].

The meaning of a sign can be extracted mainly from a combination of hand gestures, body motions, and facial expressions. In addition, subtle differences can lead to different meanings. For example, as shown in Fig.1, the signs for "dance" and "read" only differ in the orientation of one's hands [3][4].

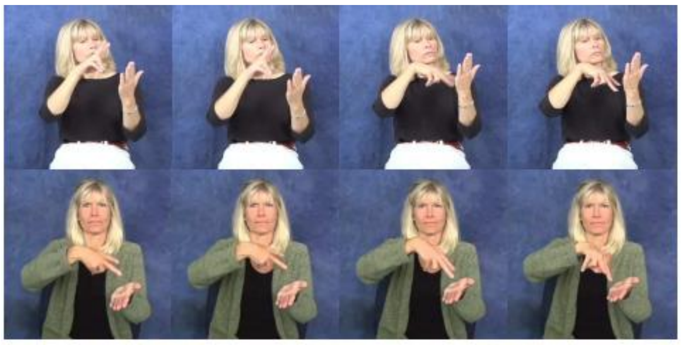


Figure 1: ASL signs "read" (top row) and "dance" (bottom row)[3].

Thus, the desire to develop an automatic sign language translator that could convert hand kinematics and facial expressions into text or sound and provide better communication opportunities to the deaf community, becomes more intense for scientists and researchers; an effective way to do that is through Artificial intelligence and especially deep learning [5][6].

Sign recognition has two main methods of capturing gestures, the sensor-based and the vision-based. In the sensor-based approach, the user must wear special equipment, such as colored-gloves or special-gloves that allow the obtainment of finger key-point data, in addition to a motion capture system that captures the signs [2][7][6]. On the other hand, in the vision-based approach, the equipment required is closer to a real time problem and more suitable for day-to-day situations, because the system relies on the deep learning techniques that classify and predict data which consists of images and videos, captured simply by a normal smartphone camera [6]. Another method is to combine these two and make a hybrid approach using both gloves and a camera. This approach is not commonly used because the cost and computational overheads are much higher, and the researchers don’t prefer it [1], [7].

Gestures can be divided into two categories, Static and Dynamic. Static gesture recognition is a pattern recognition problem where you extract the information from a single image that represents a specific shape and pose of the hands. On the other hand, a Dynamic gesture is a moving gesture, represented by a series of slightly different images which may also contain head movement, making dynamic gestures recognition more complex and challenging [4][7].

The objective of this thesis was to apply and compare various deep learning methods in order to provide better communication between the deaf community and hearing majority.

2. BACKGROUND

2.1 Artificial Intelligence (AI)

Artificial intelligence (AI) is currently one of the most popular subjects in Computer science, referring to the ability of a machine to have human intelligence. AI is connected to the idea of a machine accomplishing human tasks, such as “learning“ and “problem solving”. The Birth of Artificial intelligence begins in 1950 from the “father of computer Science”, Alan Turing [8][9]. The method to decide whether a machine is capable of thinking like a human being is the Turing Test. The Turing test is a process between an individual that acts as an investigator, and two respondents, consisting of a human and a machine. The investigator asks multiple questions within a specific subject area, using a specified format and context. At the end, the investigator has to decide which of the respondents was human and which was a computer. After multiple repetitions, if the investigator was correct half the time or less, the computer is considered to have artificial intelligence [10].

Artificial intelligence combines computer science with datasets to create classification and prediction algorithms for problem solving, using the two sub-fields of machine learning (ML) and deep learning (DL) (Fig.2) [8][9].

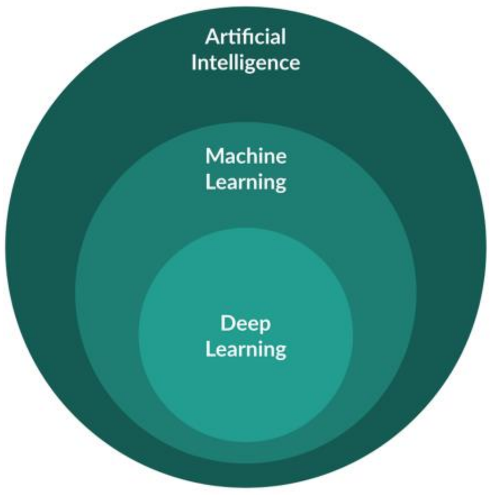


Figure 2:ML and DL subfields of AI.

2.1.1 Machine learning VS Deep learning

Machine learning (ML) works similarly to how a human brain has the ability to make decisions based on the knowledge and experience they gain after training. Machine learning models can be trained to identify patterns and relationships between data but always depend on human supervision to understand the hierarchy of features [8][11].

On the other hand, deep learning (DL) automates a big part of the feature extraction process without the supervision of a human expert. Moreover, deep learning models require more data to achieve greater accuracy but, contrary to machine learning, have the ability to do that (Fig.3) [12][8][13]. The word “deep” in deep learning refers to the depth of the neural network layers. If a neural network has multiple layers specifically more than 3 including input and output we can define it as a deep [12][13][9]. The first layers close to the input data on a deep neural network, extract mostly low-level features, such as edges and lines. However, in the case of image classification and as we approach the last layers, the features become more specific [13][9].

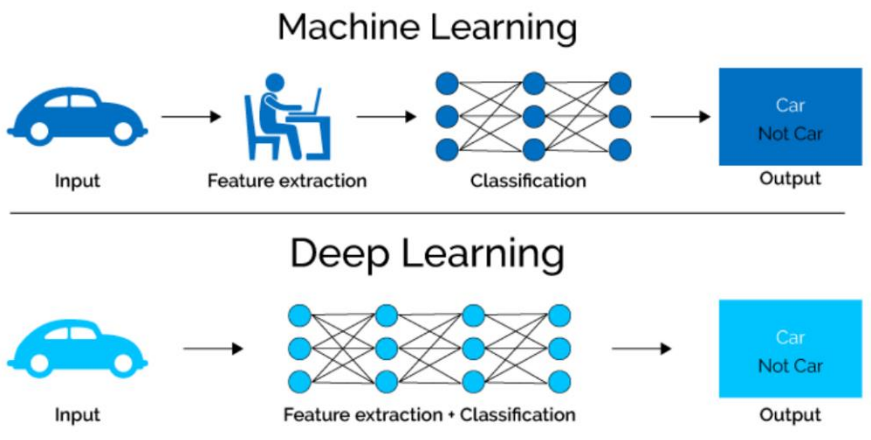


Figure 3: ML VS DL on image classification[14].

Deep learning can be performed with 3 main techniques which depend on the input data. The first technique is the supervised learning which requires all data to be paired with a label before the classification or the prediction [13][15][12]. The benefit of this technique is that, by using labeled data for training, the algorithm “learns” from the accuracy of the prediction, which helps to evaluate the model with greater accuracy. However, a disadvantage of supervised learning is that the decision boundary might be overstained when the training set doesn’t own samples that should be in a class. Moreover, they required human interferences for labeling at the beginning of the process [13][15].

The second technique is unsupervised learning, which contrary to the supervised, does not require labeled data and we mostly use it for data analysis and clustering. The advantage of not requiring labels is that it can discover the hidden patterns without the human interference. The main disadvantage of this technique is that you cannot get precise information regarding data sorting and that you can get less accuracy for the results because of the unknown data that will be labeled by the machine [13][15].

Finally, the third technique is a combination between supervised and unsupervised learning called semi-supervised learning. This technique uses both labeled and unlabeled datasets and is useful when you have a massive number of data and when it is difficult to extract relevant features from them [13][15].

2.2 Artificial Neural Networks (ANN)

The concept of Artificial Neural Network (ANN) is based on the biological neural network, not only in structure but also in the way they function. The main function of an ANN is to process information, more specifically, for computer science, it is useful for pattern recognition, data predictions and classification [16].

The ANN is divided into two categories, the first is the single-layer perceptron, consisting of one neuron, which receives the input data and produces the output immediately (Fig.4).

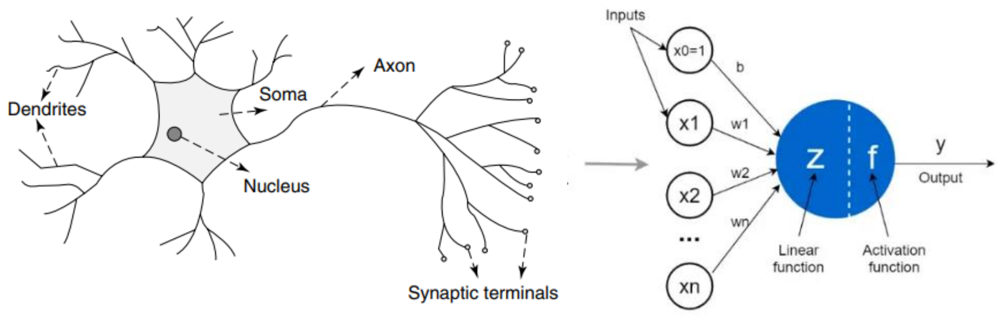


Figure 4: Biological Neural Network compared to single-layer ANN [17].

The other category is the multi-layer perceptron that, as we can see in Fig. 5, consists of multiple layers. It begins with the input-layer, followed by at least one hidden-layer and ends with the output-layer. Every layer contains nodes (neurons), known as processing units, connected by edges (like synapses) with weights that represent the importance of the connection [16].

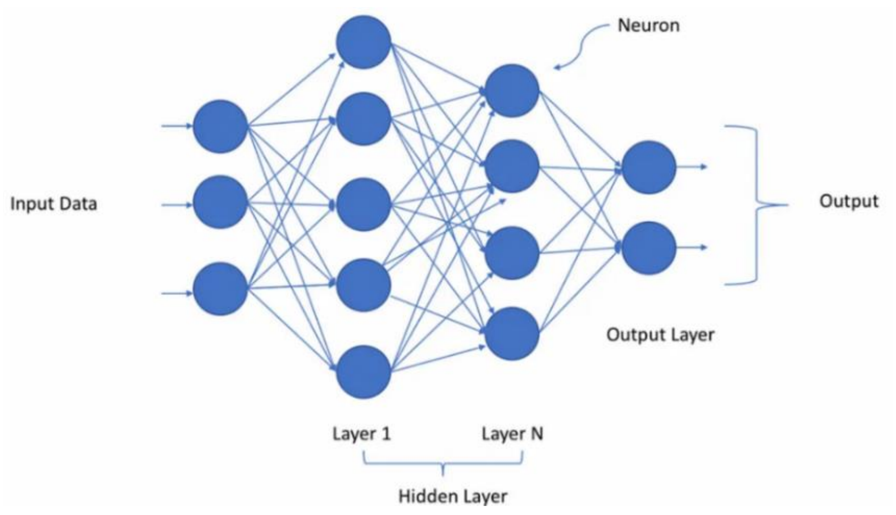


Figure 5: A multi-layer Deep Neural Network with N Hidden Layers [18].

As shown in the equation below, the neurons sum the incoming mathematically processed data multiplied by the weight and add the "bias":

$$z = \left(\sum_{i=0}^{n} x_i w_i \right) + b$$

The neurons then transfer their output to the next layer. Since the neurons on the input-layer have only one input, their output will only be the data multiplied by the weight.

A neural network is fully-connected when the output of each neuron is distributed to every neuron on the next layer, otherwise it is called partially-connected. Sometimes the system may contain a threshold restriction, which means that when the output of any neuron is below that threshold they will not transfer it to the next layer.

The learning of an ANN model is the process which consists of the repeated adjustment of the weight which in the first place was selected randomly. The network improves the accuracy by learning from the training sets until it can efficiently perform a specific task, such as image recognition, with minimum error [16].

2.2.1 Activation Functions (AF)

Activation Functions (AF) are functions used in artificial neural networks to transform the input of a layer into output which will be fed to the next layer [19]. The Activation functions, also often referred to as a transfer function, computes the weighted sum of the input and bias, which is used to decide if the neuron will be activated or not [20]. When using an activation function the output could be either linear or non-linear, depending on the function. Although a linear equation is simpler and easier to solve than a non-linear, they don't have the ability to learn and recognize complex mapping from data [19][20].

Some of the most common Activation Functions will be described below:

2.2.1.1 Sigmoid

The sigmoid function, also sometimes referred to as a logistic function, is a non-linear activation function used mostly in a feedforward neural network. The sigmoid function predicts a probability for the classification by transforming the values in the range of 0 to 1 [19][20].

