Emirati Sign Language Detection and Translation

Applying Deep Learning and Dynamic Time Warping with Google MediaPipe to Convert Motion to Tangible Data

Mohannad Janahi
Department of Computing and Informatics
University of Sharjah
Sharjah, United Arab Emirates
U23106345@sharjah.ac.ae

***Abstract*— The Emirati Sign Language is understood by a small percentage of the population, and having a smart translator would prove very beneficial. The main objective of this research is to tackle that problem and to provide a practical solution for the deaf, hard of hearing, and mute community.**

Two methods are employed and compared with each other: Dynamic Time Warping and Deep Learning. All programming is done through Python and its corresponding libraries.

Dynamic Time Warping is chosen for its ability to vectorize time-series data, as the Emirati Sign Language is deeply reliant on movement, unlike other Sign Languages. Deep Learning is chosen for its ability to fit and classify very non-linear data as is the case with image classification.

Both methods yield accurate and satisfactory results. However, using deep learning is more practical, and combining the two methods together might be the best way to tackle this problem, as each method has its weaknesses that can be covered up by the other method. This research can be applied on any video feed, including live video call translations.

Keywords—deep learning; transfer learning; dynamic time warping; machine learning; motion detection; artificial intelligence; neural networks; vgg-19; resnet50

I. INTRODUCTION

This research aims to tackle the problems faced by the population not understanding sign language, by giving a direct translation of each sign made by the speaker. Artificial Intelligence will be used to classify each sign to its respective word. The goal of this research is to find the best method of classification, by testing multiple methods and sub methods, and comparing them to find the best performing model.

The Emirati Sign Language Dictionary was released to the public in 2018 [3], yet no major promotion and advertisement has been done for it causing most of the local population to not even be aware of its existence. This makes communication challenging for the minority group that speaks through this distinct sign language.

It was created by the Zayed Higher Organization for People of Determination [6], with the goal of following the directives of the late Sheikh Zayed bin Sultan Al Nahyan and his late son Sheikh Khalifa bin Zayed Al Nahyan, may their souls rest in peace. This sign language is part of their efforts in their goal of rehabilitating disabled people for the inclusion into the community [8]. The language has up to 5000 words, with instructions and tutorials available through their website for anyone to see and learn.

Sign Language Classification is not a new topic. Many applications can be found for different languages, with each attempting the problem differently. However, not all sign languages are the same, as some rely on shapes, movement, or a mix of both, while some use one hand and others use both. The Emirati Sign Language poses a new challenge, as it is very variant. Some words use one hand, while others use both. Some use shapes, some use movement, and some use a combination of all techniques.

The dataset to be used was created by myself. It includes 5 labelled classes (hundred, mother, ninety-two, relax, and salute), with each class having 166 video entries. The dataset is considered small for a real-world application. The number of classes is small, and the number of videos per class can preferably be higher. However, for the goal of this research, it should prove sufficient to see how well the model can deal with such a task and if it can differentiate between the classes. A good accuracy score is critical for theoretical testing, otherwise the model would be destined to fail in a real-world application.

Throughout the report, the research will be linked with previous work, and then the technical aspects of the research will be explained, specifically the working principle of each method used and their applications in sign language detection. Finally, the research will be critiqued and methods to improve and extend on the current research will be discussed.

II. PREVIOUS WORK

A. MediaPipe and Dynamic Time Warping

The MediaPipe Framework is a suite of libraries and tools created by Google and aimed at developers who want to implement machine learning techniques into their projects [4]. Within the framework, the hand landmark detection task will be used to track the position of each joint in each hand, and vectorize it, in 2D. This can be used alongside dynamic time warping to calculate the “distance” between an input vector and a reference vector. This method was used by Gabriel

Guerin for French Sign Language detection [2], and dynamic time warping was implemented by Stan Salvador and Philip Chan [5].

B. Pre-processing a Video for a Neural Network Input

Neural networks require inputs in the shape of a tensor, such as a 2D image. However, for the case of a time-series data such as a video, it poses a new hurdle. The input will need to be pre-processed in a way that a neural network would be able to accept it. The pre-processing algorithm of choice to convert a video to an image that can be converted to a tensor array, while still retaining information from all frames, is the difference of frames algorithm, which is used by Angela Caliwag, Han-Jeon Hwang, Sang-Ho Kim, and Wansu Lim in their research for movement in a video detection [1]. This research will utilize the algorithm mentioned above to convert the video instances to photos to be fed to a neural network.

