UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Systems and Computation

Heterogeneous Gesture Recognition based on Deep Learning for Natural Interaction in Mixed Reality Applications.

Master's Thesis

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

AUTHOR: Andreu Villar, Mario

Tutor: Martínez Hinarejos, Carlos David

ACADEMIC YEAR: 2022/2023

Abstract

The use of Mixed Reality (MR) has experienced an outstanding growth in recent years. Mixed Reality allows the users to have an immersive experience, but one in which they can see the real world around them and interact with the information and digital elements displayed on it, usually using translucent glasses where information is projected. However, this field is still relatively new and constantly evolving, so it is necessary to address the challenge of creating natural and comfortable experiences for the user. One of the areas of interest is communication and interaction with the elements of the Mixed Reality environment, through natural interaction methods such as gestures. Therefore, it is essential that MR devices can recognize and understand gestures made by users, allowing applications to react appropriately to these actions.

This work focuses on the development of a heterogeneous (both static and dynamic) Hand Gesture Recognition (HGR) system using Deep Learning techniques. The main objective is to create a classifier that can recognize and distinguish the gestures made by the user. In addition, the aim is to develop a complete system that allows real-time recognition, so that users can interact in a natural way with Mixed Reality environments using gestures. For model training, we use data captured using Microsoft HoloLens 2 Mixed Reality glasses, which provides spatial information about finger joints.

Through this study, we explore different Neural Network architectures and analyze their performance in accurate Hand Gesture Recognition. In addition, we present a self-developed dataset, considerably larger than previously published datasets in this field. Overall, this study provides a new perspective on heterogeneous gesture recognition for natural interaction in Mixed Reality applications, using Deep Learning techniques and a larger proprietary dataset. The results obtained are intended to highlight the feasibility and potential of this system to improve the user experience in MR environments, paving the way towards new types of interaction and the development of immersive and intuitive applications.

Key words: Deep Learning (DL); Mixed Reality (MR); Hand Gesture Recognition (HGR); Human-Computer Interaction (HCI); HoloLens 2; Joint tracking; Machine Learning (ML); Neural Networks (NN); Convolutional Neural Networks (CNN);

Resumen

El uso de la realidad mixta (MR) ha experimentado en los últimos años un crecimiento destacable. La realidad mixta permite al usuario una experiencia inmersiva, pero en la que puede ver el mundo real a su alrededor e interactuar con la información y elementos digitales que se muestran sobre el mismo, habitualmente utilizando unas gafas traslúcidas sobre las que se proyecta información. Sin embargo, este campo es aún es relativamente nuevo y está en constante evolución, por lo que es necesario abordar el desafío de crear experiencias naturales y cómodas para el usuario. Una de las áreas de interés es la comunicación e interacción con los elementos del entorno de realidad mixta a través de métodos de interacción natural como los gestos. Por lo tanto, es fundamental que los dispositivos de realidad mixta puedan reconocer y comprender los gestos realizados por los usuarios, permitiendo que las aplicaciones reaccionen de manera adecuada a estas acciones.

Este trabajo se centra en el desarrollo de un sistema de reconocimiento de gestos heterogéneos (tanto estáticos como dinámicos) de las manos, empleando técnicas de aprendizaje profundo. El objetivo principal es crear un clasificador que pueda reconocer y distinguir los gestos realizados por el usuario. Además, se busca desarrollar un sistema completo que permita el reconocimiento en tiempo real, de forma que los usuarios puedan interactuar de manera natural con entornos de realidad mixta mediante gestos. Para el entrenamiento del modelo se utilizan datos capturados mediante las gafas de realidad mixta Microsoft HoloLens 2, las cuales proporcionan información espacial sobre las articulaciones de los dedos.

A través de este estudio, exploramos diferentes arquitecturas de redes neuronales y analizamos su rendimiento en el reconocimiento preciso de los gestos de las manos. Además, presentamos un *dataset* propio, considerablemente más extenso que los conjuntos de datos previamente publicados en este campo. En conjunto, este trabajo proporciona una nueva perspectiva en el reconocimiento de gestos heterogéneos para la interacción natural en aplicaciones de realidad mixta, utilizando técnicas de aprendizaje profundo y un *dataset* propio de mayor tamaño. Los resultados obtenidos tratan de demostrar la viabilidad y el potencial de este sistema para mejorar la experiencia del usuario en entornos de realidad mixta, allanando el camino hacia nuevas formas de interacción y desarrollo de aplicaciones inmersivas e intuitivas.

