

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**GRADO EN INGENIERÍA DE
TECNOLOGÍAS Y SERVICIOS DE
TELECOMUNICACIÓN
TRABAJO FIN DE GRADO**

**DESIGN AND IMPLEMENTATION OF A
SYSTEM FOR MODELLING HUMAN
BEHAVIOUR USING DEEP LEARNING**

**JOSÉ MANUEL BRAVO PACHECO
2024**

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Título: Design and implementation of a system for modelling human behaviour using deep learning

Autor: D. José Manuel Bravo Pacheco

Tutor: D. Manuel Gil Martín

Ponente: D.

Departamento: Ingeniería Electrónica

MIEMBROS DEL TRIBUNAL

Presidente: D.

Vocal: D.

Secretario: D.

Suplente: D.

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de:
.....

Madrid, a de de 20...

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN
TRABAJO FIN DE GRADO**

**DESIGN AND IMPLEMENTATION OF A
SYSTEM FOR MODELLING HUMAN
BEHAVIOUR USING DEEP LEARNING**

**JOSÉ MANUEL BRAVO PACHECO
2024**

RESUMEN

Los sistemas basados en inteligencia artificial se están convirtiendo cada día en un aspecto más relevante de nuestras vidas. Sin embargo, aunque sus respuestas son cada vez más precisas, aún están lejos de una interacción natural con los usuarios. El diseño de sistemas más interactivos puede dar lugar a respuestas más personalizables y precisas, que los transformen en herramientas más útiles. Para ayudar a esta interacción hombre-máquina, resulta crucial el reconocimiento de los comportamientos humanos. Mediante la construcción de sistemas capaces de comprender las emociones o los gestos de los usuarios, estos objetivos pueden ser alcanzados. De este modo, este proyecto se centra en el diseño de dos sistemas independientes: uno para el Reconocimiento de Gestos con las Manos (HGR) y otro para el Reconocimiento de Emociones con el Habla (SER).

Para el entrenamiento de los modelos hemos utilizado dos bases de datos. Por un lado, para el reconocimiento de gestos, hemos empleado la base de datos IPN Hand. Esta base de datos contiene gestos de 13 clases distintas pensadas para el control de pantallas. La base de datos ofrece diferentes entornos e iluminaciones, ideales para entrenar modelos robustos. Por otro lado, para la clasificación de emociones, hemos empleado la base de datos perteneciente al reto de Odyssey 2024, MSP Podcast. Esta base de datos contiene muestras obtenidas de distintos podcasts públicos en internet, clasificadas en 10 emociones distintas. Al ser audios de podcasts estos contienen una mayor naturalidad que otras bases de datos usadas en esta área, siendo útiles para implementar sistemas reales.

En torno al reconocimiento gestual, proponemos el uso de puntos de referencia (o *landmarks*) de la mano, extraídos con MediaPipe, para realizar la clasificación con un modelo conocido como *Video Vision Transformer* (ViViT). La extracción de puntos característicos de la mano nos permite la reducción de información de las imágenes, irrelevante para el modelo, aspirando a sistemas menos complejos y con mejores o similares resultados. Hemos comparado diferentes configuraciones, tanto para el modelo como para el preprocesamiento de los datos, optimizando sus ajustes hasta conseguir un modelo con una tasa de acierto de un $88.10 \pm 1.91\%$, resultado similar al del estado del arte usando un modelo 6 veces más pequeño.

La segunda tarea corresponde a la participación en el reto Odyssey 2024. Para este, comparamos el uso del mismo modelo (ViViT) con una ResNet-50 y una solución multimodal compuesta por Whisper V3 y un LLM (Gemma 2B o Phi 1.5B). Primero hemos realizado la clasificación de forma aislada con solamente audio, para posteriormente integrarlo con las transcripciones, comprobando su efectividad. Además, en esta tarea hemos comprobado la adaptación del ViViT a distintos tipos de datos, tanto a los puntos de la mano como los espectrogramas de Mel, y mostramos el valor de las soluciones multimodales para obtener mejores resultados. Al final, implementamos un sistema multimodal capaz de conseguir una tasa de F1 de 0.3426 ± 0.0192 en la prueba del reto. El reto finalizó, consiguiendo al final una séptima posición con nuestro sistema. El primero de la clasificación consiguió una tasa de 0.3569 ± 0.0194 , sin embargo, esta se superpone con nuestra solución y, por tanto, pueden considerarse similares. De cualquier modo, estas bajas tasas demuestran la complejidad de la tarea a resolver.

PALABRAS CLAVE

Interacción persona-ordenador, reconocimiento de gestos con las manos, reconocimiento de emociones del habla, aprendizaje profundo, modelado del comportamiento humano, puntos de referencia de MediaPipe.

SUMMARY

Artificial intelligence-based systems are becoming an increasingly relevant aspect of our lives. However, while their responses are becoming more accurate, they are still far from a natural interaction with users. The design of a more interactive systems can lead to more customizable and accurate responses. To help this human-machine interaction, the recognition of human behaviors becomes crucial. By building systems capable of understanding emotions or gestures of users, these goals can be achieved. In this sense, throughout this project we focused into the design of two independent systems: one for Hand Gesture Recognition (HGR) and other one for Speech Emotion Recognition (SER).

We used two datasets for the training of the models. On the one hand, for gesture recognition, we used the IPN Hand database. This database contains gestures of 13 different classes designed for screen control. The database offers different environments and illuminations, ideal for training robust models. On the other hand, for the classification of emotions, we have used the database belonging to the Odyssey 2024 challenge, MSP Podcast. This database contains samples obtained from different public podcasts on the Internet and classified into 10 different emotions. As they are segments of podcasts, they are more natural than other databases used in this area, being useful to integrate real systems.

Around gesture recognition, we propose the use of hand landmarks, extracted with Mediapipe, to perform the classification with a Video Vision Transformer (ViViT) model. The extraction of landmarks from the hand allows us to reduce information from the images, irrelevant for the model, aiming for less complex systems with better or similar results. We have compared different configurations, both for the model and the preprocessing of the data, optimizing their settings until finally obtaining a model with an accuracy of $88.10 \pm 1.91\%$, which is a similar result to the state of the art using a model 6 times smaller.

The second task corresponds to participation in the Odyssey 2024 challenge. For this, we compared the use of the same model (ViViT) with a ResNet-50, and a multimodal solution composed of Whisper V3 and a LLM (Gemma 2B or Phi 1.5B). We first performed the classification in isolation with audio only, and then integrated it with the transcriptions to test its effectiveness. In addition, in this task we have tested the adaptation of ViViT to different types of data, both hand points and Mel spectrograms, and showed the value of multimodal solutions to obtain robust results. In the end, we implemented a multimodal system capable of achieving an F1 rate of 0.3426 ± 0.0192 in the challenge test. The challenge ended with our system achieving a seventh position in the end. The first in the ranking achieved a rate of 0.3569 ± 0.0194 , however, this overlaps with our solution and can therefore be considered similar. In any case, these low rates demonstrate the complexity of the task to be solved.

KEYWORDS

Human-computer interaction, Hand Gesture Recognition, Speech Emotion Recognition, deep learning, human behavior modelling, MediaPipe landmarks.

ÍNDICE DEL CONTENIDO

1. INTRODUCTION AND OBJECTIVES	7
1.1. Introduction.....	7
1.2. Objectives	7
2. RELATED WORK	8
2.1. Hand Gesture Recognition.....	8
2.2. Speech Emotion Recognition	11
3. DATASETS DESCRIPTION	13
3.1. IPN Hand dataset	13
3.2. MSP Podcast dataset.....	14
4. DATA PROCESSING.....	16
4.1. Image processing	16
4.1.1. Landmarks extraction	16
4.1.2. Mediapipe non-recognitions	17
4.1.3. Landmark normalization.....	17
4.1.4. Gesture durations.....	18
4.1.5. Frames interpolation	20
4.2. Audio processing	21
4.2.1. Mel spectrograms.....	21
4.2.2. Data selection.....	23
5. MODELS' ARCHITECTURES.....	24
5.1. Convolutional Neural Network with landmarks.....	24
5.2. Video Vision Transformer.....	24
5.3. Convolutional neural network with spectrograms	27
5.4. Audio Language Model	27
6. RESULTS AND DISCUSSION	31
6.1. Hand Gesture Recognition.....	31
6.1.1. Experimental setup	31
6.1.2. Results.....	32
6.1.3. State of the art comparison	37
6.2. Speech Emotion Recognition	38
6.2.1. Experimental setup	38
6.2.2. Results.....	39
6.2.3. Models comparison.....	42
7. CONCLUSIONS AND FUTURE WORK	43
8. BIBLIOGRAPHY	44

9. ANNEX A: ETHICAL, ECONOMIC, SOCIAL AND ENVIRONMENTAL ASPECTS	47
A.1 Introduction.....	47
A.2 Description of relevant impacts related with the project	47
A.3 Detailed analysis of some of the main impacts.....	47
A.4 Conclusions.....	47
ANNEX B: ECONOMIC BUDGET	48
ANNEX C: TOOLS USED IN THE PROJECT	49
C.1 Python	49
C.2 WandB (W&B).....	49

1. INTRODUCTION AND OBJECTIVES

1.1. INTRODUCTION

Artificial Intelligence based systems are becoming increasingly popular in recent times. The apparition of Large Language Models (LLM's) has brought the wide use of them for both personal and professional tasks. Nevertheless, these utilities are far from perfection. In most of their applications the interaction is still based on text. Getting these models into understanding different types of information leads to making them more useful and capable of doing their task in a better way.

At the same time, the increasing incorporation of smart devices in our daily lives allows for more recordings of human behavior signals, such as inertial signals, images, audio, or video. These sources of information make it possible to monitor human beings to recognize behavioral aspects, such as the activity they are performing or the emotion they are feeling. These aspects are of particular importance when developing artificial intelligence systems for human-machine interaction, making them capable of capturing information from multimedia content and enabling more personalized responses by utilizing information beyond the text.

Therefore, throughout this Final Degree Project, an artificial intelligence system for modelling those behaviors is proposed. Specifically, we focus on the recognition of hand gestures from video, and the recognition of emotions from audios. With the implementation of these two interfaces, more natural interactions can be achieved, transforming the nowadays utilities into a more complete generation.

1.2. OBJECTIVES

As mentioned at the introduction, the main objective of this Final Degree Project is to design and implement a human behavior modelling system. In this sense, signal processing and deep learning methods could be applied to recognize hand gestures captured in images or detect emotions through audios. As this is a generic objective, different objectives are set that underlies the main one.

- Selection and description of the databases to be used.
- Study and implementation of image and audio processing techniques, including hand landmarks and audio spectra extraction.
- Study and optimization of neural networks architectures for the purpose of extracting signals characteristics. More specifically, deep learning models such as Convolutional Neural Networks (CNNs), Video Visual Transformers (ViViTs) and Large Language Models (LLMs)

2. RELATED WORK

This chapter examines the current state of the art for human behavior modelling systems related to the objectives of this work, presenting various approaches as a baseline for later comparison.

2.1. HAND GESTURE RECOGNITION

Early work around Human Gesture Recognition (HGR) was based on hand-crafted algorithms that pre-processed frames to extract relevant features from the image. For example, a previous work developed by Y. Wang and R. Yang [1] proposed an algorithm for extracting hand depth from a Kinect camera and, from this depth representation, classify the gestures using a Support Vector Machine (SVM) [2]. They run experiments over two different datasets: NTU-Microsoft-Kinect-Hand and a subset of the Microsoft American Sign Language (ASL) dataset. The first dataset is conformed from 10 different subjects performing 10 instances for each one of 10 different postures, getting a total of 1,000 instances. Moreover, for making it more challenging, they extended the dataset adding 500 instances of 6 new postures. Against this dataset the results obtained cross-validation accuracy of **97.1%**. The second dataset consists of 60,000 depth and color hand images corresponding to 24 of the 26 American Sign Language letters. The results on this dataset showed a cross-validation accuracy of **96.2%**. Figure 1 shows examples of gestures for the two datasets.

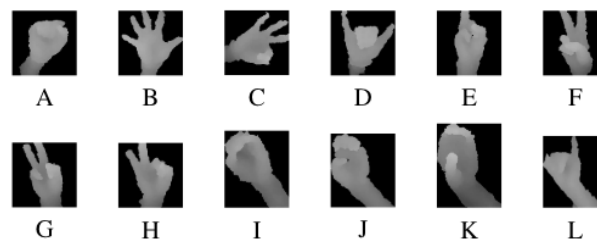


Figure 1. Different dataset hand postures. Hand posture A-D are from NTU dataset; Hand Posture E-H are from the expanded NTU dataset; Hand posture I-L are from ASL dataset. Adapted from [1].

On the other hand, since the great results of Convolutional Neural Networks (CNN) on image recognition, some approaches for gesture classification began using it. These models are based on convolutional layers, where a dot product operation is made between the layer's kernel and a part of the input tensor. The operation is shifted along all the input to generate a new feature map, which is then pooled to generate a smaller representation, and to be fed into the next layer. By repeating this process, as the size of the representations decreases, the last layers operate all the input at the same time, integrating all the information to make a final decision.

Following this line, Núñez Fernández Dennis and Kwolek B [3] proposed a method for extracting the hand features from an image using the skin color and a Gabor filter and classify them using a CNN. Specifically, the authors proposed an architecture which consisted of the use of 2 convolutional layers, with a REctified Linear Unit (RELU) as the activation function, connected by a subsampling layer. Following this, two fully connected layers are appended to complete the model that can be seen in Figure 2. The experiments were performed over a dataset of 10 different gestures designed to interact with computers. To obtain the dataset, the images were first recorded in RGB format. After that, based on manually selected wrist positions, they implemented a Neural Network Regressor to identify the wrist in the images and the background was then automatically removed. At the end, the dataset contains 6,000 grey scale hand images of a size of 38 x 38 pixels. Examples of the 10 classes can be found in Figure 3. To also test the Neural Network Regressor that detected the wrist, they experimented with images where the wrist had been detected automatically and manually. They obtained an accuracy of **95%** when using the manually labelled images and an accuracy of **90.5%** when using the automatically labelled ones.

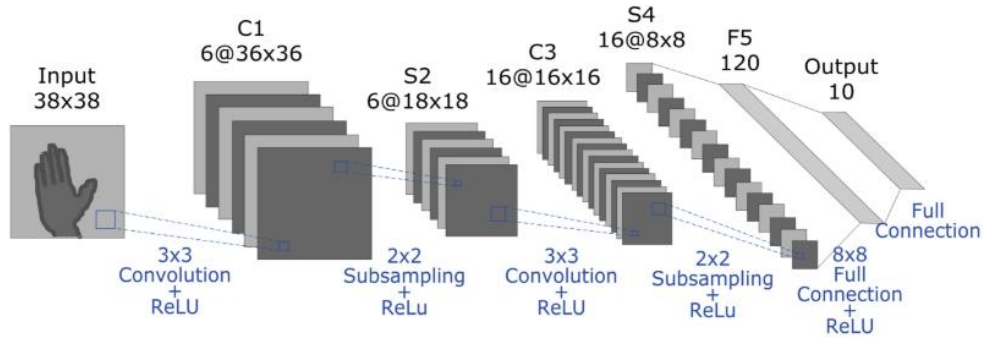


Figure 2. Architecture proposed by Núñez Fernández Dennis and Kwolek B. Adapted from [3].