It can be defined mathematically as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

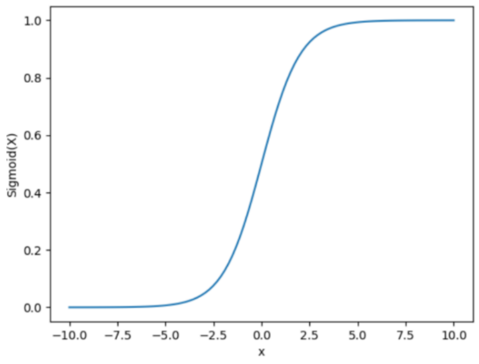


Figure 6: Sigmoid function plot [21].

2.2.1.2 Hyperbolic Tangent (tanh)

The Hyperbolic Tangent Functions (tanh) is similar to the sigmoid function but is preferred due to the fact that it gives better training performance for multi-layer neural network. The tanh function is mostly used in RNNs for natural language processing and speech recognition tasks. The output values of a tanh function lies in the range of -1 to 1 [19][20].

It can be defined mathematically as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

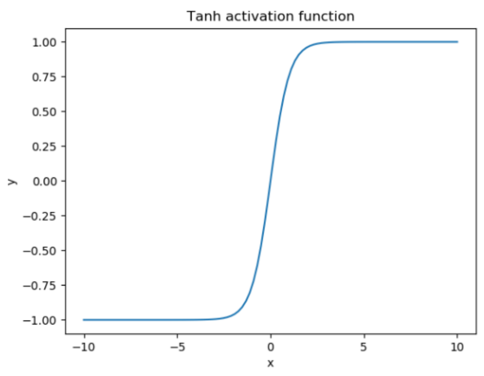


Figure 7: tanh function plot [21].

2.2.1.3 Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is a non-linear function which happens to be the most widely used activation function for deep learning models. The ReLU is a fast-learning activation function and, in comparison to sigmoid and tanh, performs and generalizes better. This function produces a positive output by forcing the output of the negative input to be 0 [19][20].

It can be defined mathematically as:

$$f(x) = \begin{cases} 0 & if\ x \leq 0 \\ x & if\ x > 0 \end{cases}$$

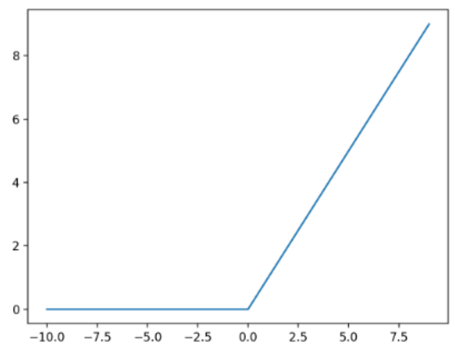


Figure 8: ReLU function plot [21].

2.2.1.4 Softmax

The softmax activation function computes probabilities for multi-class models. The output of a softmax function is a range of values between 0 and 1, with the sum of probabilities being equal to 1. The softmax function mostly appears in the output layer of a deep learning model, and the highest probability defines the class of the target [19][20].

It can be defined mathematically as:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

2.3 Recurrent Neural Networks (RNN)

The Recurrent neural networks' (RNN) learning method is similar to how the human brain learns from a sequence of patterns and that is why it is useful for tasks that contain sequence data, such as stock prediction or speech recognition [22][23]. The RNN model belongs to the category of feedforward neural networks, meaning that the information moves only in one direction, with the addition of edges that span adjacent time steps, introducing a notion of time to the model [24]. Moreover, the edges that connect the time steps are called recurrent edges and may include cycles, allowing the output from some neurons to affect the input of the same neurons [24][23].

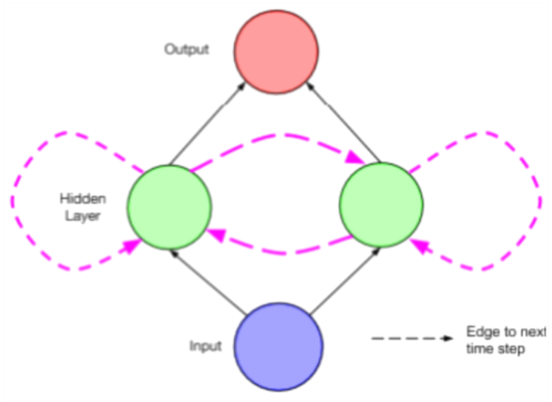


Figure 9: A simple RNN model [24].

According to [24] in a simple RNN model at each time step we follow the two equations below:

$$h^{(t)} = \sigma(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b_h)$$

$$y^{(t)} = softmax\left(W^{yh}h^{(t)} + b_y\right)$$

<u>Variables and Functions</u>

σ, SoftMax: Activation functions.

x: Input data vector.

h: Hidden layer vector is used as a memory to capture long term information from a sequence.

W^{hx}: Matrix of conventional weights between the input and the hidden layer.

W^{hh}: Matrix of recurrent weights between the hidden layer and itself.

b_y, b_h: Bias parameters which allow each node to learn an offset.

As we can see the output y(t) at the time t depends on the hidden state h(t). Moreover, the hidden state h(t) depends not only in the current data point x(t) but also in the previous hidden state h(t-1). Because of that relationship the input x(t-1) at time t-1 can influence the output y(t) at time t [24][23].

2.3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM), which was initially proposed by Hochreiter and Schmidhuber in 1997, is a variant of RNN that was introduced to efficiently solve the gradient vanishing problem [6][23][25]. Moreover, LSTM can learn how to bridge more than 1,000 time steps in contrast to RNN that can bridge only 5-10 [26]. LSTM extends an ordinary RNN model by introducing memory cells which replace the ordinary nodes in the hidden layers. Each memory cell contains a node with a self-connected recurrent edge of fixed weight, one to ensure that the gradient will not vanish after many time steps [24][23]. The memory cell has the same inputs and outputs as an ordinary RNN, with the addition of more gating units which control the information flow [23]. As we can see in Fig.10, the input and output gate are in charge of controlling the information that enters and exits from the memory unit [23]. In addition, the memory cell itself can be controlled by a forget gate which has the ability to reset the memory unit with a sigmoid function and can also decide which part of the previous state's information will be necessary to keep [24][23].

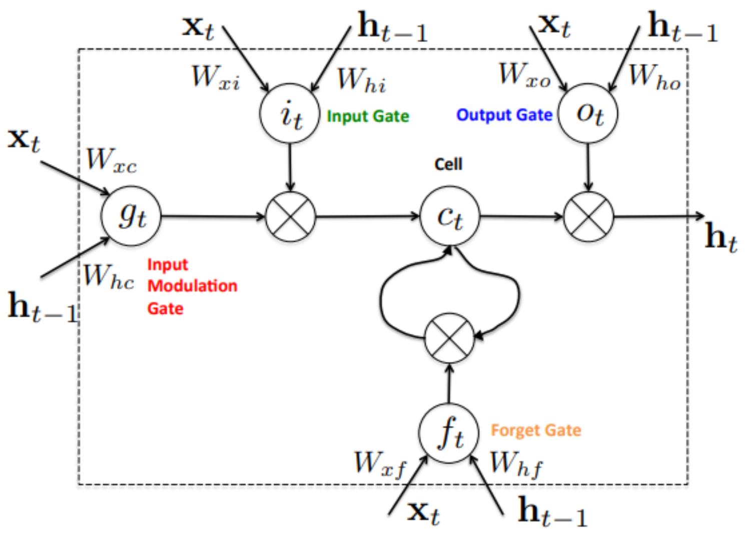


Figure 10: Unit structure of LSTM [23].

According to [23] the gate definitions and the LSTM algorithm are given as follows:

$$f_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f)$$

$$i_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i)$$

$g_t = tanh(W_{xc}X_t + W_{hc}h_{t-1} + b_c)$

$c_t = f_t \odot c_{t-1} + i_t \odot g_t$

$o_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o)$

$h_t = o_t \odot tanh(c_t),\ z_t = softmax(W_{hz}h_t + b_z)$

Where f_t represent the forget gate, i_t input gate, g_t input modulation gate, c_t memory cells, o_t output gate and ⊙ is an element-wise multiplication [23].

2.3.2 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) was first proposed and tested by Cho et al. in 2014 on a statistical machine translation problem. Similarly to LSTM, GRU has gating units which are responsible for controlling the information flow inside the unit and adaptively capture the dependencies on different time scales. The GRU model depends mainly on two different gates. The first is the update gate, which combines the forget and the input gate from the LSTM model and decides how much the unit should update the current content. The second is the reset gate, which when is off (the value close to 0) can forgot the previous state and reset the memory of the cell, making the unit act as if the next input was the first in the sequence (Fig.11) [25][27].

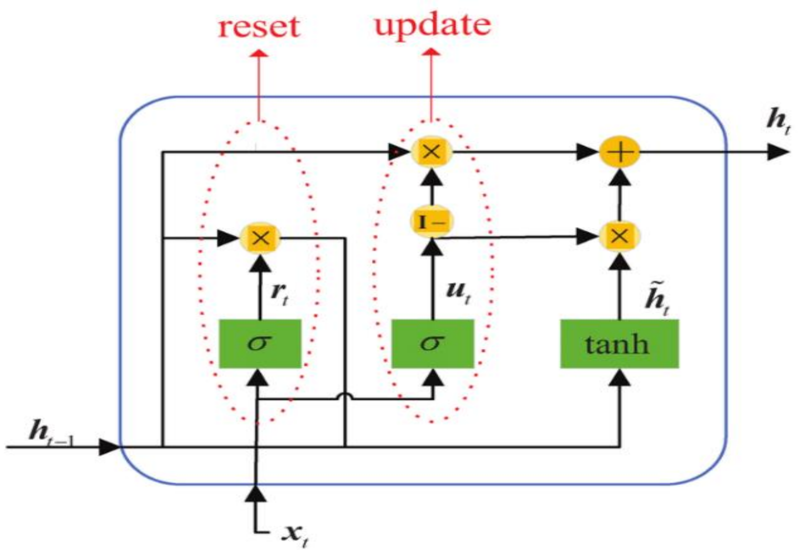


Figure 11: Structure of GRU [28].

According to [27] the state equations of the GRU are the following:

$r_t = \sigma(W_r h_{(t-1)} + R_r x_t + b_r)$

$h'_t = h_{(t-1)} \odot r_t$

$z_t = g(W_z h'_{t-1} + R_z x_t + b_z)$

$$u_t = \sigma(W_u h_{t-1} + R_u x_t + b_u)$$

$$h_t = (1 - u_t) \odot h_{(t-1)} + u_t \odot z_t$$

Where the equations r_t represent reset gate, h'_t current state, z_t candidates state, u_t update gate and h_t new state. Moreover, the equations depend on the rectangular matrices W_r, W_z, W_u, the square matrices R_r, R_z, R_u, and the bias vectors b_r, b_z, b_u.

2.4 Convolutional Neural Networks (CNN)

Convolutional neural network (CNN) is deep learning network architecture with neurons that self-optimize through learning directly from the data [22][29]. CNN its very popular due its efficiency to extract spatial features and recognize pattern without human supervision, especially in cases of image classification [6][29]. CNN architecture consists mainly of three types of layers, convolutional, pooling and fully-connected [22].

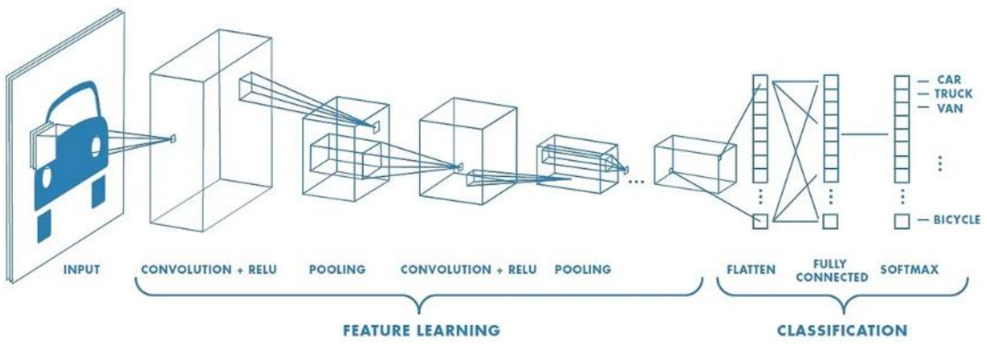


Figure 12: CNN architecture [30].