Palabras clave: Aprendizaje profundo (DL); Realidad mixta (MR); Reconocimiento de gestos de la mano (HGR); Interacción Hombre-Computadora (HCI); HoloLens 2; Seguimiento de articulaciones; Aprendizaje automático (ML); Redes neuronales (NN); Redes neuronales convolucionales (CNN);

Resum

L'ús de la realitat mixta (MR) ha experimentat un creixement notable en els darrers anys. La realitat mixta permet a l'usuari una experiència immersiva, però en la qual pot veure el món real al seu voltant i interactuar amb la informació i els elements digitals que s'hi mostren, generalment utilitzant ulleres translúcides sobre les quals es projecta informació. No obstant això, aquest camp encara és relativament nou i està en constant evolució, per la qual cosa cal abordar el repte de crear experiències naturals i còmodes per als usuaris. Una de les àrees d'interés és la comunicació i interacció amb els elements de l'entorn de la realitat mixta mitjançant mètodes d'interacció natural, com ara els gestos. Per tant, és fonamental que els dispositius de realitat mixta puguen reconéixer i comprendre els gestos realitzats pels usuaris per permetre una interacció adequada amb les aplicacions.

Aquest treball es centra en el desenvolupament d'un sistema de reconeixement de gestos heterogenis (tant estàtics com dinàmics) de les mans, fent servir tècniques d'aprenentatge profund. L'objectiu principal és crear un classificador capaç de reconéixer i distingir els gestos realitzats pels usuaris. A més, es busca desenvolupar un sistema complet que permeta el reconeixement en temps real, per facilitar la interacció natural dels usuaris amb els entorns de realitat mixta mitjançant gestos. Per a l'entrenament del model s'utilitzen dades capturades mitjançant les ulleres de realitat mixta Microsoft HoloLens 2, les quals proporcionen informació espacial sobre les articulacions dels dits.

Per mitjà d'aquest estudi, s'exploren diferents arquitectures de xarxes neuronals i s'analitza el seu rendiment en el reconeixement precís dels gestos de les mans. A més, es presenta un conjunt de dades propi, considerablement més extens que els conjunts de dades prèviament publicats en aquest àmbit. En conjunt, aquest treball aporta una nova perspectiva en el reconeixement de gestos heterogenis per a la interacció natural en aplicacions de realitat mixta, fent servir tècniques d'aprenentatge profund i un conjunt de dades pròpies més gran. Els resultats obtinguts busquen demostrar la viabilitat i el potencial d'aquest sistema per millorar l'experiència dels usuaris en entorns de realitat mixta i obrin el camí a noves formes d'interacció i desenvolupament d'aplicacions immersives i intuïtives.

Paraules clau: Aprenentatge profund (DL); Realitat mixta (MR); Reconeixement de gestos de la mà (HGR); Interacció Home-Computadora (HCI); HoloLens 2; Seguiment d'articulacions; Aprenentatge automàtic (ML); Xarxes neuronals (NN); Xarxes neuronals convolucionals (CNN);

Contents

List of figures

List of tables

List of acronyms

- **AI**: Artificial Intelligence
- **ANN**: Artificial Neural Network
- **AR**: Augmented Reality
- **ASTID**: Área de Soporte Técnico a Proyectos de I+D — R&D Projects Technical Support Area
- **BN**: Batch Normalization
- **CCW**: Counter-ClockWise
- **CE**: Classification Error
- **CI**: Confidence Interval
- **CNN**: Convolutional Neural Network
- **CPU**: Central Processing Unit
- **CS**: Computer Science
- **CV**: Computer Vision
- **CW**: ClockWise
- **D**: Dynamic (gesture)
- **DA**: Data Augmentation
- **DDNet**: Double-feature Double-motion Network
- **DHG**: Dynamic Hand Gesture
- **DI**: Digital Imaging
- **DL**: Deep Learning
- **DoF**: Degrees of Freedom
- **DRAM**: Dynamic Random Access Memory
- **DSIC**: Departamento de Sistemas Informáticos y Computación — Department of Computer Systems and Computation
- **FC**: Fully Connected (layer)
- **FG**: Fine-Grained (gesture)
- **FN**: False Negative
- **FNR**: False Negative Rate
- **FP**: False Positive
- **FPR**: False Positive Rate
- **FPS**: Frames Per Second
- **FSM**: Finite State Machine
- **FullHD**: Full High Definition
- **GAP**: Global Average Pooling
- **GER**: Gesture Error Rate