Figure 3. Hand postures of the " Hand Posture Recognition Using Convolutional Neural Network " dataset. Reprinted from [3].

The gestures of the previous datasets are considered static. Their classification does not depend on the movement of the hand, just on the static position of it. On dynamic gestures, the authors of IPN Hand dataset [4] proposed the use of three different 3D convolutional neural networks (3D-CNN) for the classification of isolated dynamic gestures. The dataset will be described in detail in the following chapter as it is the one that will be used in this project. These 3D-CNN networks add the temporal dimension to the kernel to learn not only temporal but also temporal patterns. To be precise, the dataset was tested on the 3D-CNN models of C3D [5], as well as on the 3D versions [6] of Resnet-50 [7] and ResNeXt-101 [8]. Table 1 shows that the ResNeXt-101 with RGB-Flow obtained 86.32% of accuracy, using an architecture with 46.56M of parameters. This will be the baseline used for comparison for the Hand Gesture Recognition task.

Model	Modality	Results	Parameters
C3D	RGB	77.75%	50.75M
ResNeXt-101	RGB	83.59%	47.51M
ResNeXt-101	RGB-Flow	86.32%	47.56M
ResNeXt-101	RGB-Seg	84.77%	47.56M
ResNet-50	RGB	73.10%	46.25M
ResNet-50	RGB-Flow	74.65%	46.27M
ResNet-50	RGB-Seg	75.11%	46.27M

Table 1. Results of isolated HGR Task on IPN Hand. Adapted from [4].

Recent methods also search for multimodality approaches. These solutions opt for fusing information from different representations, mixing the relevant data of each source for the classification. Nevertheless, there are multiple methods of combining the information. While classic methods added or averaged different sources, modern solutions apply neural models do this. The use of convolutional-transformer blocks was proposed on [9]. Specifically, they use what they call *Efficient Convolutional Self-Attention* (ECSA) blocks, which consists in a fusion between convolutional and transformer's self-attention layers. They applied the model with the IPN Hand dataset, obtaining the results shown on Table 2.

Model	Modality	Results	Parameters
Multimodal ECSA	R+F	$91.4 \pm 1.66 \%$	27.2M
Multimodal ECSA	R+S	$90.8 \pm 1.71 \%$	27.2M

Table 2. Results of multimodal model proposed on [9]. R-RGB, F-optical flows, S-segmentation masks.

Nevertheless, all these solutions still rely on the use of optical flows and segmentation masks, that are obtained using handcrafted algorithms. In this sense, another previous work [10], developed a method for classifying static poses through landmarks. Specifically, they used two datasets: Multi-modal Leap Motion, which contains 16 different poses, and Tiny Hand Gesture Recognition Dataset, containing 7 different poses. Their technic relies on the use of MediaPipe for extracting the hand landmarks from the RGB frames as can be seen in Figure 4. After obtaining that representation, they applied a CNN model, seen in Figure 5, to classify them, obtaining an accuracy of **97.25%** with Multimodal Leap Motion dataset, and **98.23%** with Tiny Hand Dataset.

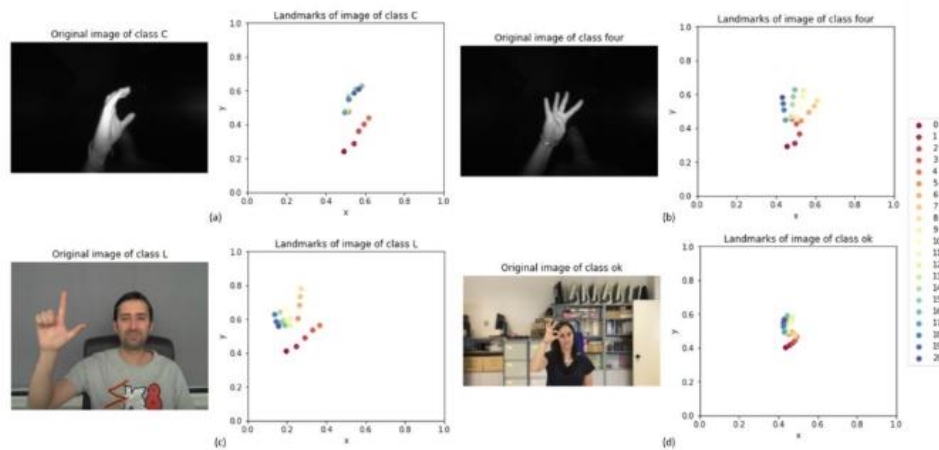


Figure 4. Landmark extraction from the datasets. (a) and (b) from Multi-modal Leap Motion Dataset and (c) and (d) from Tiny Hand Gesture Recognition Dataset. Reprinted from [10].

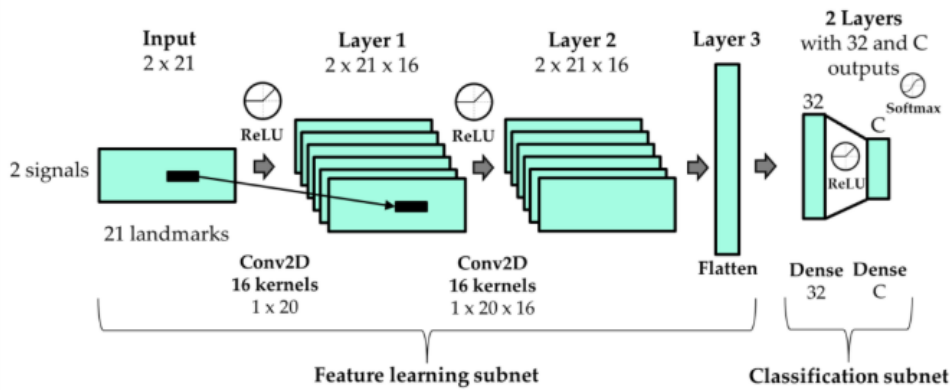


Figure 5. CNN model for classifying gestures from landmarks. Reprinted from [10].

In summary, current methods for classifying dynamic gestures still rely on handcrafted representations of the hands. The creation of a model capable of performing a full classification on its own is still in progress, as the advanced developments still focus on classifying static poses. Following this line, a model for making the classification of not only static but also dynamic gestures from just the RGB images is proposed.

2.2.SPEECH EMOTION RECOGNITION

Emotional recognition task is one of the most relevant features for upgrading communication between humans and computers. It is found to be useful in multiple fields. For example, since the Coronavirus Pandemic (COVID-19) in 2020, the learning process took a turn into being more telematic, this means, the use of online tools to make the teaching process via internet. For example, Kim and Kim [11] stated that e-learning can help the students to retain knowledge with less effort than classical methods, but more interactive systems are needed for this to succeed. For example, the implementation of emotion recognition can be useful for a teacher to know if the students are understanding the material or not.

In the 19th century, Darwin [12] studied the emotions of humans and animals. At his book, he claimed that emotional content can be seen through the face, the voice tears and postures of a person. Many systems that aim to classify emotions are based on facial gestures. In this sense, Chowdary, Nguyen and Hemanth [13] studied the application of transfer learning techniques to classify face images between 7 different emotions: anger, contempt, disgust, fear, happy, sadness, surprise. Examples of the gestures, including an additional one that was not used (neutral), can be seen in Figure 6. As mentioned above, they used transfer learning techniques, which consists of reusing a model that was trained for a more general task and specializing it for the desired task. This is particularly useful when the training set is small, as reusing other information often helps to get better results. They tested four different CNN models pre-trained on the ImageNet dataset and obtained the results shown in Table 3.

Model	Results	Parameters
VGG19	96 \pm 3.16 %	20.56M
Resnet50	97 \pm 2.75 %	25.69M
MobileNet	98 \pm 2.26 %	4.29M
Inception V3	94 \pm 3.83 %	23.91M

Table 3. Results obtained on Chowdary, Nguyen and Hemanth's experiments. Adapted from [13].



Figure 6. Examples of images to the emotion recognition task done by facial expressions. The classes shown are: (a) Disgust, (b) Happy, (c) Surprise, (d) Fear, (e) Angry, (f) Contempt, (g) Sadness, (h) Neutral - not included in experiments. Reprinted from [14].

On the other hand, the Speech Emotional Recognition (SER) is less studied. The first problem that appeared is how the information of the audios should be represented. Gaurab, Emily and Allison [15] studied different possibilities. In their study, they experimented with 5 different representations: Mel spectrogram [16], Mel Frequency Cepstral Coefficients (MFCC) [17], spectral contrast [18], Tonnetz representation [19], and chromagram [20]. To evaluate them, they obtained the 5 representations for the RAVDESS dataset. After that, they tested these input formats on different models: a 5-layer CNN shown in Figure 7, a Vision Transformer (ViT) [21], a ResNet-50 [22], and a DenseNet-201 [23]. From these experiments, they found that the ones made with Mel-Spectrograms, MFCC and Tonnetz representations obtained better results consistently, concluding them as the best representations for this task.



In January 2024, the University of Texas at Dallas published the Odyssey 2024 Challenge [25]. This challenge proposed two different tasks around emotion recognition: categorical emotion recognition and emotional attribute prediction. The MSP-Podcast dataset was used for both tasks, which consists of speech segments obtained from audio-sharing websites. Our research group participated in the first task of the challenge, submitting one of the approaches that are explained in this project. The challenge ended up with a top F1 macro score of **0.3569**, showing the complexity of the task.

3. DATASETS DESCRIPTION

This chapter deeps into the description of the datasets used for training the systems that appears in the project.

3.1. IPN HAND DATASET

The IPN Hand [4] database is intended to be a reliable dataset for benchmarking deep learning models in the task of Hand Gesture Recognition (HGR). It consists of 4,218 gesture instances classified into 13 different classes.

The participants were instructed to record the videos using their personal computer or laptop. This resulted in variations in distance from the screen, lighting, and background. Additionally, three scenarios were provided for those unable to record on their own. Finally, each one of the 50 participants were given a list with random chosen gestures, some of them repeated, to record with 3, also random, breaks. The subjects had to record themselves performing the instructed gestures of the list, all in a single video. All this gives a great variation to the dataset, making it an excellent choice for training and implementing robust and reliable systems.

The gestures in the dataset, shown in Figure 8, were thought for controlling virtual screens. In this sense, two types of gestures can be found: ones for the control of the pointer, and others for taking actions. The gestures can also be classified into static and dynamic. Static gestures are those that do not depend on temporal information to be detected, such as pointing the screen. Dynamic ones carry temporal information, such as zooming in or zooming out, and must be interpreted within their context. There is an additional gesture, which represents the *non-gesture* class, whose instances were ignored.

All videos share a resolution of 640x480 and 30fps framerate. The authors also provide all compressed frames at a resolution of 320x240 in three different formats: RGB, optical flow, and segmentation frames. Some examples of these formats can be seen in Figure 9. The frames are also annotated with the label, as well as the first and last frame of each gesture, allowing the classification of the gestures isolated in the video.

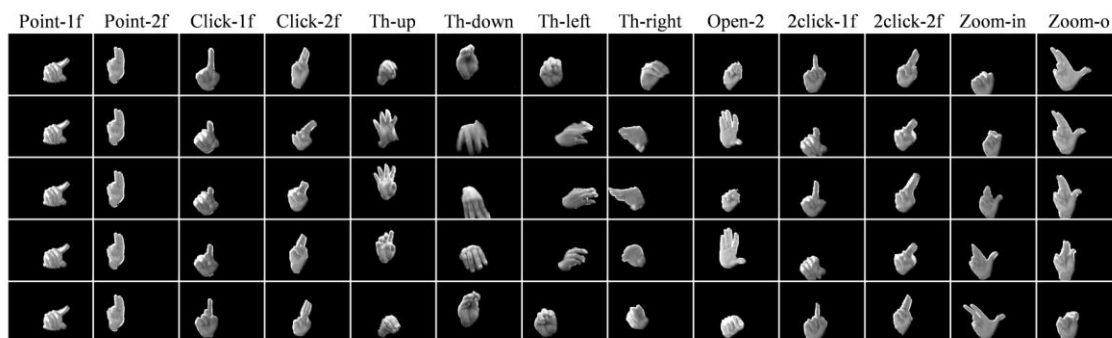


Figure 8. Example of all the gestures of the database, blended with segmentation masks for visualization. Adapted from [4].

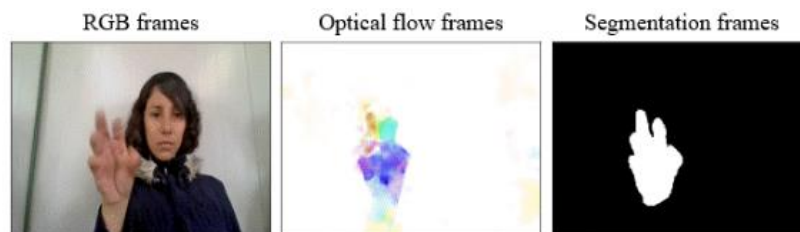


Figure 9. Different representations of the dataset frames. Reprinted from [4].

Id	Gesture	Instances	Mean duration (std) [frames]
0	No gesture	1,431	147 (133)
1	Pointing with one finger	1,010	219 (67)
2	Pointing with two fingers	1,007	224 (69)
3	Click with one finger	200	56 (29)
4	Click with two fingers	200	60 (43)
5	Throw up	200	62 (25)
6	Throw down	201	65 (28)
7	Throw left	200	66 (27)
8	Throw right	200	64 (28)
9	Open twice	200	76 (31)
10	Double click with one finger	200	68 (28)
11	Double click with two fingers	200	70 (30)
12	Zoom in	200	65 (29)
13	Zoom out	200	64 (28)

Table 4. Statistics of the gestures. Adapted from [4].

3.2.MSP PODCAST DATASET

The MSP Podcast dataset [26] consists of 87,934 audio instances from 454 different speakers. These audios belong to one of 10 possible classes: angry, sad, happy, surprise, fear, disgust, contempt, neutral, other and none.


Some datasets opt for using actors for the recordings, which produces overreacted emotions, that do not correspond with the natural daily behavior of people. Others try to avoid this by using natural interactions. But, since in normal day conversations the most common emotion is the neutral one, these datasets are often unbalanced. Moreover, the protocol used in the recordings could lead to positive or negative emotions depending on the topic of the conversation. Another important aspect is that many other datasets in emotion recognition are limited in number of speakers. This is important to obtain a system capable of generalizing learning, as increasing it also means increasing diversity, which allows for more robust models.


This MSP Podcast dataset aims to overcome these problems. The audios were selected to obtain a balanced emotional set and, as they are extracted from podcasts, they still retain the naturalness. Also, as being able to select from every public podcast, the dataset contains a great number of different speakers.


In this dataset, each audio is a speech fragment of only one person with a duration between 2.75 and 11 seconds. In addition, a Signal Noise Ratio (SNR) filter is passed through them to ensure that the audio is of high quality. The labelling process was carried out by different annotators, who were constantly evaluated to ensure they did not get tired of doing the task and to achieve a more reliable result. These annotators filled out the questionnaire of Figure 10 for each instance. In it, they were asked to select one primary emotion from a list, a secondary emotion and an evaluation of the valence, arousal and dominance presented in the audio. Then, the majority primary emotion label of the annotators is selected for the audio. If no majority is achieved, the label will be “None”.