III. USING GOOGLE MEDIAPIPE WITH DYNAMIC TIME WARPING

Dynamic time warping is an algorithm that can find the optimal alignment between two time-series data, such as a video, even if their lengths are mismatched. It can be used to compare how similar two series are. In case of a mismatch in length, the algorithm warps the series by either shrinking or stretching it along its axis [5].

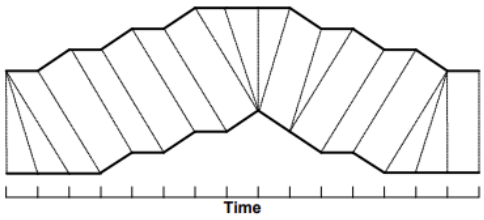

Fig. 1. Example of dynamic time warping between two time series data

Fig. 1 shows the working process of the algorithm. The measured distance is the difference between the two series post-warping. It is calculated by summing the differences between each pair of points that are connected by the vertical lines between the two series. Dynamic time warping has an $O(N^2)$ time and space complexity, which limits the practicality of the algorithm. In the case of sign language detection, the videos in the dataset do not surpass 3 seconds. With a 30 frames per second sampling rate, which is the case in the dataset, there would be at most 90 frames in each data entry. That makes dynamic time warping an ideal algorithm for this specific use case.

Google's MediaPipe Framework contains a holistic task that will be used to detect hands in a video stream. Furthermore, it can be used to extract the x and y coordinates of each joint in each hand, adding up to 21 coordinates per hand.

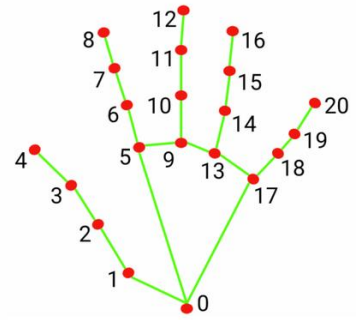

Fig. 2. The joints that can be detected by the MediaPipe holistic model

The output will be a list of arrays of x and y positions of each joint. Each array will contain the x and y positions of each joint in a single frame. Since order is important, the frames will be recorded sequentially. The list index will be the same as the frame index. That list will then be used to apply dynamic time warping between the input series and the reference series.

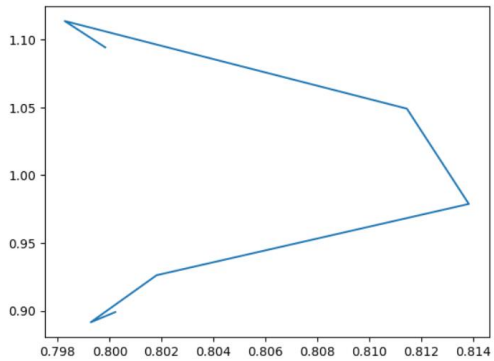

Fig. 3. Wrist coordinates extracted from a hand using MediaPipe

Fig. 3 shows how the extracted coordinates look like on a plot, where the x-axis is the x coordinates and the y-axis are the y coordinates of the wrist from a hand, over time.

First, The MediaPipe task is run on the entire dataset, and the positions of the coordinates of the joints of each frame from each video is extracted and recorded in a dictionary. A dataframe can be created, containing the name of the sign, the extracted coordinates from the joints from each hand, and an empty distance column that will be used to store the result of dynamic time warping between the input and each entry in the dataframe. Then, the MediaPipe task is run on the input video, and the coordinates of the joints in each frame are extracted. The input is now compared with the entire dataframe, through dynamic time warping. Each comparison will yield its own distance. The shortest distance can then be considered as the predicted sign, as it is the closest match to the input.

name	distance
ninetytwo	32.675345
ninetytwo	33.356644
ninetytwo	33.642933
ninetytwo	36.445593

Fig. 4. The result of running dynamic time warping on the input video (gesturing ninety two) and the dataframe, and then sorting by distance, lowest to highest

The predicted class in Fig. 4 is then “ninety two”. A threshold value is chosen for the distance, where in any distance higher than that value would mean that the model failed to predict the class. That threshold adds a concept of uncertainty to the model, and prevents it from predicting a class when the calculated distance is too high.