- **GN**: Gaussian Noise
- **GPU**: Graphics Processing Unit
- **GRU**: Gated Recurrent Unit
- **HCI**: Human-Computer Interaction
- **HGR**: Hand Gesture Recognition
- **HMD**: Head-Mounted Display
- **HPU**: Holographic Processing Unit
- **ID**: IDentifier
- **IMU**: Inertial Measurement Unit
- **IP**: Internet Protocol
- **IQR**: InterQuartile Range
- **IR**: InfraRed
- **ITI**: Instituto Tecnológico de Informática
- **JCD**: Joint Collection of Distances
- **JPD**: Joint Pairs' Directions
- **LPDDR**: Low-Power Double Data Rate
- **LSTM**: Long Short-Term Memory
- **Mfast**: Motion (fast)
- **MIARFID**: Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital — Master's Degree in Artificial Intelligence, Pattern Recognition, and Digital Imaging
- **ML**: Machine Learning
- **MLP**: MultiLayer Perceptron
- **MP**: MegaPixel
- **MR**: Mixed Reality
- **Mslow**: Motion (slow)
- **NN**: Neural Network
- **P**: Periodic (gesture)
- **ParaRed**: Parameter Reduced (model)
- **PO**: Palm Orientation
- **PR**: Pattern Recognition
- **Q1**: First Quartile
- **Q2**: Second Quartile (a.k.a. median)
- **Q3**: Third Quartile
- **RAM**: Random Access Memory
- **ReLU**: Rectified Linear Unit
- **ResNet**: Residual Network
- **RGB**: Red Green Blue
- **RNN**: Recurrent Neural Network
- **ROC**: Receiver Operating Characteristic
- **SDG**: Sustainable Development Goal
- **SDK**: Software Development Kit

- **SHREC**: SHape REtrieval Contest
- **SiDi**: Sistemas Distribuidos y Cloud — Distributed and Cloud Systems
- **SoC**: System-on-Chip
- **ST**: STatic (gesture)
- **ST-GCN**: Spatial Temporal Graph Convolutional Network
- **STRONGER**: Simple TRajectory-based ONline GEsture Recognizer
- **SVM**: Support Vector Machine
- **TCN**: Temporal Convolutional Network
- **TCP**: Transmission Control Protocol
- **TN**: True Negative
- **ToF**: Time-of-Flight (sensor technology)
- **TP**: True Positive
- **TPR**: True Positive Rate
- **Transf**: Transformations (block)
- **UDP**: User Datagram Protocol
- **UFS**: Universal Flash Storage
- **UPV**: Universitat Politècnica de València
- **USB**: Universal Serial Bus
- **ValgrAI**: Valencian Graduate School and Research Network of Artificial Intelligence
- **VR**: Virtual Reality
- **VRAM**: Video Random Access Memory
- **XR**: Extended Reality
- **ZMQ**: ZeroMQ — Zero Message Queue
- **1D**: One-Dimensional
- **2D**: Two-Dimensional
- **3D**: Three-Dimensional

CHAPTER 1
Introduction

This chapter outlines the reasons behind choosing this topic as a final master's project, analyzing the objectives we pursued throughout the experimentation. Furthermore, we state how we have structured this report, which presents and consolidates the obtained results.

1.1 Motivation

The motivation for this project arises from the increasing adoption of Mixed Reality (MR) and the need to enhance natural user interaction in this environment [1]. As the use of Mixed Reality has gained popularity, there is a recognized importance in creating immersive and comfortable experiences for users. However, recognizing and understanding gestures performed by users remains a challenge [2].

Natural gestures provide an intuitive and powerful means of communication and interaction with elements in the Mixed Reality environment. Gestures can provide a more fluid and expressive means of communication, compared to other interaction methods such as voice commands or physical controllers [3].

Accurate gesture recognition is then essential to enable smooth and natural interaction between the user and the Mixed Reality application. This entails developing systems capable of reliably and real-time recognizing and distinguishing a wide range of user gestures.

Deep Learning techniques [4], such as Neural Networks, have demonstrated great potential in the field of Hand Gesture Recognition (HGR). These techniques can learn complex patterns from large datasets and can adapt to individual variations in gestures [5]. In this work, we will employ such techniques to train a classifier capable of real-time recognition and differentiation of user gestures.

However, a major challenge we face is the limited availability of suitable datasets with an adequate number of samples for training. For instance, the DHG14/28 dataset [6] consists of only 1400 samples of full hand gestures. Similarly, the SHREC22 dataset [7] offers an even smaller labeled set of 576 hand gesture samples.

To address the limited number of samples in these datasets, we have taken the initiative to create our own dataset, capturing the 3D joint points of hand articulations using a Mixed Reality device. By doing so, we aim to provide a solid and comprehensive foundation for the development and training of our Deep Learning models.

This project has been undertaken in collaboration with the Human-Computer Interaction (HCI) department at the *Instituto Tecnológico de Informática* (ITI). Initiated as a part

of the company internship during this master's degree program, this work evolved into a complete research. This hands-on participation has allowed an integration of theoretical knowledge into real-world applications, contributing to the exploration and refinement of HGR methodologies.

The development of this gesture recognition system will attempt to enhance the user experience in Mixed Reality environments, enabling more intuitive and natural interactions. This will pave the way for the development of immersive applications that allow the user to express themselves through hand gestures. Ultimately, this project is expected to contribute to the advancement and widespread adoption of Mixed Reality as a more enriching form of communication and working.