Nevertheless, the dataset is still unbalanced. As can be seen in Table 5, “Neutral” appears the most, while emotions like “Fear” hardly appear at all. This could lead to a problem when training the model, as it will tend to classify the audios as the emotions that appear the most.


Please rate the negative vs. positive aspect of the video
Click on the image that best fits the video.



☐
(Very negative)



☐
(negative)


☐
(somewhat negative)


☐
(neutral)



☐
(somewhat positive)



☐
(positive)



☐
(Very positive)


(a) Valence


Please rate the calm vs. excited aspect of the video
Click on the image that best fits the video.



☐
(Very calm)



☐
(calm)


☐
(somewhat calm)


☐
(neutral)



☐
(somewhat active)



☐
(active)



☐
(Very active)


(b) Arousal


Please rate the weak vs. strong aspect of the video
Click on the image that best fits the video.



☐
(Very weak)



☐
(weak)


☐
(somewhat weak)


☐
(neutral)


☐
(somewhat strong)


☐
(strong)


☐
(Very strong)

(c) Dominance

Is any of these emotions the primary emotion in the audio? If not, select **Other** and specify the emotion.

☐ Angry ☐ Sad ☐ Happy ☐ Surprise ☐ Fear ☐ Disgust ☐ Contempt ☐ Neutral ☐ Other

(d) Primary emotion

Please pick all the emotional classes that you perceived in the audio (include the primary emotions selected in previous question)

☐ Angry ☐ Sad ☐ Happy ☐ Amused ☐ Neutral

☐ Frustrated ☐ Depressed ☐ Surprise ☐ Concerned

☐ Disgust ☐ Disappointed ☐ Excited ☐ Confused

☐ Annoyed ☐ Fear ☐ Contempt ☐ Other

(e) Secondary emotion

Figure 10. Questionnaire of MSP Podcast dataset. Reprinted from [26].

Emotion Label	Emotion	Instances	Mean duration (std) [s]
N	Neutral	30,773	5.68 (2.34)
X	None	17,722	5.80 (2.37)
H	Happy	16,780	5.87 (2.44)
A	Angry	5,466	5.99 (2.39)
S	Sad	4,983	5.75 (2.31)
C	Contempt	3,766	6.15 (2.37)
U	Surprise	3,626	5.67 (2.46)
D	Disgust	1,912	6.16 (2.38)
O	Other	1,726	5.95 (2.32)
F	Fear	1,421	5.56 (2.47)

Table 5. MSP Podcast distribution.

4. DATA PROCESSING

The methods used for the data analysis and preprocessing are exposed along this chapter. As having video images and audio data, different approaches have been done.

4.1. IMAGE PROCESSING

The section deeps into the steps that take part from loading the images to the final representation that is used to feed the model. Also, several preprocessing alternatives that were studied are presented in this chapter for a posterior analysis in the experimental section (Sec. 6.1.2).

4.1.1. LANDMARKS EXTRACTION

To reduce the complexity of the model and help it to focus on the important information, the hand landmarks are extracted from the RGB frames. As can be seen in Figure 11, these landmarks consist of 21 key points of the hand that allows for the fully representation of it. The extraction is made with the MediaPipe hand detection utility [27], which also uses machine learning for detecting those points from an image.

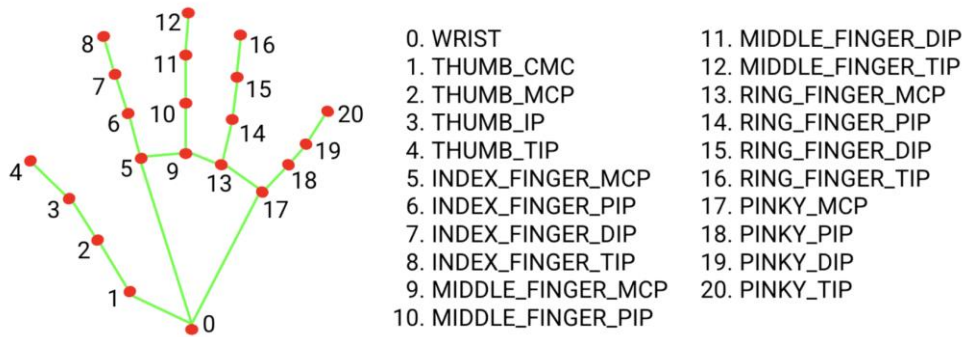


Figure 11. Landmarks points extracted from the frames. Adapted from [27].

Once MediaPipe was installed (version 0.10.8), the frames are passed one by one to the utility, which extracts the landmarks detected in the image as Figure 12 shows. The library returns a list with the 21 hand points, which is flattened to generate a vector of 42 elements ($x_0, y_0, x_1, y_1, \dots, x_{20}, y_{20}$). Finally, these vectors are stacked into a single $L \times 42$ tensor, where L is the length of the video in frames, and stored in a csv file.

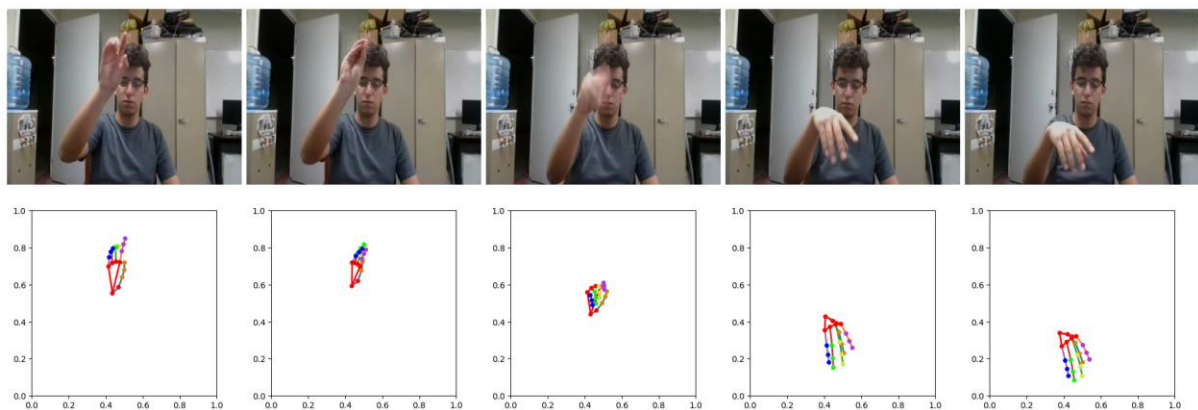


Figure 12. Example of hand landmarks extraction from images.

4.1.2. MEDIAPIPE NON-RECOGNITIONS

The extraction process made by MediaPipe is not optimal. In fact, 10.61% of the total frames processed cannot be recognized. Figure 13 shows the percentage of instances from each class where the landmarks were not correctly detected by MediaPipe. Among other causes, these errors are produced because of the blur of the hand or the subject going out of camera shot. Figure 14 shows some examples from the dataset where the landmarks were not extracted. This issue becomes a critical problem for fast gestures such as clicking, as losing one frame may result in losing lot of information.

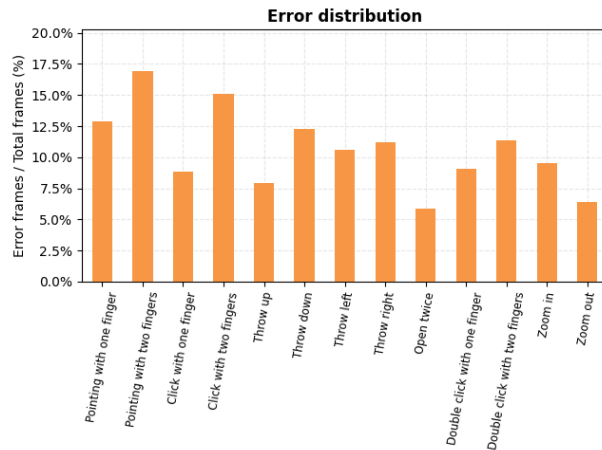


Figure 13. Distribution of MediaPipe detection errors per class.



Figure 14. Errors examples.

Two different methods were proposed to overcome this issue: replacing the errors with zeros or a linear interpolation between the last frame detected and the first following one detected again. The final system implemented uses the interpolation method as it obtains a more linear movement of the hand. Although this does not completely solve the problem, it helps to obtain a better and smoother representation of the gesture instances.

4.1.3. LANDMARK NORMALIZATION

Since the position of the hand in the frame is unrelated to the classification of the gestures, a landmark normalization is established. This method also helps to reduce irrelevant information and biases that can be found in the data. Two different approaches were proposed: *0 landmark* normalization, and *first frame* normalization. An example of these technics is shown in Figure 15.

The **0 landmark** (OL) normalization aims to keep the wrist landmark of each gesture at the origin. This is made by subtracting the coordinates of the wrist landmark of each frame to the rest of landmarks of that same frame. As after this normalization, its position is fixed at the origin. This approach implies the loss of the translational information of the hand along the gesture, however, this information might be irrelevant for the classification, and removing it may lead to better results.

The **first frame** (FF) normalization entails the manual positioning of the hand in such a way that the wrist landmark of the initial frame is situated at the origin. To achieve this, the coordinates of the wrist landmark of the first frame of each gesture are subtracted from all the frames of that same gesture. It is crucial to note that, as the classification is conducted on a per-gesture basis, the normalization process must also be conducted on a per-gesture basis and not on the full video.

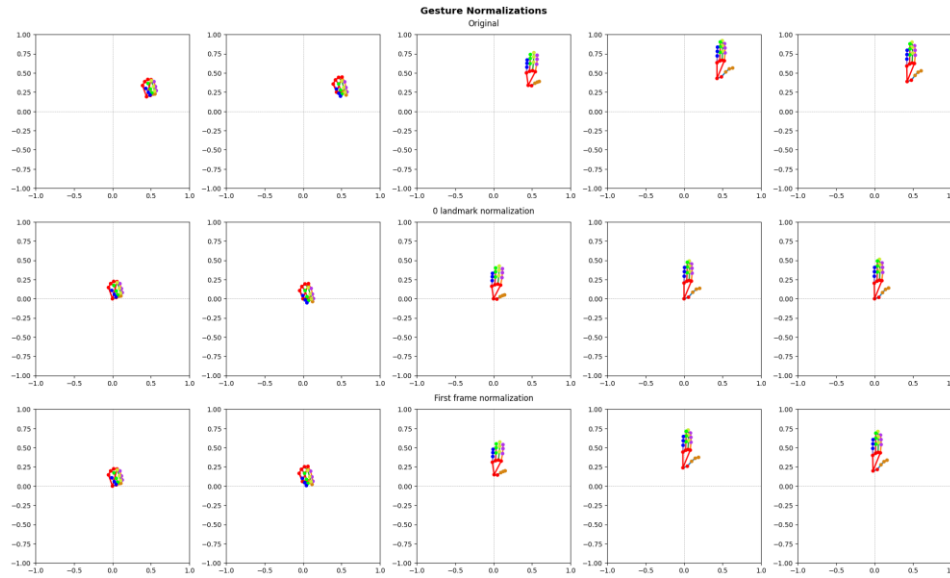


Figure 15. Gesture normalizations for a throw up gesture instance.

4.1.4. GESTURE DURATIONS.

All tensors fed into the model must be of equal size and, hence, the videos must be of same duration. As this is not the case with the dataset, as shown in the histogram of Figure 16, some pre-processing is required. Figure 17 shows that the longer videos in the dataset belong to static classes, which are independent of time and can be detected within a shorter time interval. Additionally, reducing the video length decreases the model's complexity. From all this, we experimented with four different options: full videos, 150 frames (5s), 100 frames (3.33s) and 50 frames (1.67s) and, finally, a fixed duration of 150 frames was chosen. This duration allows most of the videos of dynamic gestures to fit without being trimmed, precisely, 71.38% of the videos of them are shorter than this duration. Videos longer than this are trimmed, and shorter ones are padded.

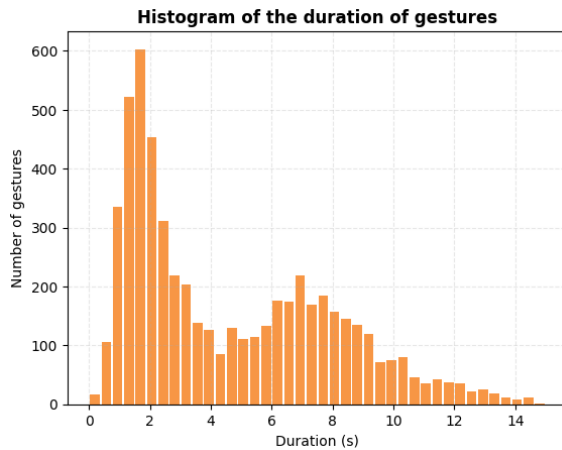


Figure 16. Distribution of gesture durations.

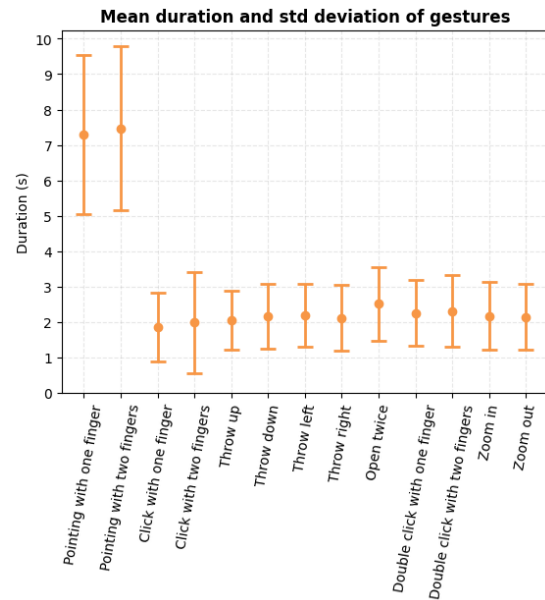


Figure 17. Gesture duration mean by class.

GESTURE ACTIVITY DETECTOR (GAD)

Selecting only the first N frames of the video can result in losing the gesture if it is not located at the beginning. To avoid these losses, a Gesture Activity Detector (GAD) has been developed. This system aims to detect the exact moment where a gesture is made, ensuring it is not lost when clipping.

Its operation is based on measuring the energy of the movement. Specifically, the energy between a frame k and the previous one is defined as the accumulate distance, considering all the landmarks:

$$E_k = \sum_{i=0}^{20} \sqrt{(x_i^k - x_i^{k-1})^2 + (y_i^k - y_i^{k-1})^2}$$

where k is the number of the frame and i is the number of the landmark point. By making this operation between every pair of frames of the sequence, a representation of the energy flow along the gesture can be obtained. For example, Figure 12 shows the energy obtained for one of the gestures in each class.

This energy graph can be used for selecting the window. When the gesture occurs, the energy should be maximum as is when the movement happens. Nevertheless, there are not obvious peaks so, instead of taking where the peak happens, the selected window is the one containing the more energy summed up.