IV. Deep Learning with Difference of Frames Algorithm

The difference of frames algorithm is a motion detecting method where consecutive frames are subtracted from each other, removing areas where the pixel values do not change. Finally, all the differences are added together, yielding one frame summarizing all the motion detected throughout the frames. The pseudocode followed is given by Angela Caliwag, Han-Jeon Hwang, Sang-Ho Kim, and Wansu Lim [1].

1 $frame_{count}$ = total number of frames extracted;
2 $frame_s = 0$;
3 **for** each frame, $frame_i$, **do**
4 import $frame_i$;
5 $frame_k = frame_i$;
6 **if** $i \neq 1$ **do**
7 $frame_d = frame_{k-1} - frame_k$;
8 **else**
9 $frame_{k-1} = frame_k$;
10 **end if**
11 **for** each element in $frame_d$, $frame_{d,jk}$ **do**
12 **if** $frame_{d,jk} > 200$ **or** $frame_{d,jk} < 55$ **do**
13 $frame_{d,jk} = 0$;
14 **end if**
15 **end for**
16 **for** each element in $frame_d$, $frame_{d,jk}$ **do**
17 **if** $frame_{d,jk} \neq 0$ **do**
18 $frame_{d,jk}{}^* = \lfloor \frac{255}{frame_{count}-1} \rfloor$;
19 **end if**
20 **end for**
21 $frame_s = frame_s + frame_d$;
22 $frame_{k-1} = frame_k$;

Fig. 5. The difference of frames algorithm pseudocode

The code was slightly altered to optimize it for speed. Lines 16-18 utilize a nested for loop, which greatly increases cell runtime in python. A better solution is to use the where() function from the Numpy library, instead of looping through each pixel in each row. That alteration cut the runtime of the algorithm from taking minutes per video to less than 5 seconds.

An example of the output of the algorithm is shown in Fig. 6 below.

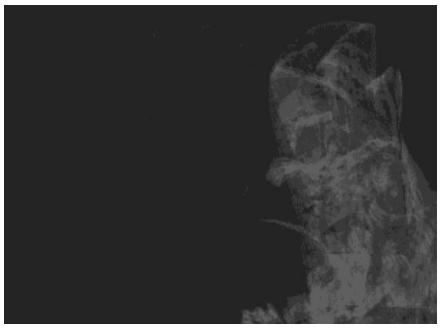

Fig. 6. Difference of frames output (brightness increased for ease of view)

The entire video dataset is converted to a difference of frames image dataset, using the given algorithm, to prepare it for a deep learning application. The dataset, for the purposes of this research, consists of 5 classes, with 120 training images per class, 16 testing images per class, and 30 validation images per class.

A. *Transfer Learning*

Transfer learning refers to using a pre-trained neural network to classify a new set of outputs, keeping the early layers as generic feature extractors, and removing the final layers that are too specific to a certain classification application. The original layer weights are kept unchanged. 2 famous neural networks will be used and compared: resnet-50 and vgg-19.

The last 2 layers of the resnet-50 architecture are removed and replaced with a Flatten layer, and a Dense layer having 5 outputs and a softmax activation function.

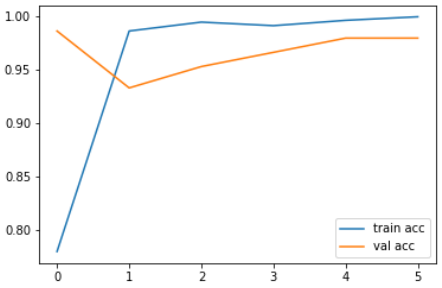

Fig. 7. Training and validation accuracy of ResNet-50 model

Fig. 7 shows that the model was quick to overfit, and that could be due to 2 factors:

1. The model being too complex for the given task
2. The dataset being too small for the given task
3. The dataset having weak generalization

```
[[ 4  0  0  0  0]
 [ 0 16  0  0  0]
 [12  0 16  0  0]
 [ 0  0  0 15  0]
 [ 0  0  0  0 16]]
```

Fig. 8. ResNet-50 confusion matrix

The confusion matrix shows where the model suffers. All of the error comes from mislabeling index 0 (ninety two) as index 2 (salute). The model's test accuracy is 85%

The same logic can be applied to a different model: VGG-19. The last two layers are replaced by a flattening layer and a dense prediction layer with 5 outputs and a softmax activation function.