1.2 Objectives

The general objective of this work is to develop a heterogeneous gesture recognition system that allows real-time recognition within Mixed Reality applications, employing a Deep Learning approach.

More specifically, we can state the following objectives:

- To explore and analyze different Neural Network architectures for gesture recognition.
- To develop a comprehensive dataset for gesture recognition.
- To develop a gesture classifier model that accurately recognizes and classifies specific gestures from input data.
- To design and implement a binary classifier that can determine the presence or absence of a gesture within a given window of joint data.
- To create a complete system that extracts real-time windows of joint data, processes them using the gesture recognition model, and provides instantaneous gesture detection and classification.
- To evaluate the performance of the gesture recognition system in terms of accuracy, speed, and robustness, using appropriate evaluation metrics.
- To showcase the feasibility and potential of the developed system by integrating it into a basic Mixed Reality application, demonstrating its effectiveness in enabling natural and intuitive interactions.

1.3 Structure of the thesis

Throughout this document, we present the work carried out, explaining the decisions taken and the most relevant results.

In chapter 2, we start with a section of fundamental concepts. This work brings together two large areas of knowledge within Computer Science, the part related to Artificial Intelligence, Machine Learning, Pattern Recognition, etc. and the part related to Computer Graphics, Extended Reality, Human-Computer Interfaces, etc. In this chapter, we briefly explain the basic concepts of both parts that it is essential to keep in mind in order to be able to understand the content of this work.

Later, in chapter 3, we analyze the current state of the art, indicating the advantages and disadvantages of the existing methods that attempt to recognize hand gestures in a Mixed Reality context.

Thereafter, in chapter 4, we expose one of the main problems in this field: the absence of large datasets. To deal with this issue, we decided to create our own dataset. In this chapter we explain the whole process, indicating the steps taken to acquire the data samples, their cleaning and their basic statistics.

Afterward, in chapter 5, we present the experimentation carried out. We introduce the three major parts of this work: the gesture classifier, the gesture detector, and the final real-time detection and classification system. We also show a small proof-of-concept of how this system would work integrated in a Mixed Reality application.

Finally, chapter 6 is the culmination of this report, where we compile the definitive conclusions of the work carried out. We also propose possible improvements to achieve even better results, which could not be examined for being outside the length of the project.

CHAPTER 2
Basic concepts

In this chapter, we aim to provide a concise and simplified explanation of the fundamental concepts related to our work. For a more in-depth background, please refer to the references mentioned in each point. We divide these concepts into two parts: those related to Machine Learning (ML) and those related to Extended Reality (XR).

The ML-related concepts serve as the foundation for our gesture recognition system. We introduce key elements such as Neural Networks, MultiLayer Perceptron, Convolutional Neural Networks, etc. These concepts are essential for understanding the underlying techniques employed in this project.

On the other hand, the XR-related concepts are crucial for comprehending the context in which our system is designed to operate. We present an overview of Extended Reality (XR), which encompasses Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). We delve into the definitions and characteristics of each concept, highlighting their distinctions and applications. Additionally, we introduce key concepts such as Head-Mounted Display (HMD), a device that plays a significant role in delivering immersive XR experiences and serves as a crucial component for capturing data in our project.

2.1 Machine Learning-related concepts

Machine Learning (ML)

Machine Learning is a field of study that focuses on developing algorithms and models that enable computers to learn from and make predictions based on data [8]. It involves the creation of mathematical models and algorithms that can automatically improve their performance through experience.

Neural Networks (NN)

Neural Networks are computational models used in Machine Learning, inspired by the structure and functioning of the human brain [8]. They consist of interconnected nodes, or artificial neurons, organized in layers. Neurons receive inputs, apply transformations using mathematical functions, and pass values to the next layer. Neural Networks can learn patterns and relationships in data by adjusting the weights of connections between neurons through a process called training [8]. Figure 2.1 shows a basic representation of its architecture.

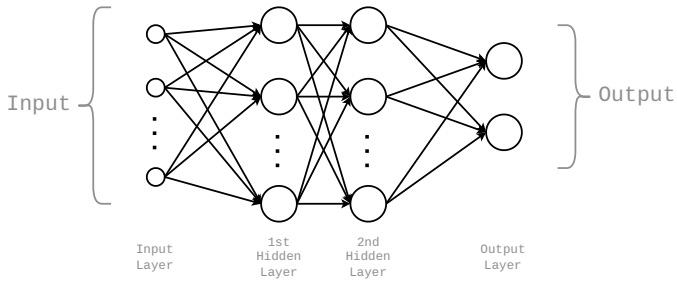


Figure 2.1: Example of the basic architecture of a Neural Network (adapted from [4]).