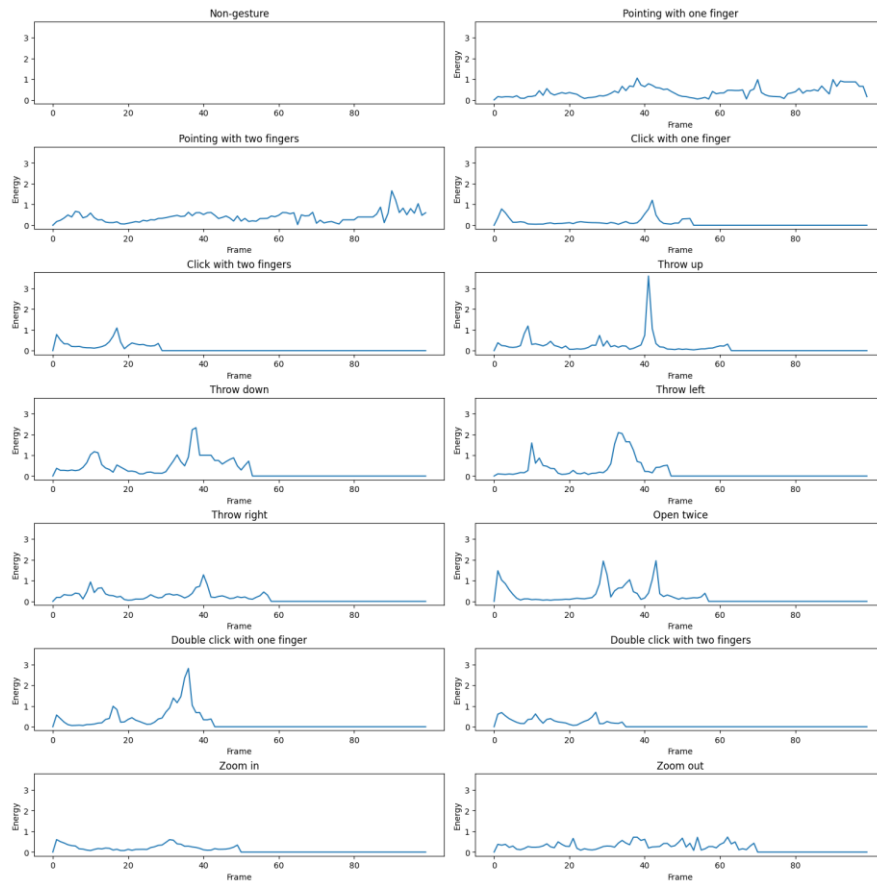


Figure 18. Energy between landmarks for dataset gestures. The gestures were zero-padded to 100 frames to fix its length.

GESTURES PADDING

The GAD helps to clip large videos, nevertheless, padding is necessary for shorter videos. Three different options were proposed: zero-padding, edge-padding and loop-padding.

The zero-padding and edge-padding process is very similar. It consists of adding a fixed value at the end of the sequence until the desired length. The difference between them is the value: while on zero-padding the value added is a zero, in edge-padding the last value of the sequence is added.

For the loop method the sequence is written again at the end of the first one. This process is repeated until the desired length. The idea behind that is to give the model more opportunities to recognize the gesture. However, the dataset has gestures that are repetitions of others, specifically, this happens between *click* and *double click* instances. Looping the sequence may end up in mixing both gestures, causing more problems in their classification as is shown in section 6.1.2. Hence, for simplicity reasons, we opted for using zero-padding to extend shorter videos.

4.1.5. FRAMES INTERPOLATION

The videos of the dataset are recorded at 30 fps. This may be small when recognizing fast gestures as the gesture itself is represented by a small fraction of the total frames inserted to the model. To account this, a linear interpolation of the landmarks was done. With this, the original framerate can be artificially incremented to extend the important part. We did experiments with a framerate of 60 fps and 90 fps. Nevertheless, the interpolation did not reflect an increase of accuracy, so the original framerate was used for the final model.

4.2.AUDIO PROCESSING

4.2.1. MEL SPECTROGRAMS

When recording audio signals, the microphone stores samples of the pressure it detects. The rate at which this sampling occurs is known as sampling rate. This process stores a representation of the signal known as a waveform, which can be seen in Figure 19. However, extracting relevant information from such representations is complex. To overcome this, the problem is taken to the frequency domain.

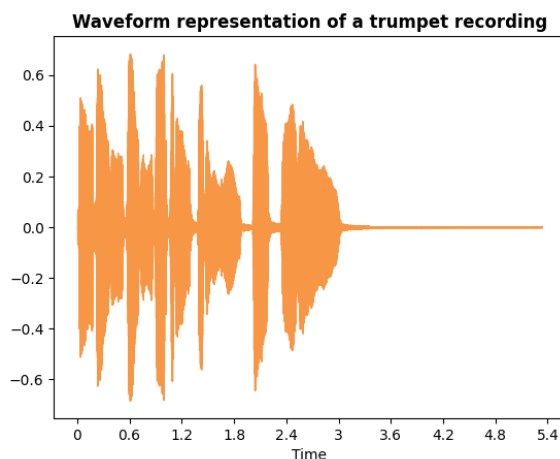


Figure 19. Waveform of a trumpet recording.

By applying the Fourier transform to the signal, a representation of the frequencies and its amplitudes can be obtained. An example of this can be seen in Figure 20. This representation gives much more relevant information of the characteristics of the signal. Still, this representation hides the evolution of the signal along the time, which is important to analyze how the emotional content varies along an audio. Another characterization which can represent this information must be achieved.

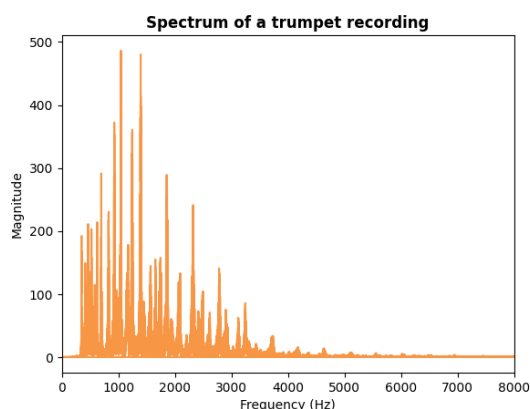


Figure 20. Spectrum of the trumpet recording.

A representation of that variation of the frequencies and its amplitude over time can be obtained by doing the Fourier transformations in small windows and concatenating them. For a better resolution, these windows are usually overlapped. Also, since humans notice the volume in decibels, the amplitudes are represented in that unit. This representation is called spectrogram and can be seen in Figure 21. In these representations, the x axis represents the time evolution, the y axis represents the frequencies, and the color of the points is the amplitude of the signal.

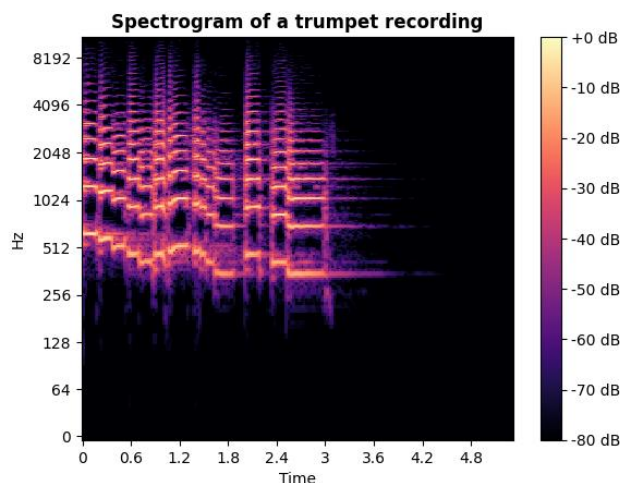


Figure 21. Spectrogram of the trumpet recording

However, as the human ear does not perceive frequencies in a linear scale, this representation can be improved. In fact, low frequencies are easier to distinguish than high ones. For example, we are barely able to distinguish between 10,000 and 10,500 Hz and, at the same time, we can easily distinguish between 500 and 1000 Hz, even though the distance is the same. This is where the Mel frequency concept comes in. In essence, it applies the non-linear transformation shown on Figure 22 to the frequency axis, giving more resolution at lower frequencies and less at higher ones. This can be seen on the Mel spectrogram of Figure 23.

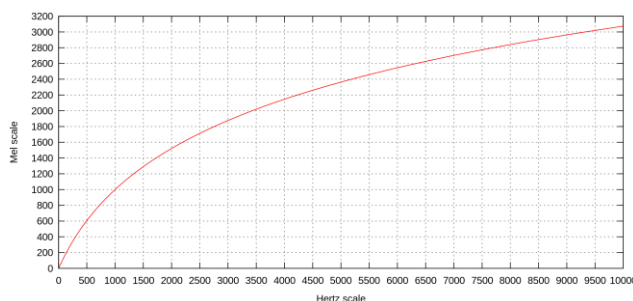


Figure 22. Plot of Mel scale versus hertz scale. Reprinted from [28].

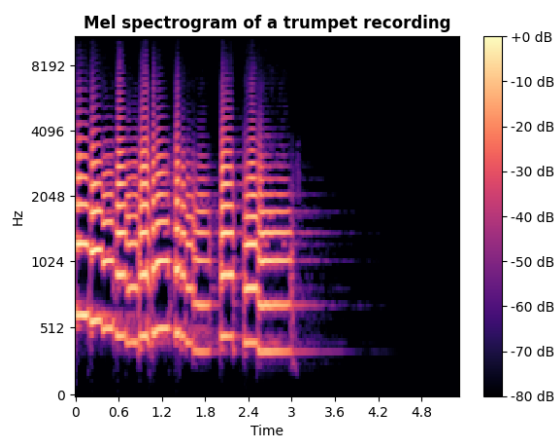


Figure 23. Mel spectrogram of the trumpet recording.

Mel spectrogram was chosen as input representation for the audio samples. There are several parameters when calculating them: number of filters, which establishes the subdivisions in Mel scale; hop length, which is the length of the sub-windows of the waveform on which the Fourier transformation is performed; Fourier frequency resolution, which affects the resolution of the Fourier transformation; and chunk length, which is used to trim or pad longer or shorter audio sequences. For the extraction of features from the audios, in some experiments we used Whisper V3 model, which is a pre-trained model for automatic speech recognition. As this model is already pretrained, it needs the input in a specific format, which is shown in Table 6. For a better comparison between different models, the same spectrograms, generated with those settings, are used in every experiment.

Parameter	Setting
Number of filters	128
Hop length	160
Fourier frequency resolution	400
Chunk length	30

Table 6. Settings of the Mel Spectrogram.

4.2.2. DATA SELECTION

The main emotion labelled on each audio comes from the annotation labelling. As explain before, this main emotion is the one that repeats the most among the labels of the annotators. When all appears in the same proportion, the emotion is set as “None”. Since the labels “Other” and “None” are not considered in the challenge, their audios were also not considered neither for training, validating or testing. Thus, every instance which main emotion is one of these two is eliminated from the entire dataset.

On the other hand, there are several audios where there is no clear agreement. For example, if the annotations of the workers are [happy, happy, surprise, other, none], the main emotion will be happy but only 40% of the workers agreed. To obtain a solid training set, we only kept the audios where at least the 50% of annotators agreed with an emotion. By making this, we ensure a robust training set, which is essential for building robust systems. Nevertheless, this selection is done after splitting the dataset and only on the train subset, as modifying the other subsets would be hiding these challenging cases. The distribution of the dataset after this selection can be seen in Figure 24. After this selection, the number of audio instances selected is 61,168 from the 88,175 that was the total.

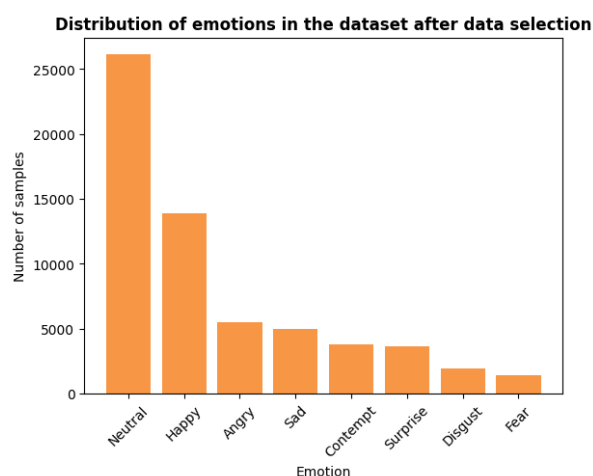


Figure 24. Distribution of data after removing “None”, “Others” and without agreement.

5. MODELS' ARCHITECTURES

We propose two different approaches for solving the HGR task, one based on CNN and a second one based on transformer networks. Moreover, we also present a multimodal approach using LLMs for solving the SER task.

5.1. CONVOLUTIONAL NEURAL NETWORK WITH LANDMARKS

The first model proposed continues with the use of CNN but, instead using the frames as the baseline models, the hand landmarks are used. As these architectures were used on the original work, this model aims to validate the use of the landmarks instead of the raw images to make the gesture classification.

Because of this, a simple CNN was built. It consists of a single convolutional layer with a $(1 \times 20 \times 16)$ kernel and 16 filters. Then, the output of this layer is flattened and fed into a Dense Network that outputs the final classification. The full model can be seen in Figure 25.

As these models iterate through patches of the input, they extract patterns from the data without considering where this feature is located. This makes the CNN a great option for image recognition. However, this becomes a problem when working with video, as the order at which frames appear becomes relevant for the classification.

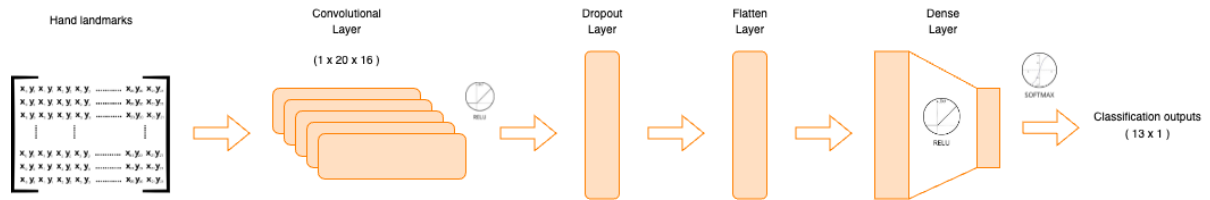


Figure 25. CNN architecture.

The hyperparameters of the model have been optimized for training the network. The batch size should be as large as possible for a better gradient updating, but not too much or it does not fit into memory, after doing some experiments we ended up using a size of 100. As optimizer, we used Adam with a learning rate of $1e-4$. Finally, we tested with 50 epochs of duration. The summary of the chosen hyperparameters can be found on Table 7.

Hyperparameter	Configuration
Batch size	100
Learning rate	$1e-4$
Optimizer	Adam
Epochs	50

Table 7. Hyperparameters for the training of the CNN.

5.2. VIDEO VISION TRANSFORMER

The second model proposed for the task is based on transformers [24]. These models are becoming the state of art in most of the applications, becoming a strong option for this classification. In a general view, these models can detect the important parts of the input sequence, giving greater attention to them even if the sequence is large.