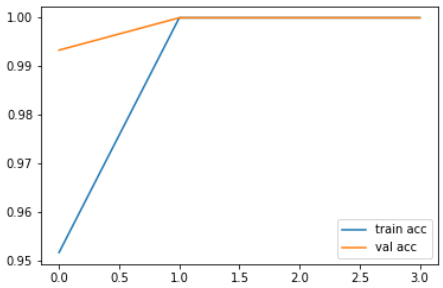

Fig. 9. VGG-19 training and validation accuracy

The VGG-19 model was quick to overfit as well. The same factors as the ResNet-50 model apply.

```
[[ 5  0  0  0  0]
 [ 0 16  0  0  0]
 [11  0 16  0  0]
 [ 0  0  0 15  1]
 [ 0  0  0  0 15]]
```

Fig. 10. VGG-19 confusion matrix

Almost the same performance can be observed. The testing accuracy is also 85%. All of these signs point to transfer learning not to be the suitable method with this specific dataset.

B. Custom Neural Network

A simpler neural network with less layers can be utilized to increase the model's performance. The new model consists of 4 folds of a convolutional layer followed by a maxpooling layer. Finally, a dense layer is added before the prediction layer with 5 outputs and a softmax activation function. Convolutional layers are the heart of convolutional neural networks. It applies the convolution function to the input with a filter to create feature maps [7]. Maxpooling layers simply shrink the image by taking a pixel window, and replacing them with one pixel which is the maximum of the pixels in the window. A dense layer is a fully connected layer, where each neuron is connected to the other through a weight, and a weighted sum of the features is calculated. Finally, the softmax activation function is the function that converts the output to a conditional probability. It is similar to the Sigmoid function that is used in binary classification.

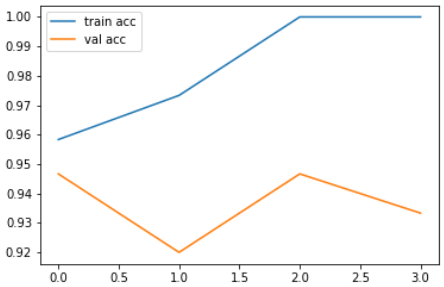

Fig. 11. Custom Neural Network training and validation accuracy

```
[[ 8  0  0  0  0]
 [ 0 12  0  0  0]
 [ 8  0 16  0  0]
 [ 0  4  0 15  0]
 [ 0  0  0  0 16]]
```

Fig. 12. Custom Neural Network confusion matrix

Although the tesing accuracy is still at 85%, we can see a minor improvement in classifying index 0 (ninety two) and index 2 (salute). However, the model is still prone to overfitting, so other measures need to be taken.

Regularization can be applied to counter overfitting of the model. Dropout layers deactivate certain layers randomly, making it harder for the model to memorize the training data.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 223, 223, 32)	416
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
dropout (Dropout)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 110, 110, 32)	4128
max_pooling2d_1 (MaxPooling2	(None, 55, 55, 32)	0
conv2d_2 (Conv2D)	(None, 54, 54, 64)	8256
max_pooling2d_2 (MaxPooling2	(None, 27, 27, 64)	0
conv2d_3 (Conv2D)	(None, 26, 26, 64)	16448
max_pooling2d_3 (MaxPooling2	(None, 13, 13, 64)	0
dropout_1 (Dropout)	(None, 13, 13, 64)	0
flatten (Flatten)	(None, 10816)	0
dense (Dense)	(None, 512)	5538304
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2565

Fig. 13. Custom Neural Network confusion matrix

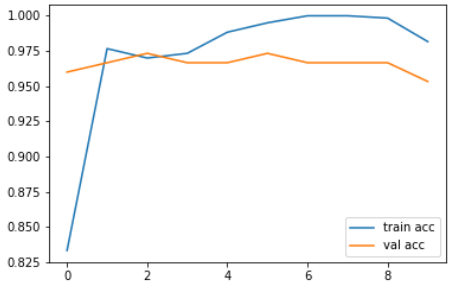

Fig. 14. Custom Neural Network with dropout layers training and validation accuracy

The model started showing signs of overfitting after 6 epochs, compared to the previous models that started overfitting much earlier.

```
[[ 6  0  0  0  0]
 [ 0 16  0  0  0]
 [10  0 16  0  0]
 [ 0  0  0 15  0]
 [ 0  0  0  0 16]]
```

Fig. 15. Custom Neural Network with dropout confusion matrix

There is a small improvement in the performance, with testing accuracy now at 87%.