Deep Learning (DL)

Deep Learning is a subfield of Machine Learning that focuses on training Neural Networks with multiple layers to learn hierarchical representations of data [8]. These Neural Networks, often referred to as Deep Neural Networks, have shown remarkable success in various tasks, such as Computer Vision [9] and Automatic Speech Recognition [10]. Deep Learning algorithms usually learn directly from raw data, automatically extracting relevant features and patterns without the need for manual feature engineering. The depth of the networks allows them to model complex relationships and capture intricate structures in the data, leading to improved performance and higher levels of abstraction [8].

MultiLayer Perceptron (MLP)

MLP is a type of Neural Network where information flows in one direction, from the input layer through one or more layers (called hidden layers) to the output layer [4]. Each neuron in an MLP is connected to every neuron in the subsequent layer, making it capable of learning complex patterns and non-linear relationships in the data [4]. Figure 2.1 is more concretely an MLP with two hidden layers and an undefined number of neurons per layer.

Convolutional Neural Networks (CNNs)

CNNs [8] are a type of Neural Network architecture designed for processing grid-like data, such as images. They employ layers that convolve filters over the input data to extract local features. CNNs have been successful in various Computer Vision tasks, including image classification [11], object detection [12], and image segmentation [13]. Figure 2.2 shows a representation of its architecture.

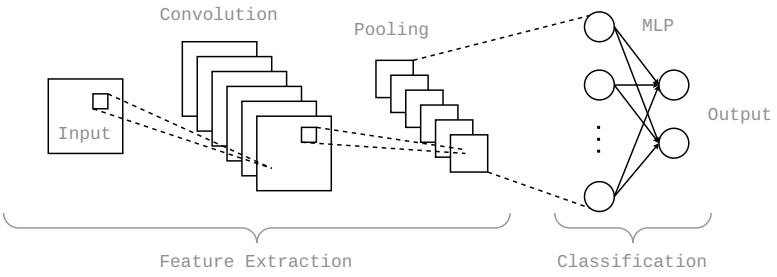


Figure 2.2: Example of the basic architecture of a CNN (adapted from [8]).

ResNet

ResNet, short for Residual Network, is a Deep Neural Network architecture that introduced residual connections [8]. Residual connections are skip connections that enable the network to learn residual mappings, capturing the difference between the input and the output of a layer. By using these connections, ResNet allows for training of much deeper networks without suffering usual problems (vanishing gradients [8]). This architecture has achieved remarkable performance in Computer Vision tasks, such as image classification [9], by effectively handling the challenges of training Deep Neural Networks. Figure 2.3 shows an example of a residual connection.

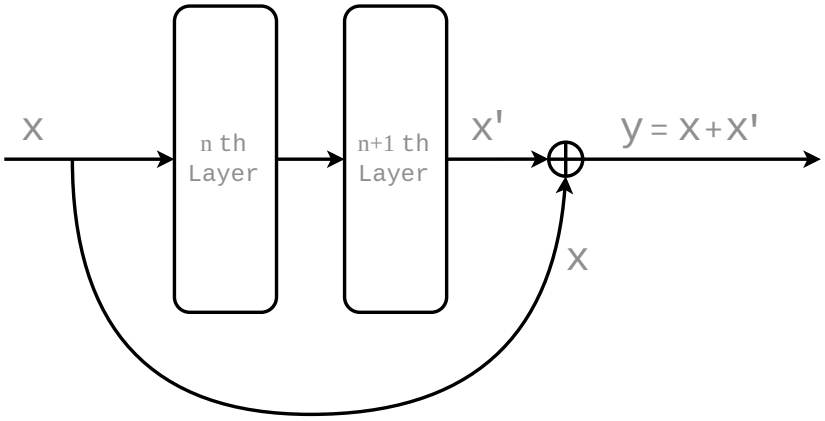


Figure 2.3: Illustration of a residual connection (adapted from [8]).

1D Convolutional Neural Networks (1D-CNNs)

1D-CNNs are Convolutional Neural Networks specifically designed for processing one-dimensional data, such as time series or sequence data [14, 15]. They apply convolutional operations along the temporal dimension to capture local patterns and extract features. 1D-CNNs have been widely used in applications such as Automatic Speech Recognition and Natural Language Processing [16].

Data Augmentation (DA)

Data Augmentation is a technique used to artificially increase the size and diversity of a dataset by applying transformations or modifications to existing data samples [17]. This technique usually helps to reduce overfitting and improves the generalization capability of Machine Learning models by exposing them to different variations of the original data [17].

Evaluation metrics

Evaluation metrics are measures used to assess the performance of Machine Learning models [5]. They provide quantitative indications of how well a model performs on a specific task. Common evaluation metrics usually include accuracy, recall, F1-score, and mean squared error, depending on the nature of the problem being addressed [5]. These metrics help in comparing and selecting the best-performing models for a given task. In section 5.1.1 we explain in detail those used in this work.