Following the success of the CNN in video classification, our approach is based on Video Vision Transformers (ViViT) [29]. This model combines both the convolutional idea, which helps to find patterns in the data, with the transformer one, that keeps track of the important ones. They are an extension for the Vision Transformers (ViT). In fact, the main difference between them is that ViViT extends the concept of spatial patterns to spatio-temporal ones. Upon initial publication, the authors of ViViT outlined four distinct implementations: spatio-temporal attention, factorized encoder, factorized

attention, and factorized dot-product attention. Although the underlying concept remains consistent across all four, the level of complexity varies. For simplicity purposes, the experiments focus solely on the *spatio-temporal* attention variant.

On *spatio-temporal* attention models, the input data is first passed through a convolutional layer, which processes the data by dividing it into tubelets and extracting new *spatio-temporal* representations or *tokens* like in Figure 26. The dimension of the tokens is defined by the number of filters that these layers have, and it is also configurable. After this process, the tokens are flattened building a new tensor of shape (L, d_k) , where L is the number of tubelets and d_k is the number of filters passed.

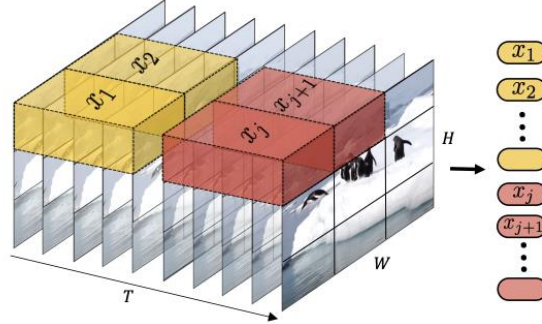


Figure 26. Tubelet embedding for spatio-temporal pattern extraction. Reprinted from [29].

When using the model on HGR task, this layer needs to be modified, while keeping the same concept, for using it with the landmarks. Instead of a 3D convolution, the model applies a 2D convolution to the input, where one dimension represents the landmarks coordinates and the other represents time as can be seen in Figure 27. The tubelet size can be configured: making them smaller increases the resolution for the transformer as more tokens are produced, but this also increases the complexity and computations, so an equilibrium must be reached. The number of filters applied for the convolution, known as projection dimensions, can also be modified. Similarly to the tubelet size, increasing the number produces more representations of the input but also increases the complexity of the model.

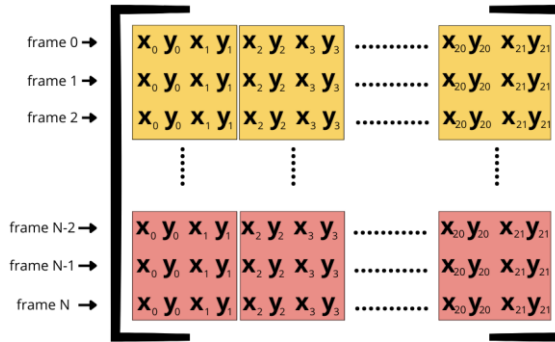


Figure 27. Tubelets modified for the HGR task.

On the other side, when using it on SER the Mel spectrograms are reshaped. Each time-unit spectrogram is stacked along a new dimension. By doing this, the input is a tensor of 3 dimensions: time; height, which corresponds to Mel frequencies; and width, which is always one. This process is like understanding each column of the spectrogram as a frame and stack all frames one over the other. A representation of this transformation can be seen in Figure 28. With this structure, the tubelets are applied in the same way as the original one does.

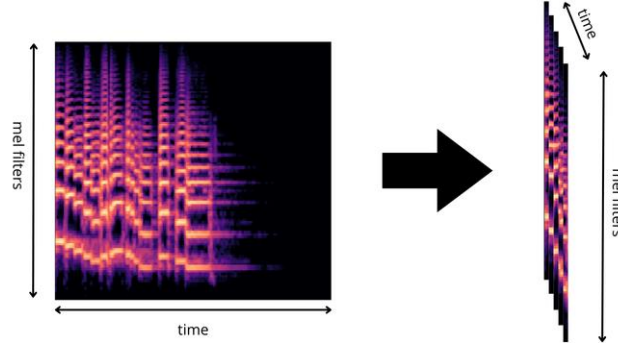


Figure 28. Transformation of the spectrograms structure.

Once the tokens are obtained, the transformer encoder is fed with them as a sequence. Its architecture can be seen in Figure 29. This encoder considers all pairwise interactions between the processed tubelets, which contrasts with classic CNN approaches where the receptive field grows linearly with the number of layers. In this process, the protagonist is the *multi-head self-attention layer* [24]. In this layer, each head performs the attention operation, defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} \quad (1)$$

Where \mathbf{Q} are the queries, \mathbf{K} the keys and \mathbf{V} the values. All three are linear projections of the input tensor. This operation calculates a score between every pair of the input sequence, allowing the model to recognize the *tokens* that are important for the classification.

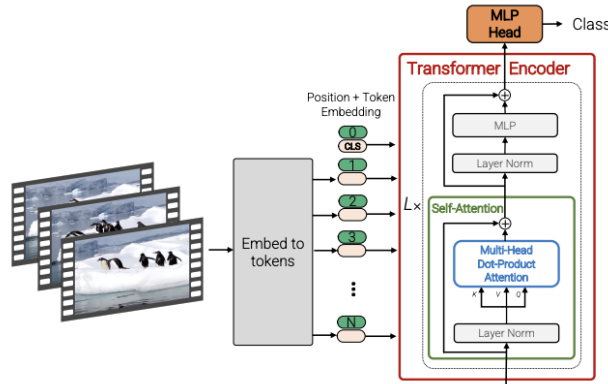


Figure 29. Transformer encoder diagram. Adapted from [29].

At the end, the MLP Head outputs P different values between 0 and 1, representing the probability of the input belonging to each of the P classes for classification.

As a summary, the hyperparameters of the final models for each task can be seen in Table 8.

Hyperparameter	Configuration for HGR	Configuration for SER
Batch size	100	4
Peak learning rate	1e - 3	1e - 4
Optimizer	Adam	Adam
Projection dim	256	64
Tubelet size	(5, 42)	(2, 8, 1)
Num layers	2	8
Num heads	4	8

Table 8. Summary of ViViT hyperparameters.

5.3. CONVOLUTIONAL NEURAL NETWORK WITH SPECTROGRAMS

Similarly to model of section 5.1, we propose a CNN model to obtain a first baseline result. In this case, we use a Residual Network (ResNet) model [7]. This model comprises a series of skip connection and bottleneck convolution blocks. The bottleneck blocks, as can be seen in Figure 30, consist of three convolutional layers with ReLU activation. The first layer compresses the dimensions of the data received, the second one extracts spatial patterns, and the last one restores the dimensionality. On the other hand, the skip connection conforms the residual part of the model, which allows to representations learned in previous blocks to propagate through the network. For our case, we use the ResNet-50 version, which concatenates 50 of these bottleneck blocks. A summary of the chosen hyperparameters for the training of this model can be found on Table 9.

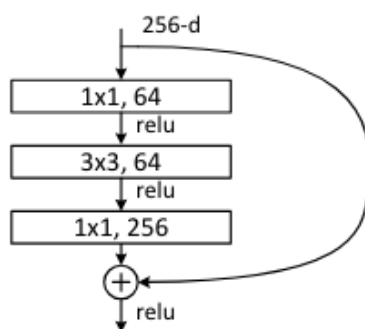


Figure 30. Bottleneck blocks for ResNet models. Reprinted from [7].

Hyperparameter	Configuration
Batch size	64
Learning rate	$1e - 3$
Optimizer	Adam
Epochs	50

Table 9. Hyperparameters for the training of the ResNet model.

5.4. AUDIO LANGUAGE MODEL

For the SER task we present a multimodal Audio Language Model (ALM) [30]. This model innovates by proposing a balance between text transcriptions and audio, by integrating Large Language Models (LLM) with audio features. By this integration, we extract relevant features from both sources of information, achieving better results than only using one source.

Large Language Models are transformer-based architectures pre-trained on a huge amount of data. Specifically, these models consist of an encoder-decoder structure (shown in Figure 31) that extracts meanings from a sequence of text, or embeddings, to allow understanding between the phrases and words of it.

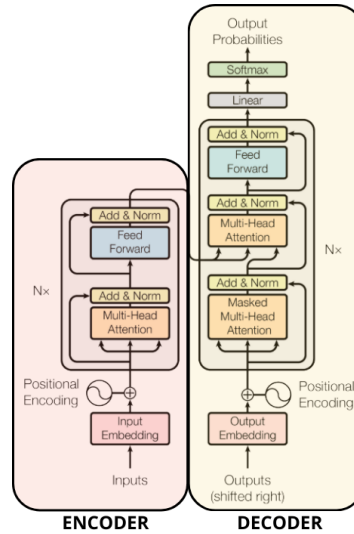


Figure 31. LLM architecture. Adapted from [24].

The architecture proposed is based on Qwen-Audio [31] structure seen on Figure 32. It consists of a LLM which is feed with audio features extracted from an audio encoder and a textual prompt. As the challenge did not allow the use of models pre-trained with emotional data, this model could not be used. To address this limitation, we developed a similar architecture by using Whisper Large V3 as audio encoder and exploring different LLMs with 1.5 and 2 billion parameters. We make it to be efficient with the resources and to reduce its training complexity. To make this, only the audio encoder will be trained, leaving the LLM at its pre-trained state.

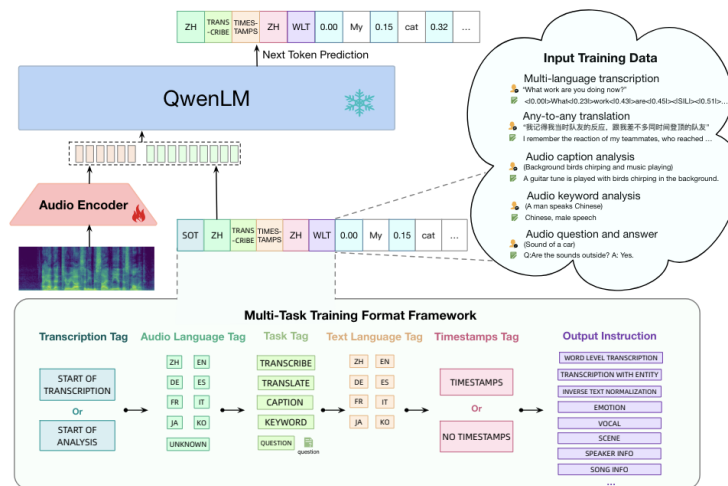


Figure 32. Qwen-Audio architecture. Reprinted from [31].

As mentioned, the audio encoder selected is Whisper Large V3 [32], developed by OpenAI. This model is defined as a pre-trained model for Automatic Speech Recognition (ASR) and speech translation. It is trained on 680,000 hours of labelled data, what gives it a strong ability to generalise over many datasets. It is designed to receive a Mel spectrogram as input and process it to obtain an embedding of that audio at the output. We used a projection layer at the end of it to adapt the embedding to the LLM input. The full structure of this encoder can be seen in Figure 33.

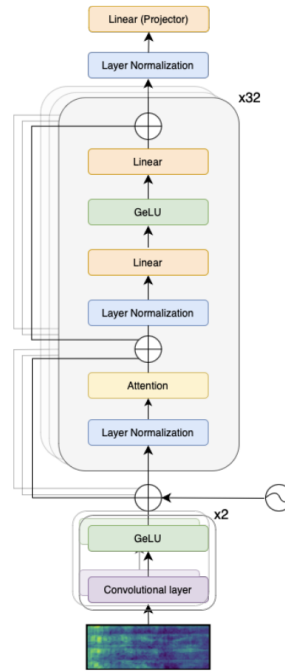


Figure 33. Whisper-large-v3 encoder architecture with a projection layer. Reprinted from [30].

As LLM, we experimented with two different models: Phi 1.5 [33] and Gemma 2. Phi 1.5 is a language model developed by Microsoft. It is based on transformers, and it has 1.3 billion parameters. The version used is not fine-tuned for instruction following or through reinforcement learning from human feedback. Even its small size, compared to other big LLMs like GPT 3.5 which has 175 billion parameters, it showed nearly state-of-the-art performance among models with less than 10 billion parameters. Also, the small size allows it for an easy implementation in other systems such is our case. On the other hand, Gemma 2 is a model developed by Google with 2 billion parameters. This model was trained with data from web documents, coding datasets, and mathematics. This variety of the training data helps it to handle more variety of inputs. Both LLMs receives the same input: a prompt containing the audio features and the transcriptions. This prompt can be seen in Figure 34. Finally, its output will contain the probabilities of belonging to each class.

```
"Instruction:
Given the following audio information and the transcription,
predict the emotion of the speaker.
The emotion can be one of the following:
[ neutral, sad, angry, happy, surprise, fear, disgust, contempt]
The transcription is: {transcripts}
Audio information:
\n
Output:
emotion ="
```

Figure 34. Prompt inserted to the LLM. Reprinted from [30].

The final model can be seen in Figure 35. As can be seen, we first obtain the embeddings of the audio features by processing them with the audio encoder (Whisper V3). At the same time, the transcriptions are processed to obtain its embeddings. This embedding is done with the tokenizer of the corresponding LLM (Phi o Gemma depending on which one is being tested). Then, the audio embeddings are projected and merged with the transcription ones, generating the input to the LLM. Finally, its output is processed to obtain the probabilities for each class.

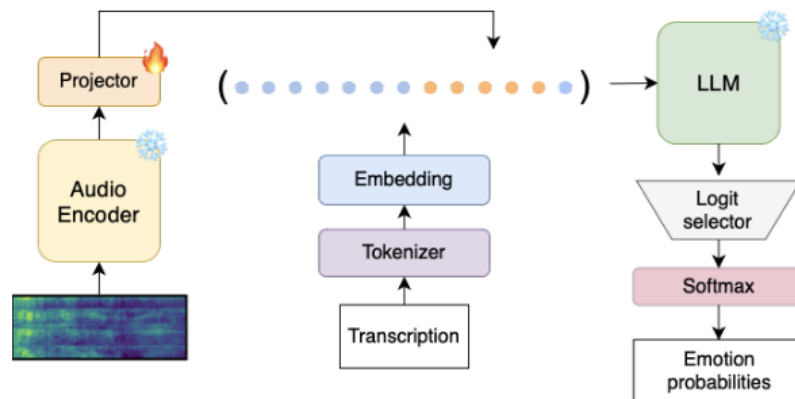


Figure 35. Proposed ALM architecture diagram. Reprinted from [30].

At its training we proposed a few alternatives for the hyperparameter configuration. The final ones that are used can be seen in Table 10. We used a learning rate decay for be more precise in the gradient updates as the training progresses. Also, as commented, the only layers that will be trained are the last layer of the audio encoder, the projection layer and the normalization layer.

Hyperparameter	Configuration
Batch size	4
Peak learning rate	1e - 4
Learning rate decay value	lr * 0.95 each 1000 steps
Optimizer	AdamW
Trainable layers	Last encoder block, Layer normalization, and Projection layer.
Gradient accumulation	1

Table 10. Hyperparameters for our proposed model.