2.4.1 Convolutional Layer

As the name indicates, Convolutional layer is a major part of CNN architecture. A convolutional layer contains multiple learnable kernels (filters) that are usually small, a choice that is commonly made is to keep the size of kernels 3x3 or 5x5. The main process of a Convolutional layer is convolving each kernel across the input data [22]. As we see in the Fig. 13 the process for each kernel begins by calculating the dot product from the top of the input image, then the kernel moves down to get the next convolution result until we find the bottom border. Afterwards, the kernel returns to the top and moves to the right. The process is repeated until the kernel reaches the bottom right corner of the input image [31]. By calculating the dot product, we get a single number for each part of the input image.

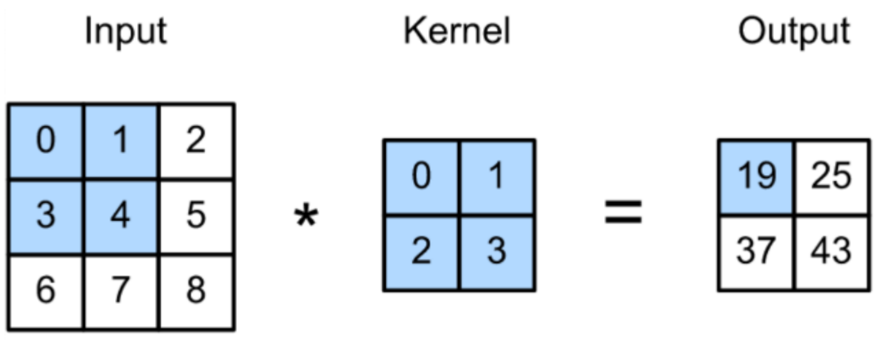


Figure 13: Illustration of the convolution operation [32].

After every calculation we have as a result a smaller image for each filter that will be a part of the output feature map. The feature map can be visualized, as seen in Fig.14 [22].

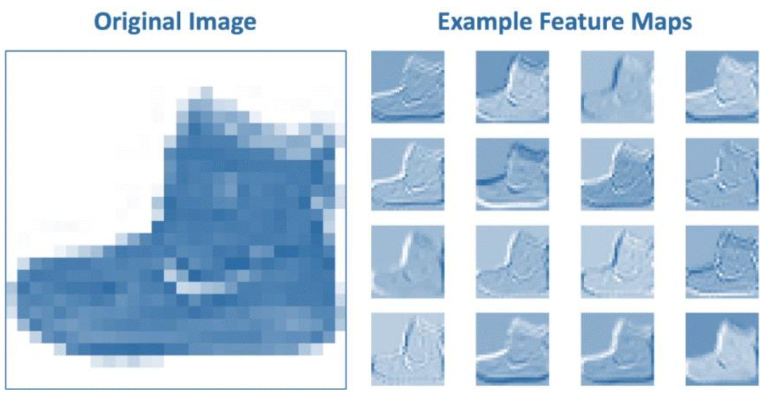


Figure 14: Example of feature map [33].

One of the challenges of the process of convolution is to reduce the complexity and optimize the output [22]. We can achieve that by adjusting the hyperparameter stride. The stride determines the number of rows or columns the kernels move each time and its purpose is to reduce the parameters (Fig.15) [34]. Additionally, we can use the method zero-padding. Zero-padding simply adds an extra border with zeroes and is used mostly to prevent information loss. It is also useful if we want to control the dimension of the output (Fig.16) [34][22].

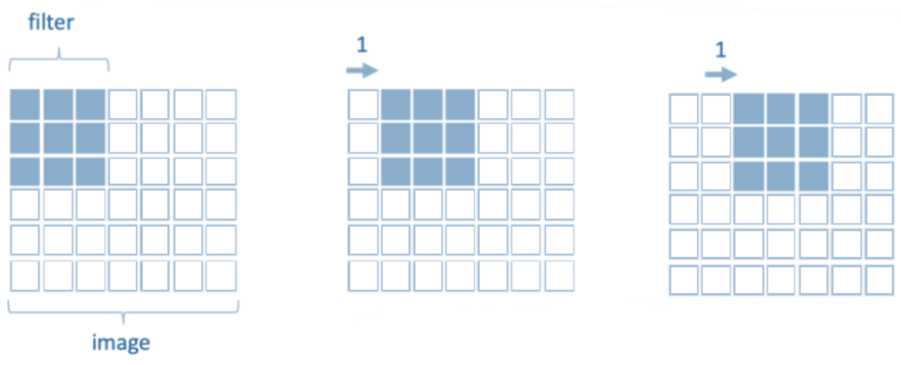
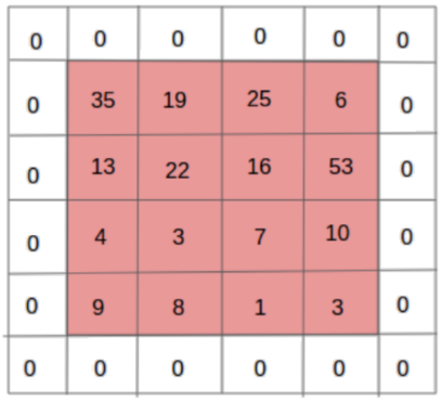


Figure 15: Stride 1, the filter window moves only one time for each connection.



0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figure 16: A zero-padded 4x4 matrix becomes a 6x6 matrix [35].

2.4.2 Pooling Layer

The pooling layers are used to reduce the number of parameters and the computational complexity by down sampling the spatial size of the input data [6][22]. Moreover, it helps to prevent overfitting by keeping only the most important features [6]. In the case of image classification down sampling the data is a process similar to reducing the resolution of an image [34]. The pooling layer divides the image into sub-region rectangles and returns a single number for each region according to the type of the layer. The most common types of pooling layers are max and min, which return the maximum and minimum value respectively, as well as average, that returns the mean value (Fig. 17) [6][34].

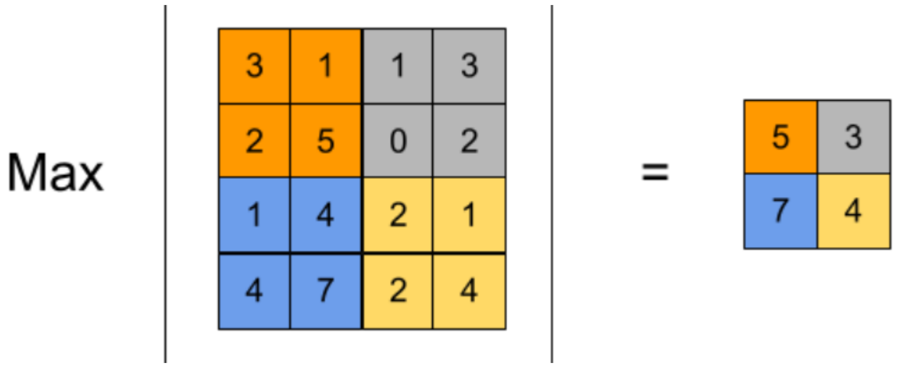


Figure 17: Max Pooling example [36].

2.4.3 Fully-Connected Layer

The fully connected layers are usually placed after the convolutional and pooling layers [31]. A layer is called fully connected when it is connected to every node, both in the previous and following layers. Overall, they are mostly useful if we want to build features with a strong capability for the next layer [31][34].

2.5 Transfer Learning

Transfer learning is a machine learning method inspired by the human ability to complete a task using previous experience you have gained by solving a different but related problem. Similarly, in machine learning we can transfer the knowledge of a pre-trained model in order to improve the learning ability of a new model with any relative problem. With transfer learning we can reduce the time complexity by using already existing knowledge for low-level features, instead of developing a new model from scratch [37][29].

Some of the most common pretrained models will be described below:

2.5.1 AlexNet

The pre-trained model AlexNet classifies high resolution images with better accuracy than the previous state of art. In ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) – 2010 achieved top1 and top5 error rates of 37.5% and 17% respectively. The input of AlexNet network is RGB images of size 227 x 227 x 3. As we can see in Fig. 17, the architecture of AlexNet consists of 5 convolutional layers and 3 fully connected (dense) layers, with the ReLU as the activation function and a softmax classification layer to classify the output [7].

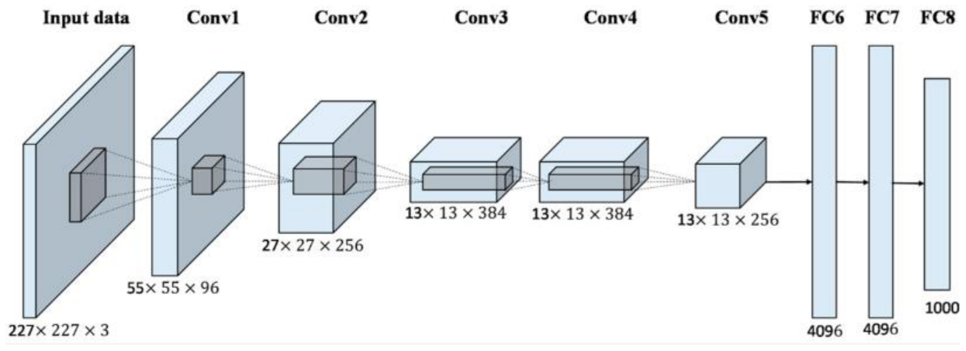


Figure 18: AlexNet architecture [38].

2.5.2 VGG-16

The VGG-16 is one of the most popular and efficient pre-train models for image classification. Introduced during the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) – 2014 contest, it had the highest performance with top1 and top 5 error rates 24.8% and 7.5% respectively. The input of VGG-16 network is RGB images of size 224x224x3. As we can see in Fig. 18, the architecture of VGG-16 consists of 13 convolutional layers and 3 fully connected (dense) layers with the ReLU as an activation function and a softmax classification layer to classify the output [7][29]. The main contribution of VGG-16 is that it increases the depth of CNN by using a filter of size 3x3 [7].

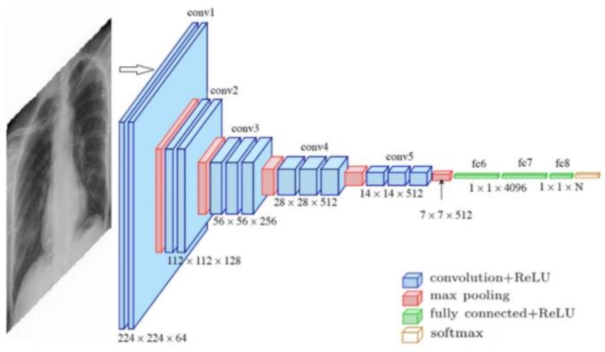


Figure 19: VGG-16 architecture [39].

2.5.3 Residual Networks (RestNet50)

RestNet50 is a convolutional neural network based on a pre-trained model that was proposed, which achieved first place in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) – 2015 contest. The input of RestNet50 network is RGB images of size 224 x 224 x 3 [29]. The RestNet50 consists of 48 convolutional layers, 1 max pooling layer and 1 average polling layer [40]. This type of pre-train models can reduce the vanishing gradient problem by adding shortcut connections that allow the gradient to flow through [41].

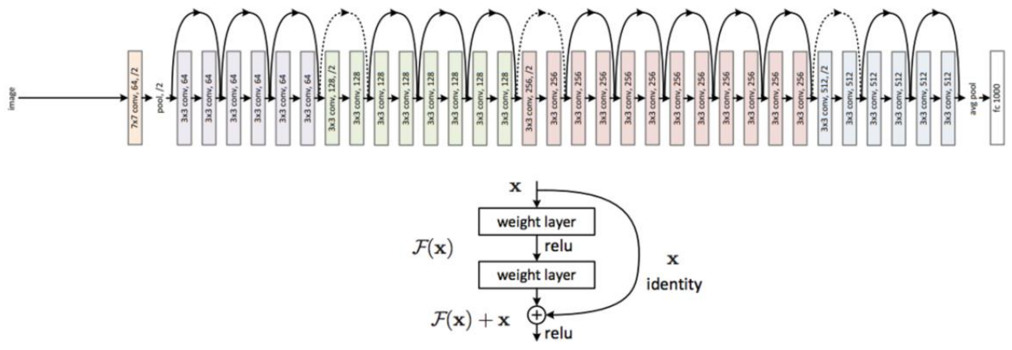


Figure 20: Residual Networks architecture [42].

2.5.4 Inception V3

The inception V3 is a convolutional neural network developed as a GoogleNet module for image processing and object detection [29]. The network uses multiple convolutional layers for feature extraction and, unlike other networks such as AlexNet, allows use of different kernel size in the process of the convolution. The inception architecture is intended to allow deeper networks while also decreasing the number of parameters [37][29].