ROC curve

In the context of Machine Learning and binary classification tasks, the Receiver Operating Characteristic (ROC) curve is a graphical representation of a model's performance in distinguishing between two classes. It plots the True Positive Rate (sensitivity) against the False Positive Rate (1 - specificity) at various decision thresholds [8].

The ROC curve is a valuable tool for evaluating the trade-off between a model's sensitivity and specificity. Sensitivity measures the model's ability to correctly identify positive instances, while specificity measures its ability to correctly identify negative instances [8]. By adjusting the decision threshold, one can achieve different levels of sensitivity and specificity.

The ROC curve is particularly useful when comparing multiple models or algorithms for binary classification. A model with a curve closer to the top-left corner of the plot is considered better as it achieves higher sensitivity while keeping false positives low [8]. An example of a ROC curve can be seen in Figure 5.8.

Fine-tuning and transfer learning

Fine-tuning and transfer learning are techniques in Machine Learning that involve using knowledge gained from one task or domain to improve the learning and performance on a different, but related, task or domain [8].

In particular, fine-tuning is a specific application of transfer learning. It involves taking a pre-trained model, typically a Deep Neural Network, that has already learned valuable features from a large dataset, and further training it on a new, related dataset. The primary goal of fine-tuning is to adapt the pre-trained model to perform well on a specific task or dataset without training it from scratch [8]. In fine-tuning, the pre-trained model serves as a starting point. Additional training is performed using the new dataset. Some layers of the model may be frozen, meaning they are not updated, while others are allowed to be modified. This approach takes advantage of the knowledge and features learned by the model on the original dataset, which can be highly beneficial when the new dataset is small or related to the original task [8].

Transfer learning, on the other hand, is a broader concept that encompasses various strategies for re-using knowledge across tasks or domains to improve Machine Learning models' performance. The main type of transfer learning is fine-tuning a pre-trained model, as explained earlier. However, transfer learning can also involve using features extracted from earlier layers of a Neural Network or adapting model parameters for the new task. The key idea is to avoid starting the learning process from scratch when solving a related problem by transferring the knowledge acquired from a source task or domain [8].

Lambda layer

In the context of Neural Network architectures, a "lambda layer" is a layer used to perform specific manually defined transformations on input data before it is fed into the Neural Network. These transformations can include mathematical operations such as normalization, scaling, color space conversion, or any other processing that prepares the input data for subsequent processing by the network [18]. For example, when working with images, a lambda layer might resize all images to a specific size, adjust their brightness, or convert them to a particular color scale before they are used as input for the network. This allows the Neural Network to work with data that is in the correct format and scale, which can significantly improve its performance and accuracy in Machine Learning tasks [18].

Embedding

In the context of Machine Learning, an embedding refers to a learned representation of data that captures essential features or relationships, usually in a lower-dimensional space [8]. It is commonly used to transform high-dimensional data, such as text or images, into a more compact and meaningful representation that can facilitate subsequent analysis or modeling tasks [8].

2.2 Extended Reality-related concepts

Extended Reality (XR)

Extended Reality is an umbrella term that encompasses various immersive technologies, including Virtual Reality (VR) [19], Augmented Reality (AR) [19], and Mixed Reality (MR) [20]. XR refers to the spectrum of experiences that blend the physical and virtual worlds, enabling users to interact with digital content and information in a more immersive and realistic manner. The "X" in XR can be seen as a placeholder for V(R), A(R), or M(R) [21] to indicate the specific technology being used.

Virtual Reality (VR)

Virtual Reality is a technology that creates a simulated, computer-generated environment in which users can fully immerse themselves [19]. By wearing a Head-Mounted Display (HMD), users are visually and audibly isolated from the real world and are presented with a fully virtual environment that surrounds them. VR provides an immersive and interactive experience, where users can explore and interact with the virtual environment.

Augmented Reality (AR)

Augmented Reality is a technology that overlays digital content onto the real world, enhancing the user's perception and interaction with the environment [19]. Unlike Virtual Reality, AR does not isolate users from the real world entirely. Instead, users view the real world through a screen or transparent display, such as a smartphone, with virtual elements seamlessly integrated into the real-world environment. AR enhances the user's perception by overlaying information, graphics, or objects onto the physical surroundings.

Mixed Reality (MR)

Mixed Reality combines elements of both Virtual Reality and Augmented Reality to create an experience where users can interact with and manipulate both virtual and real-world objects simultaneously [20]. MR allows virtual content to interact with the physical environment, providing a more seamless integration of digital and real-world elements. Users can see and interact with virtual objects that appear to coexist with the real world, enabling more immersive and interactive experiences.