6. RESULTS AND DISCUSSION

This chapter deeps into the experimental setup, presents the results obtained, and compares them with the state of the art for both tasks covered in this work.

6.1. HAND GESTURE RECOGNITION

The experiments made with the CNN model were used for validating the use of landmarks to solve the classification on dynamic gestures.

Around the ViViT model, the experiments explore the different values of the hyperparameters in search of better results and observe how they affect the gesture classification using the model. For a better understanding, they are explored one by one, isolating their effect from the rest. Experiments focus on finding the optimal projection dimensions, tubelet size and number of layers and heads. Also, some experiments were executed for finding a good front-end configuration related to the input format: MediaPipe errors strategy (Sec. 4.1.1), videos length (Sec. 4.1.4) and normalization (Sec. 4.1.3). The architecture's final version is based on the implementation by Aritra Roy Gosthipaty and Ayush Thakur tutorial in Keras [34].

6.1.1. EXPERIMENTAL SETUP

For the experiments we split the dataset into two groups: train (74%) and test (26%). This resulted in having 3,117 gesture instances for training and 1,101 for testing. This split is the same used in the original dataset paper for comparing purposes [4]. Both training and test splits are balanced between all classes except the pointing ones, which appears the most as Figure 36 shows. The results exposed along this section are the ones obtained from the test partition.

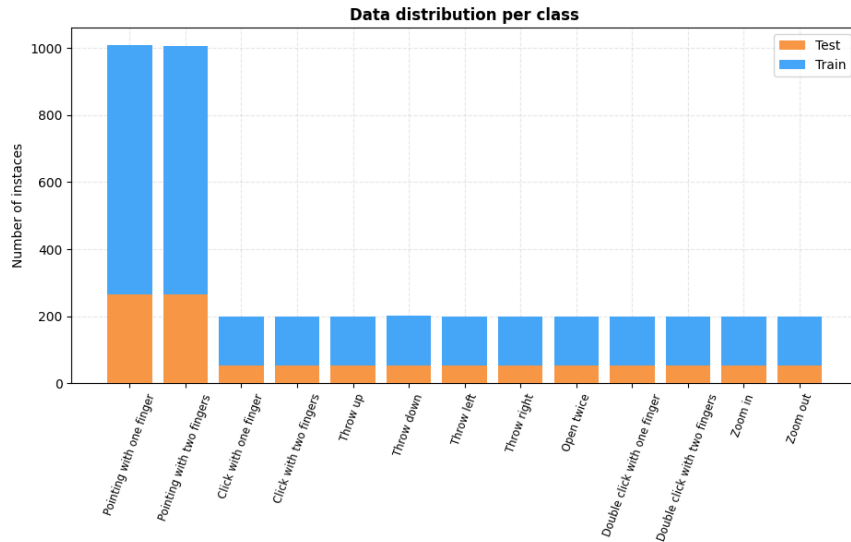


Figure 36. Data distribution per class.

The metrics used in the experiments were the Cross Entropy Loss (2) and the accuracy (3). Also, the confusion matrices were obtained for data analysis. These matrices show the comparison between the real class of the gesture and the one predicted by the model and are useful for showing up the problematic classes.

$$L = \sum_{c=1}^N -y_c \log(p_c) \quad (2)$$

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}} \quad (3)$$

The variable N represents the number of classes, with the value y_c set to 0 if the subject is not in that class and 1 if they are. The variable p_c represents the probability obtained from the model.

A 95% confidence interval was applied for results comparison: one is deemed to be superior to other when their intervals do not overlap. Equation (4) illustrates the calculation of the intervals, which require the input of the number of samples, N , and the metric to be validated, with a confidence factor of 95%.

$$CI(95\%) = \pm 1.96 \sqrt{\frac{\text{metric} \cdot (100 - \text{metric})}{N}} \quad (4)$$

6.1.2. RESULTS

CNN MODEL

The convolutional model proposed obtained results that can be compared to the baseline presented (Sec. 2.1). The results obtained with this model can be seen in Table 11. The best results were obtained with the landmarks normalized with first landmark normalization. Specifically, it obtained an accuracy of **85.20 ± 2.10 %**. These results justify that the use of landmarks could be a good input format for the gesture classification task.

Model	Accuracy
CNN	83.12 ± 2.21 %
CNN with FF norm.	85.20 ± 2.10%
CNN with 0L norm.	84.92 ± 2.11%

Table 11. Results obtained with the CNN model on MediaPipe landmarks.

VIVIT - FRONTEND EXPERIMENTS

Experiments related to MediaPipe errors solving, gestures duration, normalization, and interpolation are exposed along this section. These runs were by designing a base ViViT configuration and keeping it the same in every experiment. Specifically, its base configuration is: tubelet size of (5, 42), projection dim of 64, number of layers at 4, and number of heads also at 4. The experiments are done along 60 epochs and with a learning rate of 1e-4.

For a starting point, we first trained the model with the base settings: zero filling for MediaPipe errors, no normalization, and with a zero padding to a 970 frames duration (which is the duration of the largest video). This model got a validation accuracy of **83.56 ± 2.19 %**.

When comparing the results between the different strategies for **solving MediaPipe non-recognitions**, interpolation technics obtains better results. Specifically, a 2.09% accuracy increment can be seen with the interpolation enabled. Nevertheless, as confidence intervals overlap, this result is not sufficient to determine it as the best strategy, but a slight increment of performance is observed.

MediaPipe error strategy	Accuracy
Zero filling	83.56 ± 2.19 %
Interpolation	85.65 ± 2.07%

Table 12. MediaPipe error strategy results.

For the **normalization techniques**, Table 13 shows the results obtained in the experiments. Again, the experiments were done focusing only on the normalization effect, establishing the other parameters as default. This is why the results without normalization are the same as the baseline. While the use of normalization seems to give better results, it still overlaps with the one without it. Also, there is no difference between *first frame normalization (FF)* and *0 landmark normalization (OL)* as the confidence intervals also overlap. However, since the use of normalization shows a slight improvement, first frame normalization is used in the final system.

Normalization	Accuracy
NONE	$83.56 \pm 2.19 \%$
FF	$85.38 \pm 2.10 \%$
OL	$85.65 \pm 2.07 \%$

Table 13. Normalization experiment results.

The experiments around the **duration of the gestures** show the importance of this parameter. The experiments done compare the use of the gestures without trimming, where all ones were padded to the longer one, versus the use of shorter ones with and without the GAD system described in section 4.1.4. Videos shorter than the target duration were padded with 0s. The results can be seen in Table 14. The GAD does not seem to have any effect on the results. This leads to thinking that the gestures are well located at the beginning of the videos and the GAD is not necessary. They also show that the use of a too short duration, such as 50 frames, makes the model obtain worse results, probably since this duration does not ensure all videos fit inside. We can also see that the use of a duration of 150 frames obtains better results than the baseline.

Duration [frames]	GAD	Accuracy
970 (full videos)	Deactivated	$83.56 \pm 2.19 \%$
150	Deactivated	$86.38 \pm 2.03 \%$
150	Activated	$85.20 \pm 2.10 \%$
100	Deactivated	$84.20 \pm 2.15 \%$
100	Activated	$83.92 \pm 2.17 \%$
50	Deactivated	$81.83 \pm 2.28 \%$
50	Activated	$81.23 \pm 2.31 \%$

Table 14. Gestures duration experiments results.

In section 4.1.4, we proposed two methods for doing the **padding** to short videos. When doing the loop padding, the gestures that are repetitive are practically the same as the ones that are just one repetition, such as double click and single click. At Figure 37, a confusion matrix done with the baseline (**Acc.** = $83.56 \pm 2.19 \%$) presented at the beginning of this section is shown. At Figure 38, the results are obtained with loop-padding (**Acc.** = $84.20 \pm 2.15 \%$). Even the results obtained for both cases overlap, when using this approach, the model gets more confused between one click as double click, classifying nearly every instance of one click as two clicks.

No gesture -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pointing with one finger -	0	259	0	3	1	0	1	0	0	0	1	0	0	0
Pointing with two fingers -	0	18	241	0	3	1	1	0	0	0	0	0	0	0
Click with one finger -	0	1	0	29	0	0	1	0	0	0	21	0	0	0
Click with two fingers -	0	0	1	0	30	0	0	0	0	0	19	0	2	0
Throw up -	0	0	0	0	1	48	2	0	0	1	0	0	0	0
Throw down -	0	1	0	0	0	0	50	0	0	0	1	0	0	0
Throw left -	0	0	0	0	0	0	0	52	0	0	0	0	0	0
Throw right -	0	0	0	0	0	0	3	0	49	0	0	0	0	0
Open twice -	0	0	0	0	1	12	3	0	8	26	0	0	1	1
Double click with one finger -	0	0	0	15	0	0	0	0	0	0	37	0	0	0
Double click with two fingers -	0	0	0	0	19	0	0	0	0	0	0	33	0	0
Zoom in -	0	0	1	1	9	0	1	0	6	0	0	0	31	3
Zoom out -	0	0	0	2	5	0	1	0	5	0	1	0	8	30
	No gesture	Pointing with one finger	Pointing with two fingers	Click with one finger	Click with two fingers	Throw up	Throw down	Throw left	Throw right	Open twice	Double click with one finger	Double click with two fingers	Zoom in	Zoom out

Figure 37. Confusion matrix for baseline model. 0 padding is used.

No gesture -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pointing with one finger -	0	260	0	1	0	0	0	0	0	0	4	0	0	0
Pointing with two fingers -	0	4	254	1	2	0	0	0	2	0	0	1	0	0
Click with one finger -	0	0	0	6	0	0	0	0	0	0	46	0	0	0
Click with two fingers -	0	0	2	0	23	0	0	0	0	0	0	27	0	0
Throw up -	0	0	0	0	1	47	1	0	0	3	0	0	0	0
Throw down -	0	2	0	1	0	1	47	0	0	0	1	0	0	0
Throw left -	0	0	0	0	0	0	0	50	2	0	0	0	0	0
Throw right -	0	1	0	0	0	0	0	0	49	2	0	0	0	0
Open twice -	0	0	0	1	0	4	3	0	1	41	0	0	1	1
Double click with one finger -	0	0	0	3	0	0	0	0	0	0	49	0	0	0
Double click with two fingers -	0	0	1	1	11	0	0	0	0	0	0	39	0	0
Zoom in -	0	0	0	1	2	0	0	0	2	0	1	1	33	12
Zoom out -	0	0	0	2	0	0	1	0	1	2	2	4	11	29
	No gesture	Pointing with one finger	Pointing with two fingers	Click with one finger	Click with two fingers	Throw up	Throw down	Throw left	Throw right	Open twice	Double click with one finger	Double click with two fingers	Zoom in	Zoom out

Figure 38. Confusion matrix for baseline model. Loop padding is used.

For analyzing the **interpolation** of frames, both the input shape and patches will keep the same time scope. This means that, if the interpolation makes the framerate of 60 fps, the input shape will be $970 * 2 = 1,940$ frames. By making this we ensure the full video fits in with the model. The same occurs with the first dimension of tubelet size. We scale them so their range remains the same in time domain. The results obtained by these experiments can be seen in Table 15. As can be seen, the interpolation has no significant effect, and in fact it causes worse results. This is probably because of the increase in complexity, which makes the model overfit.

Interpolation	Accuracy
No: 30 fps	$83.56 \pm 2.19 \%$
Yes: 60 fps	$80.01 \pm 2.36 \%$
Yes: 90 fps	$81.50 \pm 2.29 \%$

Table 15. Results of the experiments with interpolation.

VIVIT – MODEL HYPERPARAMETERS

Although the results shown in the previous section do not provide a significant improvement, we chose to set up the frontend interpolation between missed frames, the first frame normalization and a duration of 150 frames with the GAD deactivated. These settings seem to be slightly better and help with a smaller input size. Experimental runs focus on 6 different hyperparameters: projection dimensions, tubelet size, number of layers, number of heads, learning rate and number of epochs. For each hyperparameter experiment only that one is modified. The value of the others remains the same as on base ViViT presented in previous section (tubelet size of (5, 42), projection dim of 64, number of layers at 4, and number of heads also at 4). As before, the experiments are done along 60 epochs and with a learning rate of $1e-4$.

The **projection dimensions** parameter defines the number of filters that are applied on the convolutional layer. The results obtained when varying it can be seen in Table 16. As can be seen, the results obtained overlapped for every value except for 512. This can be explained because of the increase of complexity of the model, that makes it overfit to the training data.

Projection dimensions	Accuracy
16	$86.29 \pm 2.25 \%$
32	$84.56 \pm 2.13 \%$
64	$86.38 \pm 2.03 \%$
128	$84.65 \pm 2.28 \%$
256	$86.29 \pm 2.00 \%$
512	$75.66 \pm 2.07 \%$

Table 16. Variation of projection dimensions.

The **tubelet size** hyperparameter represents the dimensions of the kernel of the convolutional layer that is applied to the input. Experiments in Table 17 shows the effect of varying the first dimension, which corresponds to the time axis. As we can see, the model is robust to this change, seeing no variation between different sized. At Table 18 can be seen the results when decreasing the second dimension. In this case, the accuracy decreases when making it too small.

Tubelet size	Accuracy
(1, 42)	$85.74 \pm 2.07 \%$
(3, 42)	$85.29 \pm 2.09 \%$
(5, 42)	$86.38 \pm 2.03 \%$
(10, 42)	$84.56 \pm 2.13 \%$
(15, 42)	$85.65 \pm 2.07 \%$

Table 17. Variation of time dimension of the tubelet.

Tubelet size	Accuracy
(5, 42)	$86.38 \pm 2.03 \%$
(5, 21)	$83.47 \pm 2.19 \%$
(5, 14)	$86.29 \pm 2.03 \%$
(5, 7)	$81.93 \pm 2.27 \%$
(5, 3)	$78.38 \pm 2.43 \%$

Table 18. Variation of dimension of landmark points of the tubelet.

The experiments done around the **number of layers** can be seen in Table 19. Again, it shows that the model is not affected in a big form because of this parameter.

Number of layers	Accuracy
2	85.29 \pm 2.09 %
4	86.38 \pm 2.03 %
8	84.20 \pm 2.15 %
10	86.65 \pm 2.01 %
16	86.04 \pm 2.30 %

Table 19. Experiments around number of layers of the model.

The effect of the **number of heads** is like the number of layers. The results of the experiments can be seen in Table 20. The results overlap for all settings, so it does not seem to have a relevant effect.

Number of heads	Accuracy
2	85.29 \pm 2.09 %
4	86.38 \pm 2.03 %
8	86.47 \pm 2.02 %
10	87.38 \pm 1.96 %
16	85.74 \pm 2.07 %

Table 20. Experiments around number of heads.

After trying some of the combinations from above, the full model implemented the following hyperparameters: projection dimensions = 64, tubelet size = (5,42), number of layers = 2, number of heads = 4. With this configuration, we obtained a testing accuracy of **88.1 \pm 1.91 %**. Also, the confusion matrix can be seen in Figure 39, which shows the difficulties found into classifying correctly between single and double click. Note that the “No gesture” class is removed from the dataset, so both the row and column is always 0.