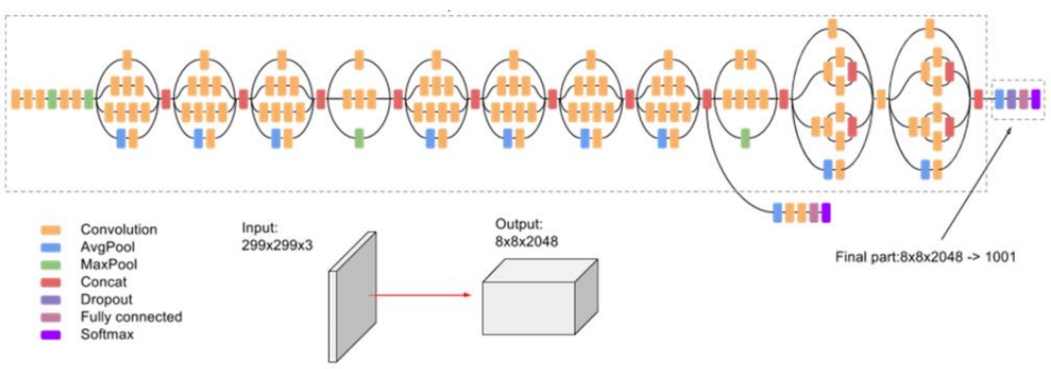


Figure 21: Inception V3 architecture [43].

3. RELATED WORK

As it has been previously mentioned, Sign language recognition is divided mainly into two categories which depend on the way we collect data that can be processed and classified afterwards. The first category is sensor-based, we use hand worn equipment such as colored-gloves or special-gloves to capture the key-points of hand and sensor-based devices to capture motion. This approach has better robustness as well as accuracy. However, it is not a realistic technique for real time problems because it requires special equipment. The second category is vision-based, that requires only a normal camera to collect data. During this approach we face many challenges whilst trying to achieve great accuracy, but it is more real time friendly.

This thesis will be focused on the vision-based approach using deep learning methods, as well as review various state-of-the-art sign language recognition papers with different approaches and databases.

3.1 RNN approaches

Due to the efficiency of RNN models in sequence data, many researchers have found them very useful in video-based data recognition. Jae Gi Son et al. [44] first proposed a new Korean sign language video-based dataset of 100 sentences, performed by 10 professional Korean signers. The authors developed a sign language recognition system based on the stacked bidirectional GRU, using human key-points extracted by the open source OpenPose, achieving classification accuracy of 89.5%. Moreover, according to Tao Liu et al. [45] the traditional methods for SLR that use handcrafted features and Hidden Markov Models (HMMs), are problematic because the handcrafted features are difficult to design and aren't able to adapt to the large variation of sign words. To solve this problem the authors proposed an end-to-end SLR model based on LSTM using skeleton joint trajectories. For the evaluation process the authors created two Chinese sign language datasets consisting of 100 and 500 words. The data was recorded by Kinect 2.0 which can capture color images, data map, and skeleton joint location in real time, however their focus lay on the trajectory of four skeleton joints. The proposed model achieved an accuracy of 86.2% and 63.3% for the small and large dataset respectively and outperformed other methods such as traditional HMM.

3.2 CNN approaches

Due to the efficiency of the feature extraction process and the existence of pre-train models which increase the accuracy and minimize the complexity, the CNN models are very popular among researchers who work in the sign language recognition field. Abul

Abbas Barbhuiya et al. [7] approached the hand gesture recognition problem by using modified pre-trained CNN models for static sign image classification. During this approach, modified pre-trained models AlexNet and VGG16 are used for the feature extraction processed, followed by multiclass support vector machine (SVM) classifier. The author decides to investigate the generalization potential of the models by using two different approaches regarding the data division. During the first approach, 70% of the random sign images from the dataset are used for training, and the remaining 30% are used for testing. The author managed to achieve 99.82% and 99.76% accuracy for the proposed modified models AlexNet and VGG16 respectively, using the cross-validation method. In the second approach, the author decides to use data from 4 distinct people for training, and data from 1 different person for testing, managing to achieve 70% accuracy in both modified pre-trained models. The author is led to the conclusion that the recognition performance of CNN architectures degrades when characters have very high interclass similarities. In addition, Celal Can et al. [29] aims to design a new deep learning model based on CNN to recognized hand gestures and improve recognition rate, training, and test time of the model. In this study, the author uses two different datasets with near-infrared and colored natural images, with the addition of data augmentation to boost the training ability of models. What's more, as the author became interested in transfer learning, they decided to compare the results of five popular pre-trained models with the proposed CNN model. The proposed CNN model, VGG16, VGG19, Resnet50, DenseNet121, and InceptionV3 achieved a recognition rate of 99.98%, 100%, 99.99%, 91.63%, 82.42% and 81.84% respectively, on near-infrared images. Regarding the colored natural images, the models achieve accuracy of 99.91%, 99.31%, 98.67%, 91.97%, 93.37%, and 93.21%, respectively. The pre-trained models achieved a slightly better recognition rate than the proposed CNN model. Nonetheless, the proposed model achieves promising results, taking the least amount of time, which makes it more suitable for real time application.

3.3 Combination of RNN and CNN approaches

Coupling CNN with RNN models can keep information over time, especially in the video-based data. The combination of the CNN's ability to recognize patterns and the efficiency of RRN models to deal with sequence data make the researchers to expect promising results in the field of dynamic SLR. Mostafa Magdy Balah [6] has created a video-based Arabic sing language dataset, containing 20 signs, performed by 72 signers, and proposed a deep learning architecture by combining CNN and RNN models. The authors have divided the data pre-processing into 3 stages. In the first stage they reduced the dimensions of each frame in order to achieve less overall complexity. In the second, they passed the output to a function that subtracts every two consecutive frames to find the motion between them. Finally, in the third stage, they unify each class's features and have 30 frames as output, where each unified frame combines (3x3) frames. The purpose of stage 3 is to reduce redundancy without losing any information. The main idea of the proposed architecture is to train two different CNN independently for feature extraction, then concatenate the output into one single vector and pass the vector to an RNN for classification. The proposed model achieved 98% and 92% on the validation and testing subsets respectively on the suggested dataset. Moreover, they achieved promising accuracies of 93.40% and 98.80% on top-1 and top-5 respectively on the UFC-101 dataset. Another approach for SLR is not only to transform image into text but also to take it a step further by converting text into speech, an aspect especially pertinent for the modern world as we can give hearing-impaired people the opportunity to join a video conference in business or educational platforms. Babita Sonare et al. [1] proposed a combination architecture of CNN for feature extraction and LSTM (RNN based model) for classification and then converted the text output into speech using the open-source Text-To-Speech API in python. The authors manage to achieve 90.1% accuracy using an Arabic sign language recognition dataset which consists of the letters A-Z.

3.4 Other approaches

I will now review some other state-of-the-art publications which are mainly focused on feature extraction and use a variety of different and interesting processes. Razieh Rastgoo et al. [5] proposed a simple yet efficient and accurate model using deep learning approaches such as Single Shot Detector (SSD), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM), benefiting from the spatio-temporal hand-based information from videos. This study includes two main steps. The first step is the extraction of three types of spatial features, including hand features, Extra Spatial Hand Relation (ESHR) features, and Hand Pose (HP) features. Initially, the authors train the SSD model for hand detection using the proposed dataset consisting of 100 Persian words. They then used the pre-trained ResNet50 model for hand feature extraction. Next, for the ESHR feature they captured two types of spatial features, the Slope feature which is the interaction of hands, and the Orientation feature which is the directions of hands. Finally, for the HP features they used an already existing CNN-based hand segmentation and localization model to predict the 3D hand key-points from 2D images. The second step is the sign recognition process for which they fused all the features from the previous step and fed them to the LSTM for temporal feature extraction. Additionally, they did a complete analysis using different pre-trained models, different spatial features, and different temporal-based models for sequence learning and outperformed state-of-the-art results on isoGD dataset. Due to the lack of video-based datasets for dynamic sign language, Gerges H. Samaan et al. [4] decided to create the DSL10-Dataset, which consists of ten vocabularies that were performed by five signers. During the course of their research the authors decided to use MediaPipe to determine the location, shape, and orientation of the hands, body, and face by extracting key-points. Additionally, they performed two experiments on the proposed dataset, with and without the use of facial key-points using RNN models such as GRU, LSTM, and Bi-directional LSTM to address the issue of frame dependency in sign movement. The MediaPipe calculate the key-points in three dimensions and extract 126 key-points for each hand, 132 for the pose estimation and 1404 for the face. The total key-points in the first experiment with the face were 1662 and managed to achieve accuracy of 100%, 99.6% and 99% on GRU, LSTM, and BILST respectively. In the second experiment the total key-points without the face were 258 and they manage to achieve accuracy of 100%, 99.6% and 99.3% respectively. On low complexity sequences, GRUs outperform LSTM networks, while on high complexity sequences, LSTMs outperform GRUs. LSTM and BILSTM require more parameters and nodes than GRU. The authors came to the conclusion that the accuracies with and without the face key-points are very close, however in the case where they include the face the number of key-points proved six times larger, which had negative effect on the prediction and learning time.

The biggest issue in the field of sign language recognition is the lack of large datasets. From the previous cutting edge studies we can deduce that most datasets don’t contain a satisfactory number of sign words and cannot be applied in practice for real time application. Dongxu Li et al. [3] introduced a unique large-scale Word-Level American Sign Language (WLASL) video-based dataset to overcome the issue. The WLASL dataset contains more than 2000 words performed by 119 signers in a total of 21,083 videos. Moreover, each video consists of only one sign and each sign is performed by at least 3 different signers, which facilitates the generalization ability of the models. The authors create the proposed dataset using data from two main sources, multiple educational sign language websites, such as ASLU and ASL-LEX and ASL tutorial videos on YouTube. After they collected all the data, they remove the signs with few samples which probably implies that those words are not frequently used. In addition, they provided some meta information for each video of the proposed dataset, such as temporal boundary, body bounding box, signer annotation and sign dialect/variation annotations. Additionally, they experimented with several deep learning methods, based on holistic visual appearance, and 2D human pose, using the proposed dataset they achieved comparable classification performance of 62.63%. The authors concluded that developing word-level sign language recognition algorithms on such a large-scale dataset requires more advanced learning algorithms.

3.5 Challenges on SLR

Sign language recognition can be really challenging if we want to use it in daily life, making our task far more challenging. Some of these difficulties can be summarized as follows [7][5][6][4]:

- Complicated Gestures: Gestures that can be complicated for the signers, especially if they are not commonly used, or signs that may have very slight differences.
- Viewpoint Diversity: The same sign poses and hand kinematics can differ between different people.
- Accessories intervention: In cases where a sign includes facial expression an individual wearing glasses or earrings may confuse the recognition.
- Datasets: The inexistence of large and complete datasets with different signs and the inability of the models to adequately process them.
- Background complexity: High-level noises such as lightning or other elements that exist in the background which may interfere with the recognition process.
- Overfitting: The model giving an accurate prediction for the training data but not for new data, compelling us to improve the generalization ability of the model to adapt properly in new data of new signers.

4. METHODOLOGY

The use of AI and deep learning in the field of sign language recognition has the potential to revolutionize the communication with hearing impaired individuals. Using deep learning algorithms, AI can accurately recognize complex movements of sign language. As such, this thesis focuses on implementing a deep learning model, capable of recognizing sign language video-based data, using two different approaches for feature extraction. The first approach uses the pre-trained model VGG-16 to extract spatial features. On the other hand, the second approach uses the MediaPipe framework to extract features based on key-points from the pose and hands of each signer. Finally, a classification procedure was follow where the output feature of each approach was fed into the LSTM and GRU models which fall under the category of RNN models (Fig.22).