Milgram Continuum

The Milgram Continuum [20] is a theoretical concept developed by researcher Paul Milgram that describes a scale ranging from pure physical reality to complete Virtual Reality. Within this continuum, different levels of immersion exist, ranging from purely physical environments, through AR, to full VR.

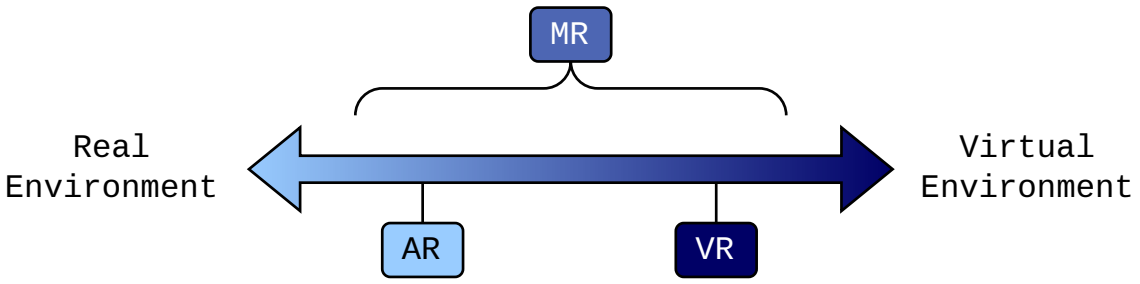


Figure 2.4: The Milgram Reality-Virtuality Continuum (adapted from [20]).

This concept is relevant for understanding the context in which Extended Reality operates. While VR provides a fully immersive experience in a computer-generated virtual environment, AR just overlays digital information onto the real physical environment, visible only through a screen. MR considers both, allowing for a more integrated interaction between virtual and real elements. Figure 2.4 shows a representation of this spectrum.

By considering the Milgram Continuum, one can appreciate how XR technologies are situated at different points along the spectrum, offering varying degrees of immersion and fusion between the virtual and the real environment. This has important implications for user interaction, the way environments are perceived, and the effects that can be achieved by blending digital and physical elements in Extended Reality applications.

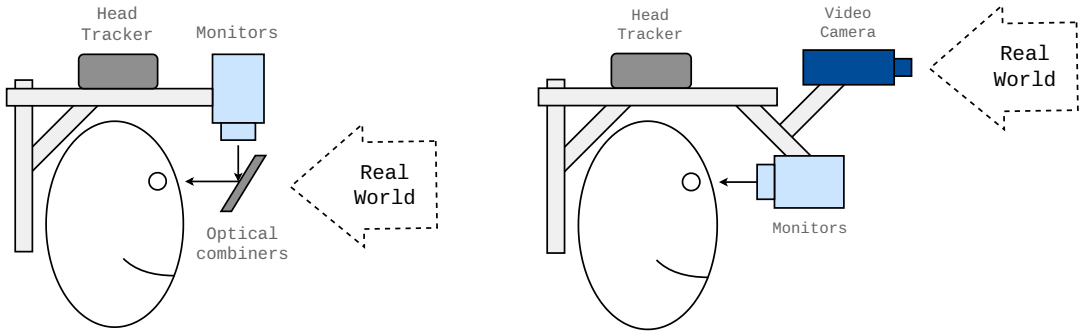
Head-Mounted Display (HMD)

A Head-Mounted Display is a wearable device that typically resembles goggles or glasses and it is worn on the head to deliver Virtual or Mixed Reality experiences. HMDs consist of display screens, sensors, and sometimes audio devices, providing users with a visual and auditory immersion into virtual or mixed environments.

In the case of MR, the HMD allows users to see the real world through a screen or transparent display while overlaying virtual elements. Figure 2.5a shows a basic schema of a MR-HMD. As an example of MR-HMD devices, we can mention the Microsoft HoloLens 2[1].

On the other hand, we have devices as the Oculus Quest Pro[2], designed specifically for VR experiences. In this case, it fully immerse users in virtual environments, blocking out the real world and providing a purely digital visual and auditory experience.

[1] https://www.microsoft.com/hololens
[2] https://www.meta.com/es/quest/quest-pro



(a) Optical see-through HMD conceptual diagram (MR-HMD devices)

(b) Video see-through HMD conceptual diagram (VR-HMD devices)

Figure 2.5: Different types of HMD devices to simulate MR experiences (adapted from [22]).

Passthrough

In the context of Extended Reality, passthrough (also known as video see-through) refers to the technique used in VR headsets, that do not have translucent displays, to simulate Mixed Reality experiences. In these devices, passthrough utilizes one or more cameras to capture the real-world view from the user's perspective and then displays it on the headset's opaque screens [23, 24]. Figure 2.5b shows a basic schema of a VR-HMD simulating a MR experience.