L1 and L2 regularization can be applied to the layers as well. With L1 kernel regularization applied to the first convolutional layer and the last dense layer with a learning rate of 0.0025, the following performance is observed:

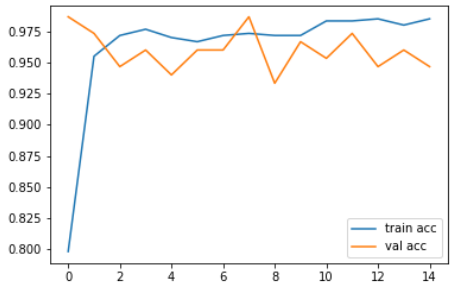

Fig. 16. Custom Neural Network with dropout and L1 norm

```
[[ 8  0  0  0  0]
 [ 0 16  0  0  0]
 [ 8  0 16  0  0]
 [ 0  0  0 15  0]
 [ 0  0  0  0 16]]
```

Fig. 17. Custom Neural Network with dropout and L1 norm confusion matrix

The effect of the L1 regularization technique has been a positive one, with testing accuracy now at 90%. However, there is still a struggle in labeling “ninety two” and “salute”.

Finally, L2 regularization can be applied to the neural network layers. A kernel regularizer is applied to all the dense and convolutional layers, with a learning rate of 0.005.

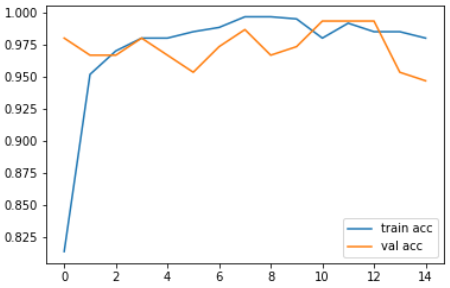

Fig. 18. Custom Neural Network with dropout and L2 norm

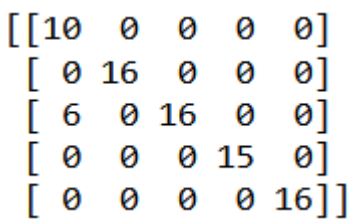

Fig. 19. Custom Neural Network with dropout and L2 norm confusion matrix

An even better performance is observed, with testing accuracy now at 92%.

C. *Applying K-fold cross-validation to the model*

When dealing with a small and limited dataset, the best course of action is to use the K-fold cross-validation algorithm. Rather than splitting the dataset into training, validation, and testing, the entire dataset can be split into K folds. For each iteration, one fold will be used for training, while the remaining folds will be used for cross-validation. This method maximizes the amount of training data and often returns better results on smaller datasets. Theoretically, the best performance will be achieved when the number of folds is equal to the amount of data. However, that would be computationally expensive. The best practice is to use a number between 5 and 10. For the purpose of this research, 7 folds are chosen. Since the difference of frames algorithm returns a lot of dead pixels, it is wise to use L1 regularization to remove unneeded features (pixels). Regularization is applied on the 1st (convolutional) layer and 9th (dense) layer. The dense layer is the most susceptible to large weights due to the huge number of trainable parameters. The model summary is shown in Fig. 20.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 223, 223, 16)	208
max_pooling2d (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_1 (Conv2D)	(None, 110, 110, 16)	1040
max_pooling2d_1 (MaxPooling2	(None, 55, 55, 16)	0
conv2d_2 (Conv2D)	(None, 55, 55, 32)	544
max_pooling2d_2 (MaxPooling2	(None, 27, 27, 32)	0
dropout (Dropout)	(None, 27, 27, 32)	0
flatten (Flatten)	(None, 23328)	0
dense (Dense)	(None, 128)	2986112
dense_1 (Dense)	(None, 5)	645

Fig. 20. Summary of model for cross-validation

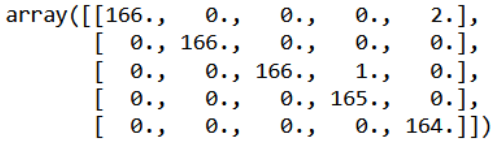

Fig. 21. K-fold confusion matrix

An accuracy of 99.64% is achieved when using K-fold. The model no longer struggles with mislabeling “hundred” and “ninety two”, and that is due to it having more data to train on. The number of data used is now much higher, with 166 cases per class, compared to 16 cases without K-fold. Training this algorithm is generally computationally expensive, and it should ideally be used with a powerful Graphical Processing Unit (GPU).