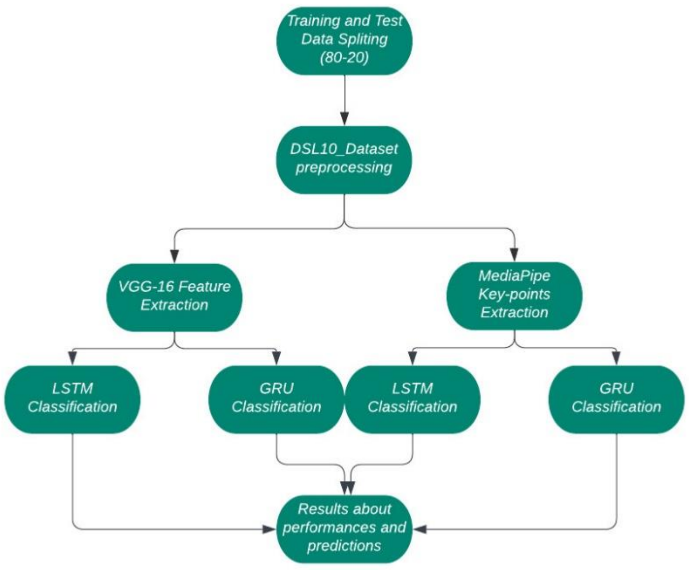

Training and Test Data Spliting (80-20)
DSL10_Dataset preprocessing
VGG-16 Feature Extraction
MediaPipe Key-points Extraction
LSTM Classification
GRU Classification
LSTM Classification
GRU Classification
Results about performances and predictions

Figure 22: Thesis Diagram.

4.1 Database

The data used for this thesis were claimed from the DSL10-Dataset which was proposed by Gerges H. Samaan et al. [4]. The DSL10-Dataset contains 10 commonly used vocabularies that were repeated 75 times by 5 different signers. All videos were recorded in indoor environments with regular lighting using an average mobile camera.

4.1.1 Data preprocessing

Data pre-processing is essential in order to make model processing more efficient using the least number of resources. Initially the frame’s dimensions were reduced to obtain the same size 224x224x3 which is the required input size for the pretrained model

VGG-16. In addition, for the uniformity of the data the color space for each frame was converted to the most commonly used RGB (Red Green Blue).

Following, the Structural Similarity Index (SSIM) was calculated in order to decide which consecutive frames are more similar compared to others. This allows us to reduce the number of frames to only 5 frames per video which means that only the frames with a variety of information were kept. The structure similarity function takes two images as input and returns a value between -1 and 1. If the value was significantly close to 1, then it means that the second frame was similar to the previous frame and thus excluded from the video. Following that, If the number of the remaining frames was greater than the desired number, then the extra frames were deleted based on a frame skip number. On the other hand, if the number of the remaining frames was less than the desired number, then the last frame was duplicated until the frames reaches the maximum number. Finally, the data were divided randomly into 80% training set and 20% test set. The benefits gained from this process were the reduction of processing time and achievement of less complexity overall.

For instance, as we can observe in Fig.23 the first row shows the first five consecutive frames of a video and the second row the first five consecutive frames after the preprocessing.

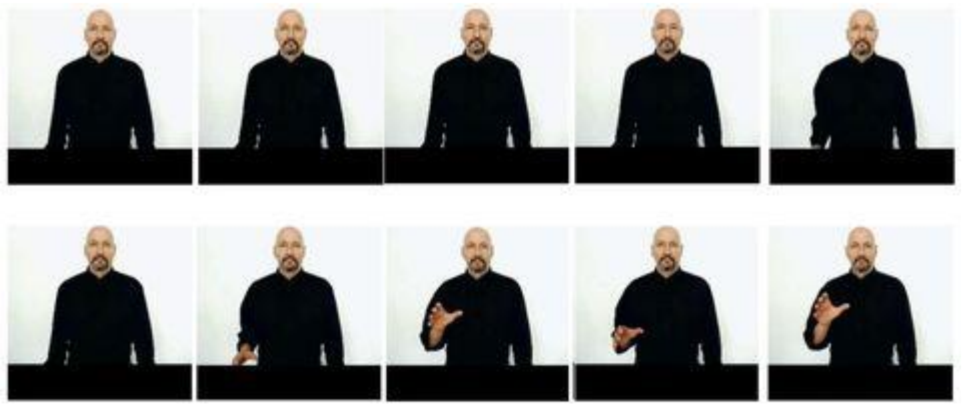


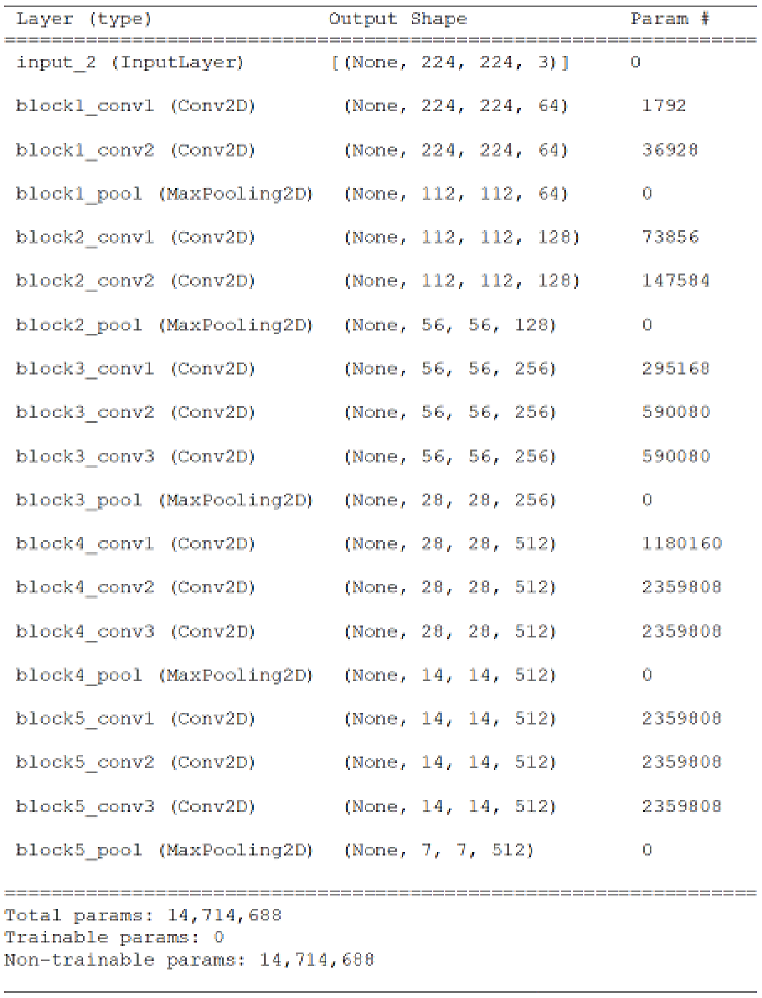
Figure 23: The first five consecutive frames before and after the video preprocessing.

4.2 Feature Extraction

The following section describes two approaches concerning the feature extraction process of this thesis. The first approach made use of the pre-trained model VGG_16 in order to extract spatial information from each frame. In the second approach the MediaPipe Framework was used in order to extract the key-points of hands and pose in each video.