No gesture -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pointing with one finger -	0	259	2	1	0	0	2	0	0	0	0	1	0	0
Pointing with two fingers -	0	3	259	0	2	0	0	0	0	0	0	0	0	0
Click with one finger -	0	1	0	28	0	0	0	0	0	0	23	0	0	0
Click with two fingers -	0	0	2	0	27	1	0	0	0	0	1	20	0	1
Throw up -	0	0	0	0	0	38	1	0	0	11	0	0	0	2
Throw down -	0	0	0	1	0	0	50	0	0	0	1	0	0	0
Throw left -	0	0	0	0	0	0	0	51	1	0	0	0	0	0
Throw right -	0	0	0	0	0	0	0	1	51	0	0	0	0	0
Open twice -	0	0	0	0	0	0	2	0	0	48	0	0	1	1
Double click with one finger -	0	0	0	15	0	0	0	0	0	0	37	0	0	0
Double click with two fingers -	0	0	0	0	13	0	0	0	0	0	0	38	0	1
Zoom in -	0	1	0	2	0	0	0	0	3	0	0	0	43	3
Zoom out -	0	2	0	1	0	0	0	0	0	2	2	0	4	41
	No gesture -	Pointing with one finger -	Pointing with two fingers -	Click with one finger -	Click with two fingers -	Throw up -	Throw down -	Throw left -	Throw right -	Open twice -	Double click with one finger -	Double click with two fingers -	Zoom in -	Zoom out -

Figure 39. Confusion matrix of the final model.

6.1.3. STATE OF THE ART COMPARISON

Results obtained along the experiments made can be compared with the state of the art of the task. For a fair comparison, the results are contrasted against the ones obtained in the IPN Hand dataset presented in section 2.1. A detailed summary of the results can be found in Table 21.

On one side, the use of landmarks by themselves could provide a similar result compared to using the RGB-Flow frames using similar architectures.

On the other hand, the application of transformers, specifically ViViT models, obtains similar results as using CNN models from RGB-Flow frames [4] or multimodal technics [9]. Nevertheless, our approach shows competitive results that overlaps with the state of the art with a model much smaller than it. Specifically, our results overlap the multimodal solution while using a model 6 times smaller than it.

Model	Accuracy	# of parameters
IPN_Hand Baseline [4] (RGB-Flow + ResNeXt-101)	$86.32 \pm 2.03\%$	47.56M
Multimodal solution [9] (RGB + Flow + ECSA blocks)	$91.40 \pm 1.66 \%$	27.2M
MediaPipe Landmarks + CNN (this work)	$85.20 \pm 2.10\%$	$2.01\text{M} + 20.86\text{M} = 22.87\text{M}$
MediaPipe Landmarks + ViViT (this work)	$88.10 \pm 1.91 \%$	$2.01\text{M} + 1.65\text{M} = 3.66\text{M}$

Table 21. Summary of models compared for HGR task.

6.2. SPEECH EMOTION RECOGNITION

Along this section we will compare the different models proposed against them. As a baseline, the challenge authors published their own results: $F1 = 0.3113 \pm 0.0187$. Unlike the results of this section, this metric was obtained against their private test set. Nevertheless, it can be used as a starting point to indicate the complexity of the task.

6.2.1. EXPERIMENTAL SETUP

The challenge gave two splits of training and validation. We kept the training subset, and, for the validation and test subsets, we took the original validation split and divided it in two while respecting the distribution of the data. All results that are exposed along this section are the ones obtained from the “in-house” test partition. From these divisions, we removed the “none” and “other” classes from all the subsets and the ones with less of 50% of agreement from the training subset as is explained in Section 4.2.2. At the end, we have 46,994 instances for training, 7,087 for validation, and 7,087 for testing (61,168 in total). The distributions of these partitions can be seen in Figure 40.

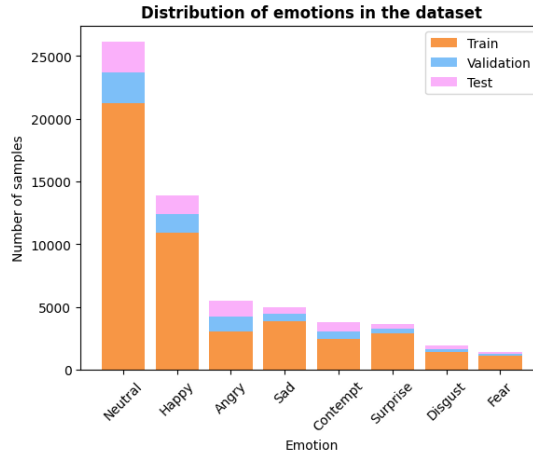


Figure 40. Distribution of emotions in the splits of the dataset.

The metrics used in the experiments were the Cross Entropy Loss (2) and the F1 macro score. This last one is the one used for the ranking in the Odyssey 2024 challenge. For a binary class, the F1 score is obtained as (5). For a multiclass, as in our case, the F1 macro is the arithmetic mean of the F1 scores for each class. The confusion matrices were also obtained for data analysis.

$$F1\ macro = \frac{2\ TP}{2\ TP + FP + FN} \quad (5)$$

Where TP stands for True Positives (the ones that were correctly predicted for that class), FP stands for False Positives (the ones that were predicted as that class but belong to another), and FN stands for False Negatives (the ones that belong to that class but were predicted as another).

For accounting the data imbalance, we opted for the weighted cross-entropy loss function shown in (6). This technic proved to be effective for attenuating the imbalance problem, helping for a better generalization.

$$L = \sum_{c=1}^N -y_c w_c \log(p_c) \quad (6)$$

Where the variable y_c is set to 0 if the emotion does not belong class and 1 if they are, the variable p_c represents the probability obtained from the model to belong that class, and w_c is the weight assigned to that class. The weight is obtained with the number of instances of each class as shown in (7).

$$w_c = \frac{\text{total instances}}{\text{instances of the class}} \quad (7)$$

6.2.2. RESULTS

CNN (RESNET-50)

The ResNet model obtained the results shown in Table 22. As can be seen, these results are still far from the baseline published by the challenge authors, encouraging the use of different architectures.

Model	F1 macro	# of parameters
ResNet-50	0.1343 ± 0.0079	23.5M

Table 22. ResNet-50 results on SER task.

VIVIT

The same way we did on the HGR task, we did experiments with different configurations of the ViViT. On the experiments, we tested with different values for the projection dimensions, tubelet size, number of layers and number of heads. All experiments are done from a base configuration with the following hyperparameters: tubelet size of (5, 16, 1), projection dimensions of 64, and number of layers and heads at 4. Moreover, the experiments were done with a duration of 60 epochs. At this point, most of the configurations are still improving their results, nevertheless, we compare them at this point to reduce the duration and the consumption of the runs.

First, we experimented with different **tubelet sizes**. The results obtained can be seen in Table 23 and Table 24. As reminder, the first dimension of the size represents the time, the second is the height of the spectrogram, and the third is set always to 1. The results show how as we increase the tubelet size, the resolution is decreased, leading to worse results.

Tubelet size	F1 macro
(2, 16)	0.1360 ± 0.0080
(5, 16)	0.1095 ± 0.0073
(10, 16)	0.0970 ± 0.0069
(20, 16)	0.0934 ± 0.0068
(30, 16)	0.0926 ± 0.0067

Table 23. Variation of time span in the tubelets.

Tubelet size	F1 macro
(5, 4)	0.1353 ± 0.0080
(5, 8)	0.1313 ± 0.0079
(5, 16)	0.1095 ± 0.0073
(5, 32)	0.1013 ± 0.0070
(5, 64)	0.0991 ± 0.0070

Table 24. Variation of height span in the tubelets.

The experiments varying the **projection dimensions** showed the results of Table 25. It can be observed that the F1 macro score increases when increasing the number of dimensions. Nevertheless, this increase is small.

Projection dimensions	F1 macro
32	0.0975 ± 0.0069
64	0.1095 ± 0.0073
128	0.1142 ± 0.0074
256	0.1192 ± 0.0075

Table 25. Variation of projection dimensions of ViViT for SER task.

The experiments done around the **number of layers** can be seen in Table 26. It seems that the experiment setting this value to 8 performs the best.

Number of layers	F1 macro
2	0.0995 ± 0.0070
4	0.1095 ± 0.0073
8	0.1405 ± 0.0081
12	0.1208 ± 0.0076

Table 26. Variation of number of layers in the model.

The effect of the **number of heads** is like the number of layers. The results of the experiments can be seen in Table 27. These experiments show that the F1 macro increases when increasing the number of heads of the model.

Number of heads	F1 macro
2	0.0978 ± 0.0069
4	0.1095 ± 0.0073
8	0.1283 ± 0.0078
12	0.1434 ± 0.0082

Table 27. Variation of number of heads in the model.

Finally, we implemented the final configuration: a tubelet size of [2,8,1]; a projection dimension of 64, 8 layers and 8 heads. We used 8 heads instead of 12 since the increase of heads also produces an increase in the size of the model, making it slower to train. With these hyperparameters we obtained a F1 macro of 0.1781 ± 0.0089 . The Figure 41 shows the normalized confusion matrix obtained using this classifier. It can be observed that the model confuses a lot the gestures with the neutral class, this can be because of the similarity to other emotions. As well, the model does recognize some instances belonging to the classes “happy” or “angry” but classifies almost none of them in the other classes.



Figure 41. Normalized confusion matrix obtained from the ViViT classifier.

LLM MULTIMODAL

For a better understanding of the multimodality benefits, we first tried to perform the task by only using one of them. To classify by only the audios, we added a dense layer on top of the Whisper model, obtaining a F1 score of **0.246**. As can be seen, this result is still far from the baseline of the challenge, nevertheless, it obtained better results than using the ViViT. On the other hand, we used a text classifier with the transcriptions. This classifier is based on DistilRoBERTa [35], which can classify a text into 7 different emotions. It obtained a F1 score of **0.240**, still far from the baseline. Finally, we also tested the LLM models on their own with the transcriptions to understand their inherent emotion recognition capabilities. The summary of these results can be seen in Table 28.

Model	F1 macro	# of parameters
Whisper V3	0.2460 ± 0.0100	1.55B
Text Classifier	0.2400 ± 0.0099	82.8M
Phi 1.5B	0.0830 ± 0.0064	1.5B
Gemma 2B	0.0920 ± 0.0067	2B

Table 28. Results with the isolated models.

After these experiments we finally combine both information sources on the final multimodal model. The results are shown in Table 29. As can be seen, the combination of both formats produces better results than using only one of them. We can also see that both models perform in a similar way, overlapping their results. Also, to better understand its operation, the confusion matrix can be seen in Figure 42. As is shown, the model confuses above all the neutral emotion. This makes sense as is the most generic one. We can also observe a curious behavior: disgust, fear and contempt are also confused with the angry emotion.

Model	F1 macro	# of parameters
Whisper V3 + Phi 1.5B	0.3040 ± 0.0107	1.55 B + 1.5 B = 3.05B
Whisper V3 + Gemma 2B	0.3270 ± 0.0109	1.55B + 2B = 3.55B

Table 29. ALM results.

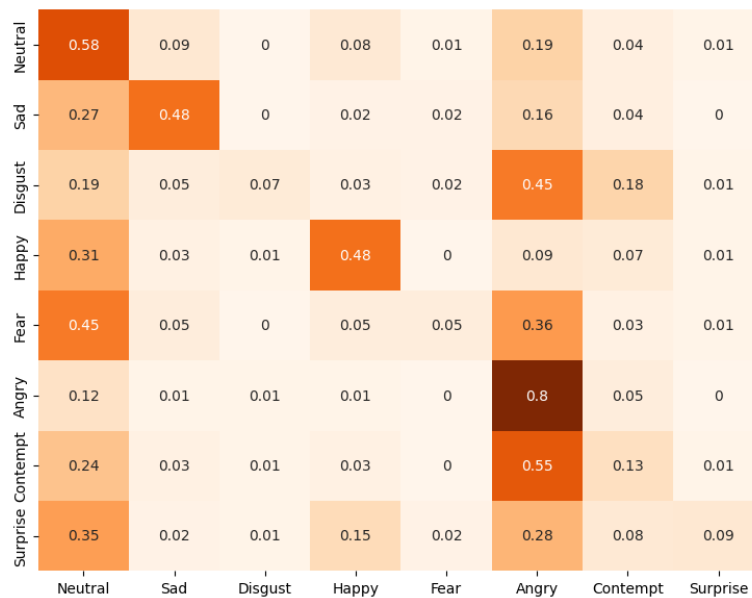


Figure 42. WhisperV3 + Gemma 2b In-house test Normalized Confusion Matrix. Adapted from [30].

6.2.3. MODELS COMPARISON

To make a fair comparison, we compared CNN, ViViT with Whisper V3 isolated. This comparison is done to visualize the classifications capabilities of the three models when only receiving the Mel spectrograms. This comparison can be seen in Table 30. As can be seen the Whisper V3 model outperforms both the ResNet and the ViViT. This can be easily explained as the Whisper V3 is already pretrained over a vast number of audios. Nevertheless, this model is much bigger than the ViViT, which can make it better if there exist space limitations as it is incredibly small. Also, it is clearly shown that multimodality produces better results than using only one modality of the information, with the disadvantage of obtaining an even bigger model.

Model	F1 macro	# of parameters
ResNet - 50	0.1343 ± 0.0079	23.5M
ViViT	0.1781 ± 0.0089	258K
Whisper V3	0.2460 ± 0.0100	1.55B
Whisper V3 + Phi 1.5B	0.3040 ± 0.0107	$1.55 \text{ B} + 1.5 \text{ B} = 3.05\text{B}$
Whisper V3 + Gemma 2B	0.3270 ± 0.0109	$1.55\text{B} + 2\text{B} = 3.55\text{B}$

Table 30. Audio classifiers comparison.

As final evaluation, the model conformed by Whisper V3 and Gemma 2B was sent to the challenge. In their private dataset, the model obtained an F1 score of $\mathbf{0.3426 \pm 0.0192}$. This score was set up as seventh on the leaderboard, overlapping with the best result (F1 score = $\mathbf{0.3569 \pm 0.0194}$). This first-place solution consisted in a similar system [36]. Since the class imbalance problem, their approach was to develop an assembly of 7 multimodal audio-text models, with slight variations in their architecture and loss functions. The final class is then decided by majority voting. They showed that by using weighted losses alone, the model tends to overfit minority classes and underfit majority classes. By combining different approaches, they can resolve the class imbalance problem in a more consistent way. The details of their implementations and the scores can be seen in Table 31. However, the low results indicate the difficulty of the task.

Index	Audio	Text	Loss	Class Weights	F1 macro
1	Whisper	Roberta	Focal ($\gamma = 2$)	Prior-based	0.3420 ± 0.0075
2	WavLM	Roberta	Focal ($\gamma = 2$)	Prior-based	0.3240 ± 0.0074
3	Hubert	Roberta	Focal ($\gamma = 2$)	Prior-based	0.3270 ± 0.0074
4	Whisper	Roberta	Focal ($\gamma = 2.5$)	Prior-based	0.3380 ± 0.0075
5	Whisper	Roberta	Cross Entropy	Prior-based	0.3380 ± 0.0075
6	Whisper	Roberta	Focal ($\gamma = 2$)	Uniform	0.3330 ± 0.0075
7	Whisper	Roberta	Cross Entropy	Uniform	0.3280 ± 0.0074
Assembly	-	-	-	-	0.3560 ± 0.0076

Table 31. Results from assembly of the 1st place solution. Adapted from [36].