V. Conclusion and Future Work

Without cross-validation, the model was quick to overfit. Due to limited data, the model was too overconfident. However, using K-fold cross validation with a custom neural network, with Dropout layers and L1 regularization, returned the best performance when compared to other models. That could be due to many factors, and it is mostly due to the size of the dataset. Small datasets cause weights of features (pixels) to increase, and having many training and validation data helps the model perform better. Furthermore, dropout and L1 regularization make it harder for the model to increase its weights. Bigger weights increase the loss function, which the model tries to minimize. L1 regularization was chosen for its sparsity effect, as many pixels in the dataset are 0.

In a real-world scenario with more than 5000 classes and around a million videos, the selected model in this research might not be the best choice, and transfer learning might perform better, as more classes mean more complexity. Regardless, using the difference of frames algorithm to prepare data for sign language prediction through deep learning is fruitful. It is noted that with all models, a high training accuracy is observed from the first epoch. That could possibly be due to the dataset not containing enough variance and noise, making the model learn how to differentiate between the classes very early.

The shortcoming of the dynamic time warping method is that applying it to a big dataframe is time consuming. It takes approximately 3 seconds to apply dynamic time warping to a dataframe with 558 rows and 5 classes. Therefore, it is an impractical solution for real-time translation, as there would be more than 5000 classes and possibly over one million rows. Furthermore, the model is simply collecting the positions of the joints and disregarding the angles between them. Information regarding curls and bends in fingers while gesturing a sign is lost. That would pose difficulties in distinguishing signs that are gestured at roughly the same position.

In the case of not needing live and real-time translation, as in there is no concern for the speed of the prediction, then using dynamic time warping with MediaPipe’s holistic model can be a suitable option, as it is accurate even when there are few entries per class. However, the implemented model can be even further improved upon by incorporating angles between joints to the calculation, as that will capture the gesture as well and not just the position of the joints.

The difference of frames algorithm has one major flaw, it expects the background to be static. In the dataset, that was true. However, in a practical scenario, that will not always be the case. If the background is not static, then that motion will be captured with the algorithm as well. The same can be said about the

person who is performing the hand gestures, as the algorithm expects to capture only hand movement. Any movement of the body or the head will be captured by the algorithm. This can be mitigated by using the MediaPipe Holistic model to track the hands, and then draw bounding boxes that cover the range of positions taken by the hand. The difference of frames algorithm can then only be applied within the bounding boxes. Any pixel outside of the bounding boxes can be blacked out. That way, the effect of a dynamic background and noise capture is reduced.

REFERENCES

[1] A. C. Caliwag, H.-J. Hwang, S.-H. Kim, and W. Lim, *"Movement-in-a-Video Detection Scheme for Sign Language Gesture Recognition Using Neural Network," Applied Sciences*, vol. 12, no. 20. MDPI AG, p. 10542, Oct. 19, 2022. doi: 10.3390/app122010542.

[2] Guerin, G. (2022) *Sign language recognition - using MediaPipe & DTW*, *SICARA*. Available at: https://www.sicara.fr/blog-technique/sign-language-recognition-using-mediapipe.

[3] Haza, R. (2018) *Deaf and hard of hearing Emiratis hail UAE's first sign language dictionary*, *The National News*. Available at: https://www.thenationalnews.com/uae/deaf-and-hard-of-hearing-emiratis-hail-uae-s-first-sign-language-dictionary-1.732996.

[4] *MediaPipe solutions guide* (no date) *Google MediaPipe Solutions*. Available at: https://developers.google.com/mediapipe/solutions/guide.

[5] Salvador, S. and Chan, P. (2004) *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*. rep. Florida Institute of Technology. Available at: https://cs.fit.edu/~pkc/papers/tdm04.pdf (Accessed: 01 December 2023).

[6] *UAE sign language dictionary* (2023) *Zayed Higher Organization for People of Determination*. Available at: https://zho.gov.ae/en/Sign-Language-Dictionary/UAE-Sign-Language-Categories.

[7] *What are convolutional neural networks?* (no date) *IBM*. Available at: https://www.ibm.com/topics/convolutional-neural-networks.

[8] *ZHO in brief* (2023) *Zayed Higher Organization for People of Determination*. Available at: https://zho.gov.ae/en/About-ZHO/About-Us.