4.2.1 VGG-16

In this thesis the VGG-16 convolutional neural network was selected based on the performance in the state-of-the-art papers. VGG-16 is a well-established and widely used pre-trained model with simple structure and small number of layers compared to other more complex models. The architecture of the model consists of 13 convolutional layers with 5 max pooling layers, followed by 3 fully connected layers. As we can observe in Fig. 24, for the purpose of this thesis the fully connected layers at the top of the model were excluded, in contrast with the first group of layers which extract low-level features. Moreover, since training a new network from scratch takes a long time and is computationally expensive, all the layers were set in the not trainable mode in order to keep the pretrained weights from ImageNet. Finally, the input data of VGG-16 in this thesis were video frames with a size of 224 x 224 x 3 resulting in the output size of 7 × 7 × 512 which may be interpreted as 7 × 7 × 512 = 25,088 features.All the data were fed to the VGG-16 model until the end point was reached creating the final dataset consisting of 25,088 features for each frame.



```
Layer (type)                Output Shape              Param #
=================================================================
input_2 (InputLayer)        [(None, 224, 224, 3)]     0

block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792

block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928

block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0

block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856

block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584

block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0

block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168

block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080

block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0

block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0

block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

Figure 24: VGG-16 Architecture.

4.2.2 MediaPipe

MediaPipe is an open-source framework developed by Google which provides a large number of models for body detection and tracking which are trained on an extensive and diverse dataset. Some of the biggest challenges in sign language recognition is to determine the location, shape and direction of hands, especially in the case of dynamic signs where it is essential to capture the relations between movements.

In this thesis the pose estimation techniques of MediaPipe were used in order to capture the location of the hands regarding the body. MediaPipe extracts the key-points for the three dimensions which determine the hand and pose estimation (Fig.25).

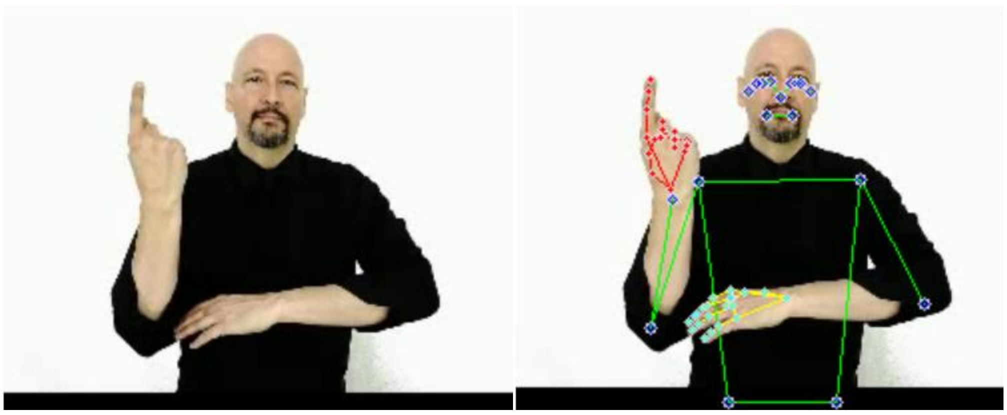


Figure 25: Key-points Layout for pose and hands .

As we can see in Fig.26 for hand estimation the MediaPipe framework extracts 21 key-points. The key-points are calculated in the three-dimensional space, resulting in the output of 126 key-points for both hands. The number for the hand key-points could be calculated as follows:

$$Hand\ key - points \times Three\ dimensions \times Number\ of\ hands = 12 \times 3 \times 2 = 126$$

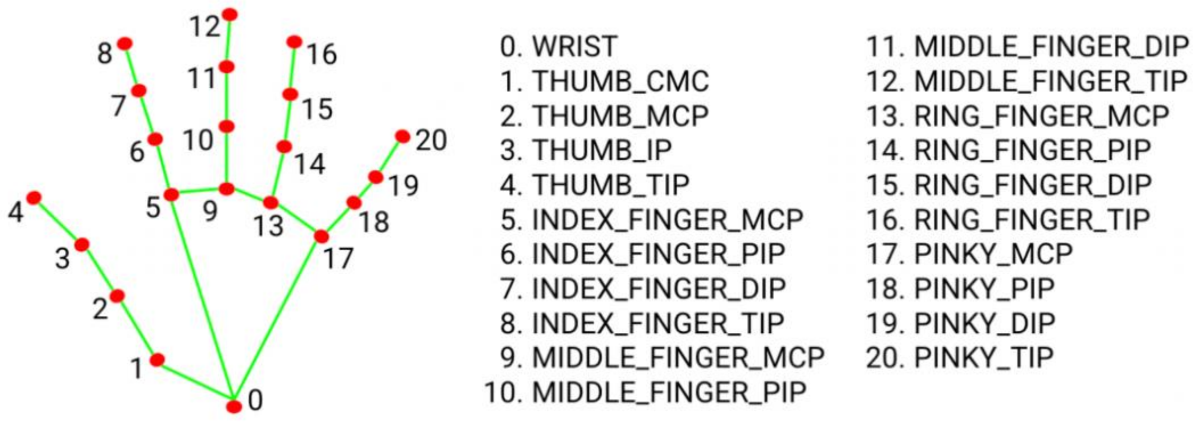
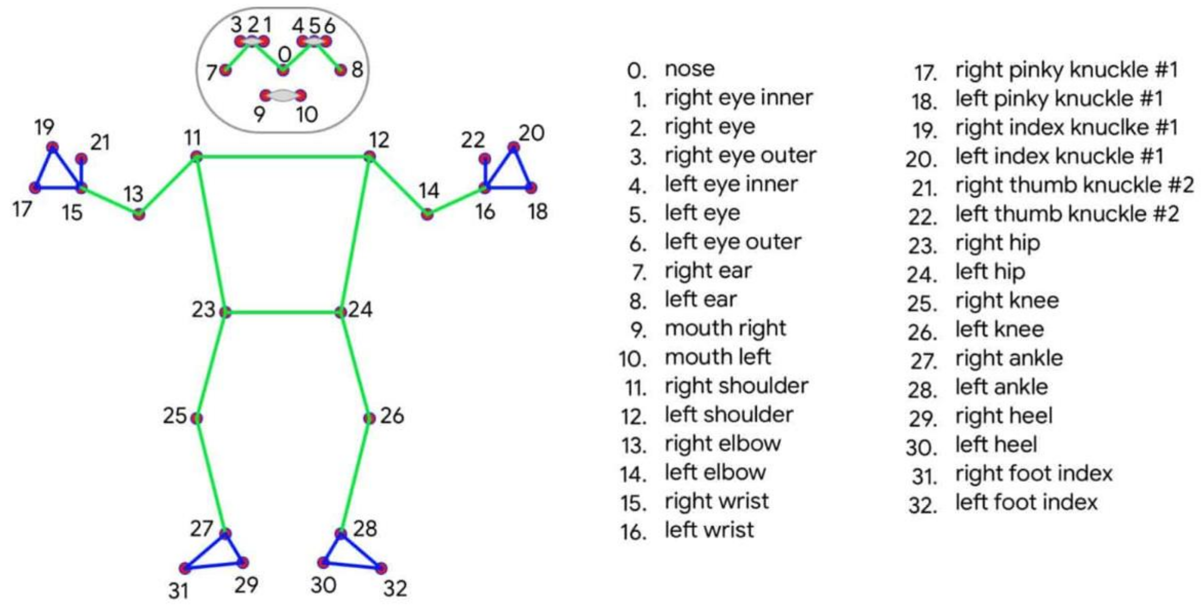


Figure 26: Key-points for Hands [46].

As we can see in Fig.27 for pose estimation the MediaPipe framework extracts 33 key-points. The key-points are calculated in the three-dimensional space in addition to the visibility, resulting in the output of 132 key-points. Visibility is the value which declares if the point was visible or hidden, for instance behind other body parts. The number for the pose key-points could be calculated as follows:

$$Pose\ key - points\ \times\ (Three\ dimensions\ + Visibility)\ =\ 33\ \times\ (3\ +1)\ =\ 132$$



0. nose
1. right eye inner
2. right eye
3. right eye outer
4. left eye inner
5. left eye
6. left eye outer
7. right ear
8. left ear
9. mouth right
10. mouth left
11. right shoulder
12. left shoulder
13. right elbow
14. left elbow
15. right wrist
16. left wrist
17. right pinky knuckle #1
18. left pinky knuckle #1
19. right index knuclke #1
20. left index knuckle #1
21. right thumb knuckle #2
22. left thumb knuckle #2
23. right hip
24. left hip
25. right knee
26. left knee
27. right ankle
28. left ankle
29. right heel
30. left heel
31. right foot index
32. left foot index

Figure 27: Key-points for pose [47].

In total the key-points extracted for each frame were 258.

$$Hand\ key - points\ +\ Pose\ key - points\ =\ 12\ 6\ +\ 132\ =\ 258$$

In addition, another aspect which could have been taken into consideration would be the use of the face key-points. As we can see in Fig.28 for face estimation the MediaPipe extracts 468 key-points. The key-points are calculated in the three-dimensional space, resulting in the output of 1404 key-points. The number for the hand key-points could be calculated as follows:

$$Face\ key - points\ \times\ Three\ dimensions\ =\ (468\ \times\ 3)\ =\ 1404$$

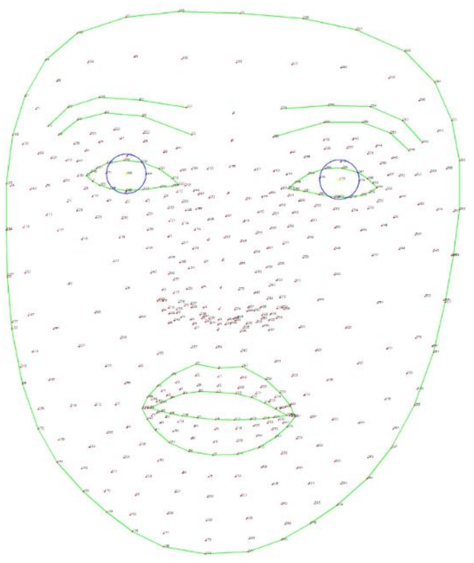


Figure 28: Key points for Face [48]**.**

In total the key-points extracted for each frame were 1662.

$$Hand\ key-points\ +\ Pose\ key-points\ +\ Face\ Key-points\ =\ 126\ +\ 132\ +\ 1404\ =\ 258$$

However, due to the fact that the inclusion of facial key-points increases the size of features by 6 times and thus negatively affecting the time complexity of the model, it was decided not to incorporate the facial key-points in this research. According to Gerges H. Samaan et al. [4] the train and test accuracy of the inclusion and exclusion of the facial key-points was very close. Nevertheless, when the facial key-points were included, the MediaPipe algorithm took additional time to extract and process 1404 points of the face. Furthermore, using facial-key-points will force hearing-impaired people to use facial expressions for each vocabulary which means that using facial key-points is not suitable for real time sign language recognition.

4.3 Classification

The section below will describe the process that follows feature extraction. This process involves the adjustments of hyperparameters and the analysis of the proposed classification models.

4.3.1 Hyperparameter tuning

Hyperparameters are parameters that are set from the developer before the training of a model, and they cannot be changed before the end of the learning process. They determine the characteristics of the learning algorithm used to train the model. The combinations of hyperparameters are countless and vary among different learning algorithms. The process of hyperparameter tuning is important in order to ensure that the model will achieve great performance. This process consists of multiple training and testing experiments, applying different combinations of hyperparameters until the best combination is found.

Some of the hyperparameters used in this thesis will be described below:

4.3.1.1 Number of Hidden layers

The number of hidden layers determines the depth of the neural network. Every new hidden layer can capture more complex features, however, it may easily lead to overfitting if not regularized properly.

4.3.1.2 Batch size

Batch size is the number of samples of data used by the model for each iteration in the training process. The batch size value varies between 1 and the total number of the samples.

Using a large batch size value implies that the model will process more samples in each iteration resulting in a decrease of the training time. However, this comes with the cost of slower convergence to the optimal solution. Moreover, a large batch size consumes more memory which could be problematic when resources are limited. In contrast, a small batch size requires less memory, and also converges faster to the optimal solution. Nevertheless, it is much more computationally expensive and time consuming.

Some factors which affect the choice of the batch size are the available computational resources, the size of the dataset, and the complexity of the model.

4.3.1.3 Epochs

Epochs refers to the number of times that the dataset will complete a cycle through the model during the learning process. Insufficient epochs could easily lead in underfitting, whereas excessive epochs could easily lead in overfitting, therefore it is advised to stop the training when the changes between each epoch are not notable.

4.3.1.4 Number of Unites

The number of units in each layer represents the ability of the model to recognize complex patterns. However, adding a large number of units increases the time complexity of the model dramatically. In addition, the increase of units may lead in overfitting where the model cannot recognize new data effectively.

4.3.1.5 Dropout

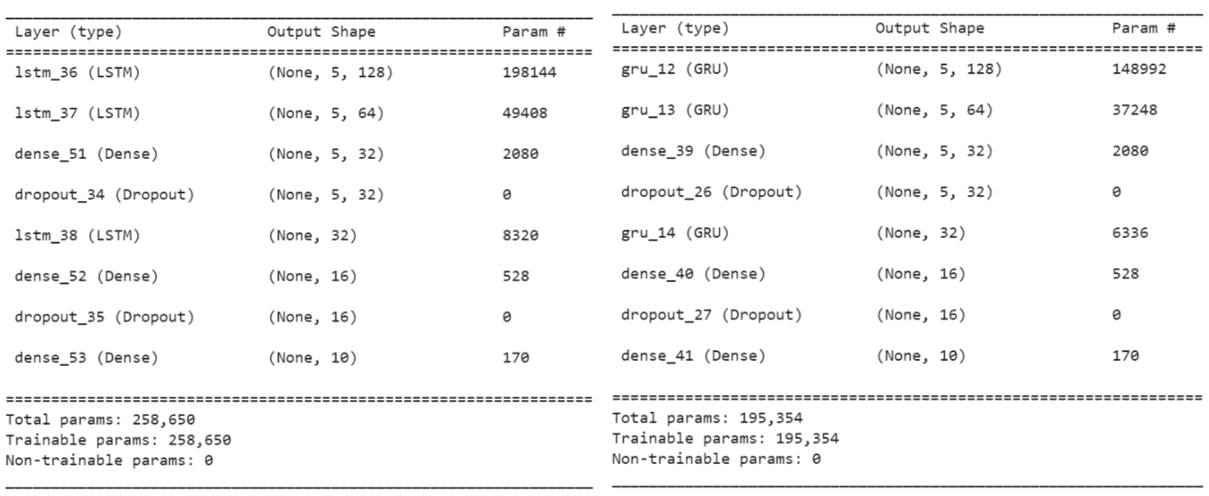
Dropout is a regularization technique that randomly drops some units from a layer in order to prevent overfitting during the learning process. A higher dropout may positively affect the generalization ability of the model, however, it may also damage the performance of the model in complex data.

4.3.2 LSTM and GRU Architecture

For classification purposes it was decided to use the LSTM and GRU models which are included in the category of RNN. Specifically in this thesis the RNN models were selected based on their ability to learn through changes over time and their effectiveness to extract temporal sequential information of movements such as dynamic signs.

In this study both the LSTM and GRU algorithms were tested using the same architecture in order to examine the effectiveness of each model in the same conditions. In general, both LSTM and GRU are using gating units which are responsible for controlling the information flow inside and outside of the network, however, the LSTM models use more gates resulting in the increase of their ability to recognize high complexity patterns. On the other hand, GRU models are faster and lighter than LSTM due to the fact that they have less parameters and complexity.

The input of the RNN models were the features extracted from VGG_16 and MediaPipe which have been presented in a previous section. As we can see in Fig.29 in the suggested model the first hidden layer is LSTM/GRU consisting of 128 units. Following that, we can observe two groups of layers. The first group has the LSTM/GRU layer consisting of 64 units, followed by a Dense layer with 32 units and Relu as activation function. On the other hand, even though the second group has the same layers as the first, the units were decreased in half. In addition, each Dense layer was followed by a Dropout layer with a rate of 0.2. Finally, these layers are followed by a fully connected layer with a SoftMax activation function which is used to predict the output.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_36 (LSTM)              (None, 5, 128)            198144

 lstm_37 (LSTM)              (None, 5, 64)             49408

 dense_51 (Dense)            (None, 5, 32)             2080

 dropout_34 (Dropout)        (None, 5, 32)             0

 lstm_38 (LSTM)              (None, 32)                8320

 dense_52 (Dense)            (None, 16)                528

 dropout_35 (Dropout)        (None, 16)                0

 dense_53 (Dense)            (None, 10)                170

=================================================================
Total params: 258,650
Trainable params: 258,650
Non-trainable params: 0
_________________________________________________________________
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_12 (GRU)                (None, 5, 128)            148992

 gru_13 (GRU)                (None, 5, 64)             37248

 dense_39 (Dense)            (None, 5, 32)             2080

 dropout_26 (Dropout)        (None, 5, 32)             0

 gru_14 (GRU)                (None, 32)                6336

 dense_40 (Dense)            (None, 16)                528

 dropout_27 (Dropout)        (None, 16)                0

 dense_41 (Dense)            (None, 10)                170

=================================================================
Total params: 195,354
Trainable params: 195,354
Non-trainable params: 0
_________________________________________________________________
```

Figure 29: Proposed LSTM (left) and GRU (right) Architectures.

The model was compiled using categorial cross-entropy as loss function, the Adam optimizer, and accuracy, precision and recall as metrics. Then the dada was trained with batch size of 64 and 50 epochs.

The hyperparameters in the suggested model are selected after multiple experiments until the model reaches a satisfactory level concerning the accuracy and the complexity of the model.

4.3.3 Evaluation Metrics

The performance of the model was evaluated using the three different metrics Accuracy, Precision and Recall.

4.3.4 Accuracy

Accuracy is the most commonly used simple metric for classification. It represents the ratio of the correctly classified predictions out of the total number of predictions.

The formula for accuracy is below:

$$\frac{TRUE\ POSITIVE\ + TRUE\ NEGATIVE}{TRUE\ POSITIVE\ + FALSE\ POSITIVE\ + TRUE\ NEGATIVE\ +\ FALSE\ NEGATIVE}$$

4.3.5 Precision

Precision represents the ratio of the correctly classified positive predictions out of the total number of positive predictions. It is a useful metric when the cost of False positive is very high, and the cost of False negative is very low.