7. CONCLUSIONS AND FUTURE WORK

To sum up, through this work we focused on modelling human behavior using deep learning models. Specifically, we focused on the implementation of two separate systems: one designed for Hand Gesture Recognition (HGR) and another for Speech Emotion Recognition (SER). With their implementation, we aspire to achieve a more natural and useful communication between humans and machines.

Around the HGR system:

- We have shown that it is possible to achieve the baseline set up by the authors of the dataset (RGB-Flow + ResNeXt-101) with a much simpler model by using a CNN and the hand landmarks. By doing this we obtained an accuracy of $85.20 \pm 2.10\%$ with a model of **22.87M** of parameters, compared with the $86.32 \pm 2.03\%$ accuracy and **47.56M** of parameters of the reference.
- We demonstrate effective ways of preprocessing the hand landmarks, such as normalizing them or cutting them up, to obtain better results in the classification.
- We proposed the use of a ViViT model to solve the task. To achieve that, we propose a baseline configuration and optimize the different hyperparameters to obtain the best possible results.
- Using the optimized ViViT model, we achieve results similar to the state of the art, while reducing the size of the solution by six. Specifically, we obtain an $88.10 \pm 1.91\%$ accuracy with a model of **3.66M** of parameters compared with the $91.4 \pm 1.66\%$ accuracy and **27.2M** parameters of the state of the art.

Regarding the SER task:

- We propose the use of audio features and text, specifically the transcriptions, to solve the problem. With the implementation of this multimodal system, we can achieve better results than using just one of them.
- To obtain the audio features we explore three different systems: a ResNet-50, a ViViT and a Whisper V3. The first two models were trained from scratch, while the last one was already pretrained for audio classification. To compare them we tested them solving the task with Mel spectrograms as input. As same as in the previous task, we optimized the hyperparameters of the ViViT model, experimenting different combinations and showing its effects. At the end, the ResNet-50 obtained an F1 macro of 0.1343 ± 0.0079 with **23.5M** of parameters, the ViViT obtained a 0.1781 ± 0.0089 with **258K** parameters, and the Whisper obtained a 0.2460 ± 0.0100 with **3.55B** of parameter. From this we observed that CNN are not up to the task of classifying emotions from audio, as using a ViViT model, which is 91 times smaller than this CNN, we obtained better results. Furthermore, it is shown that Whisper takes advantage of being an audio pretrained model, obtaining the best score among the systems we tested but being 5,814 times bigger than the ViViT, which is the second best.
- We implemented a multimodal system conformed of a Whisper V3 model and a LLM (Phi 1.5B or Gemma 2B). This model extracts the audio features using Whisper and integrates them into a prompt with the transcriptions. This prompt is then inserted into the LLM. With these models we obtained an F1 score of 0.3040 ± 0.0107 with **3.05B** of parameters and 0.3270 ± 0.0109 with **3.55B** of parameters, respectively. This showed the importance of the multimodality for this task.
- Finally, the system conformed of Whisper V3 and Gemma 2B was submitted to the Odyssey 2024 challenge. At the end, our system ended up in seventh position with an F1 score of 0.3426 ± 0.0192 . Nevertheless, this result overlaps with the first position of the leaderboard (0.3569 ± 0.0194).

Future work includes investigating the ViViT model using data augmentation techniques, which may lead to a better generalization of the model. This is because, as transformers tend to overfit the data more easily, increasing the training resources should lead to better results. In addition, in the emotion recognition task, it may be interesting first to keep increasing the heads of the ViViT model and its training time, as it appears that it can obtain even better results, and to integrate the ViViT into the multimodal solution, leading to more feature extractions from the data and, perhaps, better results.

8. BIBLIOGRAPHY

- [1] Y. Wang and R. Yang, "Real-time hand posture recognition based on hand dominant line using kinect," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, pp. 1–4. doi: 10.1109/ICMEW.2013.6618237.
- [2] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998, doi: 10.1109/5254.708428.
- [3] B. Núñez Fernández Dennis and Kwolek, "Hand Posture Recognition Using Convolutional Neural Network," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, S. Mendoza Marcelo and Velastín, Ed., Cham: Springer International Publishing, 2018, pp. 441–449.
- [4] G. Benitez-Garcia, J. Olivares-Mercado, G. Sanchez-Perez, and K. Yanai, "IPN Hand: A Video Dataset and Benchmark for Real-Time Continuous Hand Gesture Recognition," in *25th International Conference on Pattern Recognition, ICPR 2020, Milan, Italy, Jan 10–15, 2021*, 2021, pp. 4340–4347.
- [5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks." 2015.
- [6] K. Hara, H. Kataoka, and Y. Satoh, "Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?" 2018.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." 2015.
- [8] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks." 2017.
- [9] B. Hampiholi, C. Jarvers, W. Mader, and H. Neumann, "Convolutional Transformer Fusion Blocks for Multi-Modal Gesture Recognition," *IEEE Access*, vol. 11, pp. 34094–34103, 2023, [Online]. Available: <https://api.semanticscholar.org/CorpusID:257883607>
- [10] Manuel Gil-Martín, Rubén San-Segundo, and Ricardo de Córdoba, "Hand Pose Recognition through MediaPipe Landmarks," *Modeling Decisions for Artificial Intelligence - USB Proceedings*, pp. 13–23, 2023.
- [11] N. Y. Kim and S.-I. Kim, "A study on user experience of online education programs with elementary schools and art museums in non-face-to-face era," *Journal of Digital Convergence*, vol. 19, no. 8, pp. 311–317, 2021.
- [12] C. Darwin, P. Ekman, and P. Prodger, *The Expression of the Emotions in Man and Animals*. Oxford University Press, 1998. [Online]. Available: <https://books.google.es/books?id=TFRtLZSHMcYC>
- [13] M. K. Chowdary, T. N. Nguyen, and D. J. Hemanth, "Deep learning-based facial emotion recognition for human–computer interaction applications," *Neural Comput Appl*, vol. 35, no. 32, pp. 23311–23328, 2023, doi: 10.1007/s00521-021-06012-8.
- [14] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression".
- [15] G. Banerjee, E. Huang, and A. Lettieri, "Understanding Emotion Classification In Audio Data Stanford CS224N Custom Project".
- [16] Dalya Gartzman, "Getting to Know the Mel Spectrogram | Towards Data Science," Towards Data Science. Accessed: May 29, 2024. [Online]. Available: <https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0>

- [17] Z. Kh. Abdul and A. K. Al-Talabani, "Mel Frequency Cepstral Coefficient and its Applications: A Review," *IEEE Access*, vol. 10, pp. 122136–122158, 2022, doi: 10.1109/ACCESS.2022.3223444.
- [18] J. Yang, F. L. Luo, and A. Nehorai, "Spectral contrast enhancement: Algorithms and comparisons," *Speech Commun*, vol. 39, no. 1–2, pp. 33–46, Jan. 2003, doi: 10.1016/S0167-6393(02)00057-2.
- [19] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proceedings of the ACM International Multimedia Conference and Exhibition*, May 2006. doi: 10.1145/1178723.1178727.
- [20] "Chroma Feature Analysis and Synthesis." Accessed: May 29, 2024. [Online]. Available: <https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/>
- [21] A. Dosovitskiy *et al.*, "AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE", Accessed: May 29, 2024. [Online]. Available: <https://github.com/>
- [22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", Accessed: May 29, 2024. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>
- [23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks", Accessed: May 29, 2024. [Online]. Available: <https://github.com/liuzhuang13/DenseNet>.
- [24] A. Vaswani *et al.*, "Attention Is All You Need," 2023.
- [25] "Odyssey 2024 - Emotion Recognition Challenge." Accessed: May 29, 2024. [Online]. Available: <https://www.odyssey2024.org/emotion-recognition-challenge>
- [26] R. Lotfian and C. Busso, "Building Naturalistic Emotionally Balanced Speech Corpus by Retrieving Emotional Speech from Existing Podcast Recordings," *IEEE Trans Affect Comput*, vol. 10, no. 4, pp. 471–483, 2019, doi: 10.1109/TAFFC.2017.2736999.
- [27] "Hand landmarks detection guide | MediaPipe | Google for Developers." Accessed: Mar. 26, 2024. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker
- [28] "Mel scale - Wikipedia." Accessed: May 30, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Mel_scale
- [29] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "ViViT: A Video Vision Transformer." 2021.
- [30] J. Bellver-Soler *et al.*, "Multimodal Audio-Language Model for Speech Emotion Recognition".
- [31] Y. Chu *et al.*, "Qwen-Audio: Advancing Universal Audio Understanding via Unified Large-Scale Audio-Language Models", Accessed: May 31, 2024. [Online]. Available: <https://qwen-audio.github.io/>
- [32] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision", Accessed: May 31, 2024. [Online]. Available: <https://github.com/openai/>
- [33] Y. Li *et al.*, "Textbooks Are All You Need II: phi-1.5 technical report," Sep. 2023, Accessed: May 31, 2024. [Online]. Available: <https://arxiv.org/abs/2309.05463v1>
- [34] Aritra Roy Gosthipaty and Ayush Thakur, "Video Vision Transformer for Keras." Accessed: Mar. 25, 2024. [Online]. Available: <https://keras.io/examples/vision/vivit/>
- [35] J. Hartmann, "Emotion English DistilRoBERTa-base." 2022.
- [36] M. Chen *et al.*, "1st Place Solution to Odyssey Emotion Recognition Challenge Task1: Tackling Class Imbalance Problem." Jun. 2024.

-
- [37] “Electricity grids creak as AI demands soar.” Accessed: Jun. 17, 2024. [Online]. Available: <https://www.bbc.com/news/articles/cj5l189dy2mo>
- [38] “Weights & Biases: The AI Developer Platform.” Accessed: Apr. 11, 2024. [Online]. Available: <https://wandb.ai/site>

9. ANNEX A: ETHICAL, ECONOMIC, SOCIAL AND ENVIRONMENTAL ASPECTS

This annex shows an ethical, economic, environmental and social analysis of the implications and effects of the project.

A.1 INTRODUCTION

This project comes at a time when Artificial Intelligence systems are gaining enormous popularity. They have shown that they can become a powerful tool when using them correctly. Nowadays, everyone knows about it, and it is applied to nearly every system and field. It has infinity possible applications, from making a chatbot, to editing images and videos. Its versatility makes them so popular and useful. The project developed through this work is framed in the field of research. In this sense, its applications are aimed at developing and experimenting with new technologies for the improvement of existing systems. These systems, are stated before, aim to make people's lives easier by providing more versatile and useful tools.

A.2 DESCRIPTION OF RELEVANT IMPACTS RELATED WITH THE PROJECT

We found the following relevant impacts for this project:

- Social impacts: The project developed has numerous social impacts. AI-based systems are widely known by the society, being used for nearly every task. By enhancing their applications and context understanding, they can become even more powerful to the final users.
- Ethic and privacy impacts: The artificial intelligence systems are trained with tons of data. The way this data is collected must be ethic, respecting the privacy of the users.
- Economic and environmental impacts. The designing and training of deep learning models implies the use of enormous quantities of energy, which can lead to a problem if not considered. According to BBC [37], “The world’s data centers are using ever more electricity. In 2022, they gobbled up 460 terawatt hours of electricity, and the International Energy Agency (IEA) expects this to double in just four years. Data centers could be using a total of 1,000 terawatts hours annually by 2026.”. As can be seen, this opens a great problem on how to generate all this energy to keep the progress in an environmentally friendly way.

A.3 DETAILED ANALYSIS OF SOME OF THE MAIN IMPACTS

Around social impacts, we found that the implementation of human-machine interfaces can help certain minority groups to become more integrated into society. As mentioned before, gesture recognition interfaces can help people that communicate through gestures to interact more fluently with other people. Also, by building emotional interfaces we can create systems with more comprehension of humans, achieving a more natural communication and obtaining responses that can bring more to the users.

On the ethics side, the collection of millions of user data must be controlled. This collection must ensure that it complies with the rights of users, while respecting their privacy.

On the economic and environmental part, we focused on developing low consuming energy. We applied the concept of transfer learning, which reduces the expenses by reutilizing already learned information. Also, we focused on developing low consuming models, by making them small while maintaining the same capabilities.

A.4 CONCLUSIONS

To sum up, the IA tools are becoming more powerful and useful. Nearly every field is watching to introduce them as a tool. While its applications are countless and can make a good impact in society, we must also keep in mind their environmental effect, watching them to be friendly with it, and their ethics implications in society, ensuring the privacy of the users. Through the work we focused on developing two tools which can help people in different fields by creating more powerful systems, while reducing energy expenses and environmental impacts.

ANNEX B: ECONOMIC BUDGET

LABOUR COST (direct cost)

Hours	Cost/hour	Total
330	15 €	4.950 €

COST OF MATERIAL RESOURCES (direct cost)

	Purchase price	Use in months / Hours of use	Amortization (in years)	Total
Personal computer (including software)	2,000.00 €	7 months	5	233.33 €
Other equipment	300.00 €	7 months	5	35.00 €
Energy (Consumption of a RTX4090 \approx 450W)	0.1483 €/kWh	600 hours		40.00 €

TOTAL COST OF MATERIAL RESOURCES

308.33 €

GENERAL EXPENSES (indirect costs)

15%

over DC

788.75 €

INDUSTRIAL PROFIT

6%

over DC+IC

362.82 €

CONSUMABLES

Printing	20.00 €
Binding	10.00 €

SUBTOTAL BUDGET

6,439.91 €

VAT APPLICABLE

21%

1,352.38 €

TOTAL BUDGET

7,792.29 €

ANNEX C: TOOLS USED IN THE PROJECT

Along this section, the tools used for the development of this project are commented.

C.1 PYTHON

All the research and models have been developed using python. This language offers great support for data analysis and processing. Apart from the base tools, there are several libraries that helped the workflow.

- Pandas: Is an open-source library for easy data processing. It was used for loading, formatting, and saving the data used.
- NumPy: Allows for fast and versatile operations for N-dimensional arrays. Helped with the numerical processing of landmarks and arrays.
- Keras and Pytorch: Both are deep learning libraries for the creation, training, and evaluation of deep learning models.

C.2 WANDB (W&B)

It is an AI developer platform, with tools for training models, fine-tuning models, and leveraging foundation models [38]. It has a lot of different functionalities. At this project it was used for storing, representing and compare results. When integrating it into the model, it allows logging data of the model training into a defined workspace. This helps to the organization and comparison of the experiments.

The web interface allows for different ways of showing and comparing the results. It gives a table with all the parameters of each run and the results obtained but can also plot for each experiment the evolution of its metrics.

Another feature used is the sweep scheduling. With this function, a set of hyperparameters can be selected within an interval to search for. After preparing the execution, it trains the model with the different values given, searching the configuration that optimizes a given metric. It was incredibly useful for automatizing a set of experiments and analyzing its results, without the need to wait between each to run the next one.

In summary, the software was of significant assistance in the experimental phase of the project, facilitating the comparison and storage of results.