The formula for precision is below:

$$\frac{TRUE\ POSITIVE}{TRUE\ POSITIVE\ \ + FALSE\ POSITIVE}$$

4.3.6 Recall

Recall represents the ratio of the correctly classified positive out of all the actual positives. It is a useful metric when the cost of False negative is high.

The formula for precision is below:

$$\frac{TRUE\ POSITIVE}{TRUE\ POSITIVE\ \ + FALSE\ NEGATIVE}$$

5. EPRERIMENTAL RESULTS AND DISCUSSION

In this section we present the experimental results of this thesis concerning the performance of the models and the misclassification of the testing dataset.

5.1 Model Performance

The accuracy curves of the models for each method are shown in Fig.30 and Fig.31.

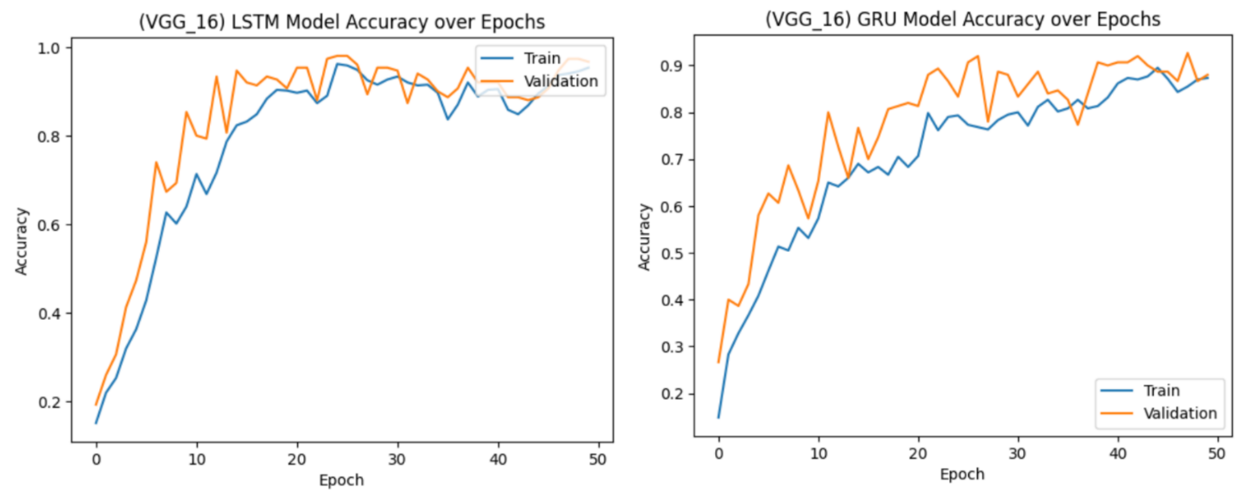


Figure 30: Accuracy curves for VGG_16 features.

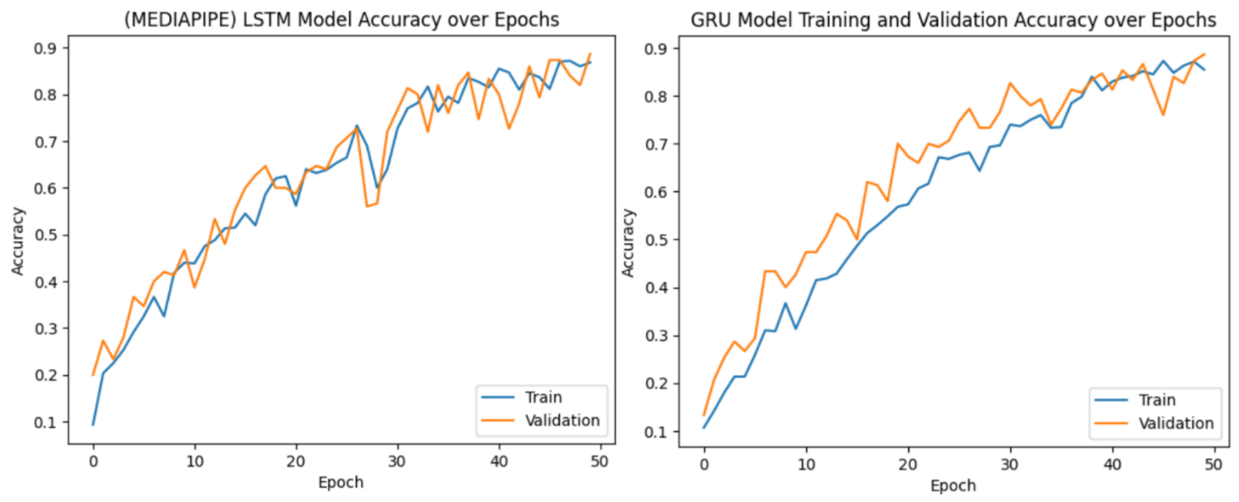


Figure 31: Accuracy curves for MediaPipe features.

Furthermore, the recognition accuracy of the DSL10_Dataset for VGG_16 and MediaPipe is shown in Table.1 and Table.2 respectively. The tables provide the results for both classification models (LSTM and GRU) using three metrics (Accuracy, Precision and Recall).

As we can observe in Table.1 the combination of VGG_16 for feature extraction and LSTM for classification, achieved average recognition accuracy of 96.44% which exceeded the performance of the other combinations. On the other hand, we can see that the average accuracy for the VGG_16 and GRU combination is considerably lower at 89.18% which leads us to the fact that, even though GRU is significantly faster with less parameters than LSTM, LSTM can recognize complex patter more effectively.

Table 1: Evaluation of VGG_16 features.

	Accuracy	Precision	Recall	Average
LSTM	96.67%	96.64%	96.00%	96.44%
GRU	88.00%	92.86%	86.67%	89.18

Concerning the MediaPipe method as it appears in Table.2, the results of LSTM and GRU are very close with average accuracy of 88.62% and 89.03% respectively. This may result from the fact that the total number of features for each frame using MediaPipe is 258 which is significantly lower than the total number of features using VGG_16 which is 25,088.

Table 2: Evaluation of MediaPipe features.

	Accuracy	Precision	Recall	Average
LSTM	88.67%	89.19%	88.00%	88.44%
GRU	88.67%	90.41%	88.00%	89.03%

Moreover, after multiple experiments it has been observed that the pretrained model VGG_16 needs less time to extract features than MediaPipe, despite the small number of key-points produced as output. However, given that the VGG_16 produces a significantly higher amount of characteristics, it also entails that the LSTM and GRU require more learning time in order to complete the classification.

5.2 Classes Prediction

This section discusses the output of the confusion matrix shown in Fig.32 and Fig.33 for the VGG_16 and MediaPipe respectively. A confusion matrix is a matrix that visualizes the number and type of misclassifications after the testing process. The words with the highest number of incorrect predictions are “no” and “sorry”, each with 5 misclassifications out of 15.

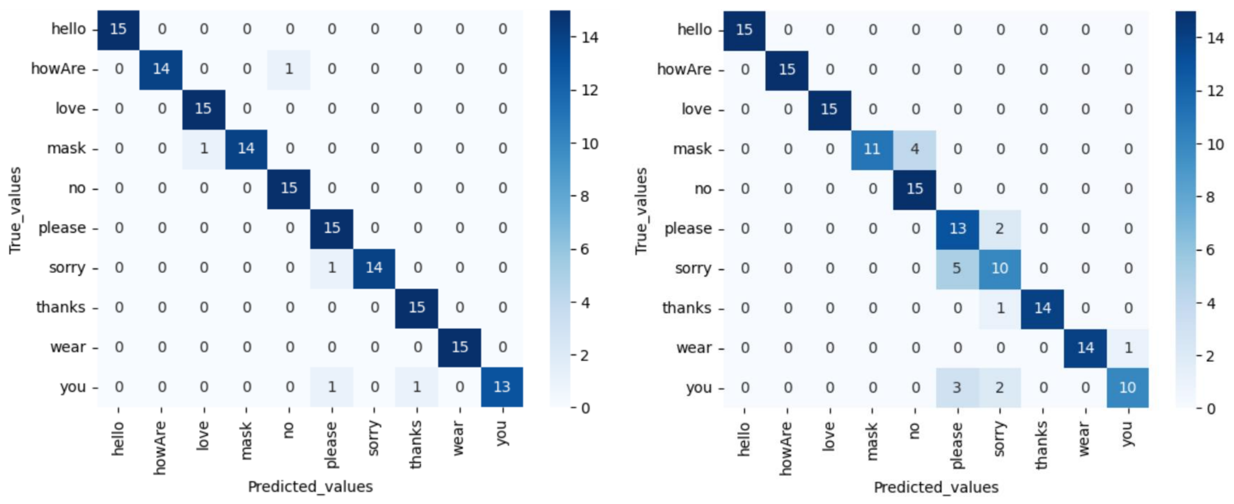


Figure 32: Confusion Matrix of the LSTM (left) and GRU (right) prediction for VGG_16 features.

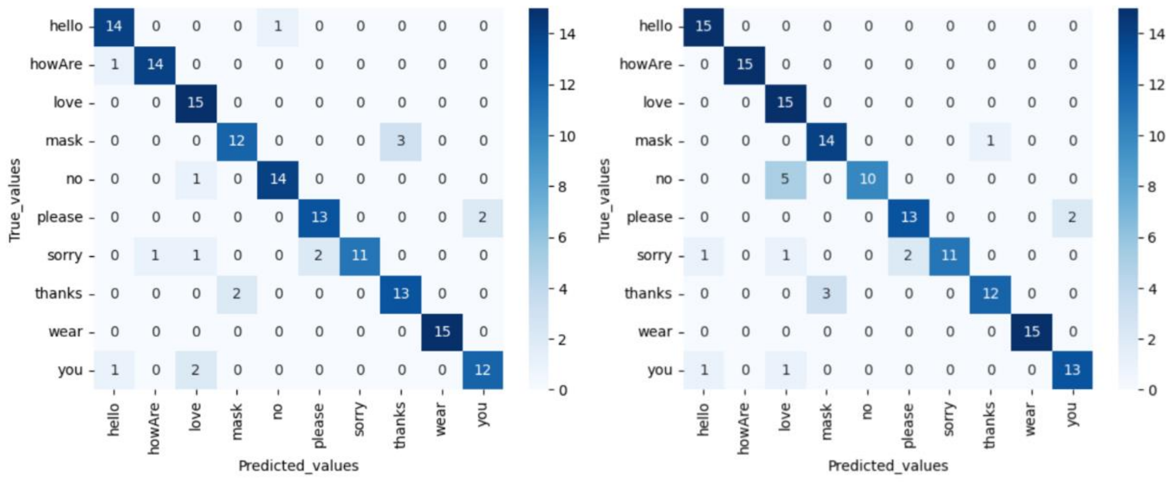


Figure 33: Confusion Matrix of LSTM (left) and GRU (right) prediction for MediaPipe features.

From the matrix we can observe that the model confuses the word “no” with the word “love”. As we can see in Fig.34 a possible reason behind this misinterpretation is that the movement for both dynamic signs are very similar because both gestures are performed with a hand lifted side of the neck and a vertical movement of fingers.

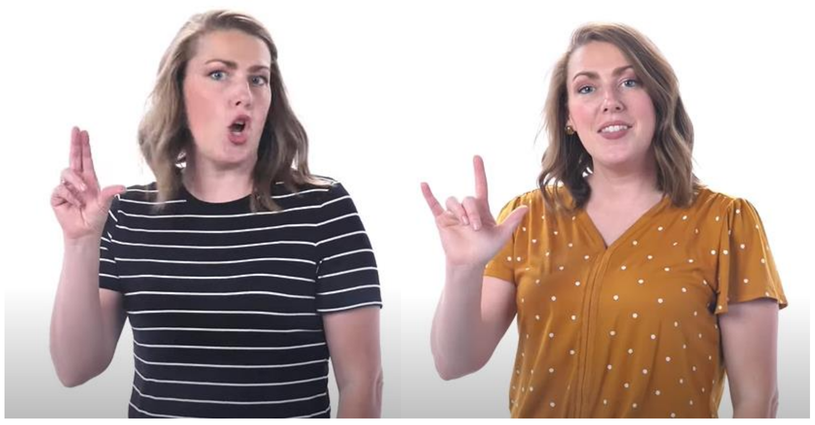


Figure 34: Snapshots from the movements of the sign for words “no” (left) and “love” (right)[49].

Using Similar reasoning, the model also misinterprets the word “sorry” with the word “please” because as we can see in Fig.35 both words share the same circular movement close to chest, with the main difference that the movement of “sorry” is performed using a close hand.

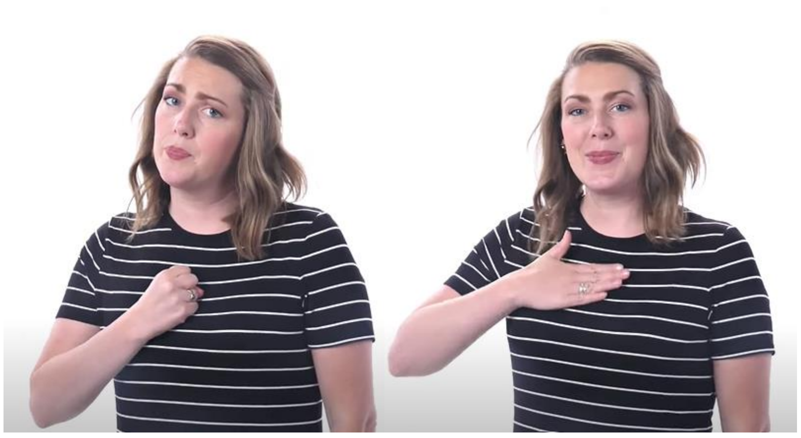


Figure 35: Snapshots from the movements of the signs for the words “sorry” (left) and “please” (right) [49].

6. CONCLUSION AND FUTURE WORK

The objective of this thesis was to apply and compare various deep learning methods in order to provide better communication between the deaf community and hearing majority.

According to the World Health Organization (WHO), almost 446 million people worldwide suffer from impaired hearing loss and studies reveal that this number may continue to rise in the future. This makes the desire to develop an effective sign language translator urgent in order to facilitate the life of the hearing-impaired community.

This research is divided into two approaches concerning feature extraction. The pretrained model VGG_16 and the MediaPipe framework were used in order to extract the spatial feature out from the videos of the DSL10_Dataset. Following that, the features from each method were passed through the proposed LSTM and GRU models in order to identify the relationship between the sequences and produce the overall prediction.

From the experimental results it can be observed that the combination of the VGG_16 and the proposed LSTM model achieved an average performance of 96.44% and outperformed the other combination. Even though using the features from VGG_16 surpass the classification performances of MediaPipe, the number of features produced by VGG_16 was considerably higher resulting in the increase of the learning time. Moreover, concerning the results of LSTM in comparison with GRU it can be concluded that LSTM outperformed GRU especially in a high complexity sequence, however, due to the fact that GRU requires less parameters than LSTM, GRU can be proved more efficient in a case of low complexity data. Finally, it has been observed that the models become erroneous when signs have very similar movements.

In the future, creating a large dataset with a variety of signers will be considered. In addition, more advanced algorithms for data preprocessing and learning can be used in order to recognize real time videos with more complex environments and different durations.

ABBREVIATIONS - ACRONYMS

WHO	World Health Organization
SLR	Sign Language Recognition
AI	Artificial intelligence
ML	Machine Learning
DL	Deep learning
AF	Activation Functions
tanh	Hyperbolic Tangent
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
RestNet	Residual Network
HMM	Hidden Markov Model
SVM	Support Vector Machine
SSD	Single Shot Detector
ESHR	Extra Spatial Hand Relation
HP	Hand Pose
WLASL	Word-Level American Sign Language
SSIM	Structural Similarity Index
RGB	Red Green Blue

REFERENCES

[1] B. Sonare, A. Padgal, Y. Gaikwad, and A. Patil, 'Video-based sign language translation system using machine learning', in *2021 2nd International Conference for Emerging Technology, INCET 2021*, Institute of Electrical and Electronics Engineers Inc., May 2021. doi: 10.1109/INCET51464.2021.9456176.

[2] A. Núñez-Marcos, O. Perez-de-Viñaspre, and G. Labaka, 'A survey on Sign Language machine translation', *Expert Systems with Applications*, vol. 213. Elsevier Ltd, Mar. 01, 2023. doi: 10.1016/j.eswa.2022.118993.

[3] D. Li, C. Rodriguez Opazo, X. Yu, and H. Li, 'Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison'. [Online]. Available: https://dxli94.github.io/

[4] G. H. Samaan *et al.*, 'MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition', *Electronics (Switzerland)*, vol. 11, no. 19, Oct. 2022, doi: 10.3390/electronics11193228.

[5] R. Rastgoo, K. Kiani, and S. Escalera, 'Video-based isolated hand sign language recognition using a deep cascaded model', *Multimed Tools Appl*, vol. 79, no. 31–32, pp. 22965–22987, Aug. 2020, doi: 10.1007/s11042-020-09048-5.

[6] M. M. Balaha *et al.*, 'A vision-based deep learning approach for independent-users Arabic sign language interpretation', *Multimed Tools Appl*, Feb. 2022, doi: 10.1007/s11042-022-13423-9.

[7] A. A. Barbhuiya, R. K. Karsh, and R. Jain, 'CNN based feature extraction and classification for sign language', *Multimed Tools Appl*, vol. 80, no. 2, pp. 3051–3069, Jan. 2021, doi: 10.1007/s11042-020-09829-y.

[8] 'What is Artificial Intelligence (AI) ? | IBM'. https://www.ibm.com/topics/artificial-intelligence (accessed Feb. 13, 2023).

[9] *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*.

[10] 'What is the Turing Test?' https://www.techtarget.com/searchenterpriseai/definition/Turing-test (accessed Feb. 13, 2023).

[11] 'What Is the Definition of Machine Learning? | Expert.ai | expert.ai'. https://www.expert.ai/blog/machine-learning-definition/ (accessed Feb. 13, 2023).

[12] 'AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference? | IBM'. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks (accessed Feb. 13, 2023).

[13] L. Alzubaidi *et al.*, 'Review of deep learning: concepts, CNN architectures, challenges, applications, future directions', *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[14] ‘Machine Learning vs Deep Learning’. https://www.linkedin.com/pulse/machine-learning-vs-deep-harpreet-singh-sachdev (accessed May 11, 2023).

[15] ‘Supervised vs. Unsupervised Learning: What’s the Difference? | IBM’. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (accessed Feb. 13, 2023).

[16] ‘What are Neural Networks? | IBM’. https://www.ibm.com/topics/neural-networks (accessed Feb. 13, 2023).

[17] ‘The Concept of Artificial Neurons (Perceptrons) in Neural Networks | by Rukshan Pramoditha | Towards Data Science’. https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc (accessed May 11, 2023).

[18] ‘A Layman’s Guide to Deep Neural Networks | by Jojo John Moolayil | Towards Data Science’. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb (accessed May 11, 2023).

[19] S. Sharma, S. Sharma, and A. Athaiya, ‘ ACTIVATION FUNCTIONS IN NEURAL NETWORKS’, 2020. [Online]. Available: http://www.ijeast.com

[20] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, ‘Activation Functions: Comparison of trends in Practice and Research for Deep Learning’, Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.03378

[21] ‘Activation Functions: Sigmoid vs Tanh | Baeldung on Computer Science’. https://www.baeldung.com/cs/sigmoid-vs-tanh-functions (accessed May 11, 2023).

[22] K. O’Shea and R. Nash, ‘An Introduction to Convolutional Neural Networks’, Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.08458

[23] G. Chen, ‘A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation’, Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.02583

[24] Z. C. Lipton, J. Berkowitz, and C. Elkan, ‘A Critical Review of Recurrent Neural Networks for Sequence Learning’, May 2015, [Online]. Available: http://arxiv.org/abs/1506.00019

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, ‘Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling’, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.3555

[26] R. C. Staudemeyer and E. R. Morris, ‘Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks’, Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.09586

[27] F. C. Maria Bianchi Enrico Maiorino Michael Kampffmeyer Antonello Rizzi Robert Jenssen, ‘Recurrent Neural Networks for Short-Term Load Forecasting An Overview and Comparative Analysis’. [Online]. Available: http://www.springer.com/series/10028

[28] ‘The structure of GRU network. | Download Scientific Diagram’. https://www.researchgate.net/figure/The-structure-of-GRU-network_fig4_351175212 (accessed May 11, 2023).

[29] C. Can, Y. Kaya, and F. Kilic, ‘A deep convolutional neural network model for hand gesture recognition in 2D near-infrared images’, *Biomed Phys Eng Express*, vol. 7, no. 5, Sep. 2021, doi: 10.1088/2057-1976/ac0d91.

[30] ‘Deep Learning Kit | MENTOR HELLAS’. https://mentorhellas.com/deep-learning-kit (accessed May 11, 2023).

[31] J. Wu, ‘Introduction to Convolutional Neural Networks’, 2017.

[32] ‘Image Filters with Convolutions - Pooja Mahajan - Medium’. https://poojamahajan5131.medium.com/image-filters-with-convolutions-9104bba1ce12 (accessed May 11, 2023).

[33] ‘Convolutional Neural Networks : The Theory - Bouvet Norge’. https://www.bouvet.no/bouvet-deler/understanding-convolutional-neural-networks-part-1 (accessed May 11, 2023).

[34] O. Bayat, S. Aljawarneh, H. F. Carlak, International Association of Researchers, Institute of Electrical and Electronics Engineers, and Akdeniz Üniversitesi, *Proceedings of 2017 International Conference on Engineering & Technology (ICET’2017) : Akdeniz University, Antalya, Turkey, 21-23 August, 2017*.

[35] G. D. Aman *et al.*, ‘PERFORMANCE ANALYSIS OF VISUAL GEOMETRY GROUP ALGORITHMS ON PNEUMONIA CLASSIFICATION’, 2022, doi: 10.48047/NQ.2022.20.15.NQ88597.

[36] ‘Explain Pooling layers: Max Pooling, Average Pooling, Global Average Pooling, and Global Max pooling. - For Machine Learning’. https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/ (accessed May 11, 2023).

[37] A. A. Barbhuiya, R. K. Karsh, and R. Jain, ‘ASL Hand Gesture Classification and Localization Using Deep Ensemble Neural Network’, *Arab J Sci Eng*, 2022, doi: 10.1007/s13369-022-07495-w.

[38] ‘Classic Structure of AlexNet Deep Neural Network [24]. | Download Scientific Diagram’. https://www.researchgate.net/figure/Classic-Structure-of-AlexNet-Deep-Neural-Network-24_fig3_335524166 (accessed May 11, 2023).

[39] K. Simonyan and A. Zisserman, ‘Very deep convolutional networks for large-scale image recognition’, *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[40] S. Mascarenhas and M. Agarwal, ‘A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification’, in *Proceedings of IEEE International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications, CENTCON 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 96–99. doi: 10.1109/CENTCON52345.2021.9687944.

[41] D. Theckedath and R. R. Sedamkar, ‘Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks’, *SN Comput Sci*, vol. 1, no. 2, Mar. 2020, doi: 10.1007/s42979-020-0114-9.

[42] ‘Residual Net - ResNet’. https://www.linkedin.com/pulse/residual-net-resnet-carmel-shalgi (accessed May 11, 2023).

[43] 'Inception-v3 Explained | Papers With Code'. https://paperswithcode.com/method/inception-v3 (accessed May 11, 2023).

[44] S. K. Ko, J. G. Son, and H. Jung, 'Sign language recognition with recurrent neural network using human keypoint detection', in *Proceedings of the 2018 Research in Adaptive and Convergent Systems, RACS 2018*, Association for Computing Machinery, Inc, Oct. 2018, pp. 326–328. doi: 10.1145/3264746.3264805.

[45] Institute of Electrical and Electronics Engineers and IEEE Signal Processing Society, *2016 IEEE International Conference on Image Processing : proceedings : September 25-28, 2016, Phoenix Convention Center, Phoenix, Arizona, USA*.

[46] 'Hand landmarks of MediaPipe Hands | Download Scientific Diagram'. https://www.researchgate.net/figure/Hand-landmarks-of-MediaPipe-Hands_fig1_354115187 (accessed May 11, 2023).

[47] 'Body Posture Detection & Analysis System using MediaPipe'. https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/ (accessed May 11, 2023).

[48] 'GitHub - k-m-irfan/simplified_mediapipe_face_landmarks: Extracts essential Mediapipe face landmarks and arranges them in a sequenced order.' https://github.com/k-m-irfan/simplified_mediapipe_face_landmarks (accessed May 11, 2023).

[49] 'Baby Sign Language - YouTube'. https://www.youtube.com/channel/UCjyPt5_3cWbyE38Yq-v4LdQ (accessed May 11, 2023).