CHAPTER 3
State of the art

In this chapter, we delve into the state of the art in Hand Gesture Recognition (HGR), exploring the advancements and approaches that have shaped this evolving field. Hand gestures play a fundamental role in human communication and interaction [3], and take advantage of their potential within Mixed Reality environments. For this reason, HGR has become an area of significant research interest [25].

Exploring methodologies and devices utilized for HGR, we seek to better understand the current panorama, which will serve as the basis for our research on developing a Hand Gesture Recognition system using 3D joint data and Deep Learning techniques. Through this literature analysis, we seek to identify challenges, opportunities, and potential solutions in HGR, ultimately enhancing natural interactions in MR applications.

3.1 Hand Gesture Recognition (HGR)

Hand Gesture Recognition is a prominent research area within the field of Computer Vision (CV) and Human-Computer Interaction (HCI) [26]. It involves the automated interpretation and understanding of hand movements and configurations to infer meaningful gestures and commands. HGR has gained great importance due to its potential to revolutionize various applications, including Extended Reality, Sign Language Recognition, and Human-Robot Interaction [25].

3.1.1. Importance and applications

The relevance of HGR lies in its ability to enable intuitive and natural interactions between humans and machines. As technology progresses towards more immersive and interactive experiences, the need for efficient gesture-based interfaces becomes paramount. HGR facilitates seamless communication between users and computers, removing the barriers imposed by traditional input devices like keyboards and mice [25].

In Virtual Reality applications, HGR allows users to interact with virtual objects and environments using natural hand movements, enhancing the sense of presence and immersion. Moreover, it offers new possibilities in the field of Mixed Reality, where real world and virtual elements blend harmoniously, enabling novel and compelling user experiences [27].

Another crucial application of Hand Gesture Recognition is in Sign Language Recognition [28]. By accurately interpreting sign gestures, HGR systems can bridge the communication gap between the hearing-impaired community and the broader population, fostering inclusive communication and accessibility [28].

Additionally, HGR plays a vital role in Human-Robot Interaction, enabling robots to perceive and respond to human gestures effectively. This capability is particularly relevant in scenarios where verbal communication may be impractical or insufficient, such as noisy environments or situations where language barriers exist [27].

As it can be observed, Hand Gesture Recognition holds significant potential in various applications [25, 7]. This thesis seeks to contribute to the development of a gesture recognition system within Mixed Reality HMDs. The proposed system aims to accurately detect user gestures, enabling seamless interactions with the application running on the glasses, which can respond accordingly to the recognized gestures.

3.1.2. Approaches to Hand Gesture Recognition

Hand Gesture Recognition can be approached through different methods, primarily using RGB-based vision [26] or 3D sensor-based data [27], each presenting distinct advantages and difficulties.

RGB-based vision methods for HGR

RGB-based vision utilizes video or image data to recognize hand gestures. While this approach has been widely explored and can work effectively for certain gestures [29, 30], it may encounter difficulties with background and foreground segmentation, lighting variations, and limited depth information [26]. Precise depth information is essential for recognizing gestures involving movements towards or away from the camera, making RGB-based methods more challenging for such scenarios.

3D sensor-based methods for HGR

On the other hand, the 3D sensor-based data approach leverages depth information to represent hand gestures in a three-dimensional space [7]. These methods face issues like noise and artifacts in depth data, sensor calibration, and limited hand coverage, but they enable a more accurate representation of complex gestures that involve depth-based movements [25].

For this research, we have chosen to adopt the 3D sensor-based approach. By taking advantage of working with 3D positions, we anticipate a more efficient and accurate gesture recognition system. 3D positions represent the spatial coordinates of key hand articulation points, which results in a much lower data dimensionality compared to RGB images. This reduction in data complexity can lead to more efficient processing and faster computation times during gesture recognition, which helps to achieve real-time performance of interactive MR applications. Also, by capturing depth information, this approach facilitates the recognition of gestures involving different hand positions and orientations, which improves the overall user experience.

3.1.3. Challenges and limitations

Despite the promising potential of HGR, developing accurate and robust recognition systems presents several significant challenges [26]. The main obstacle lies in the variability and complexity of hand gestures. Human hands can produce an extensive range of gestures, each displaying subtle differences in appearance, making it challenging to discern between them accurately.

Another pressing challenge in HGR is achieving real-time processing in interactive applications [31]. The demand for high frame rates and low-latency responses is crucial for delivering seamless user experiences. However, this goal introduces a substantial computational burden. Balancing real-time performance while maintaining gesture recognition accuracy remains an ongoing challenge, promoting the exploration of efficient algorithms and hardware optimization techniques [7].

Furthermore, the requirement for large and diverse annotated datasets (or pre-trained models) poses yet another limitation in HGR research. Building such datasets is a time-consuming and resource-intensive process, involving the collection and annotation of a vast number of hand gesture samples [7]. The availability of large datasets is crucial for training Deep Learning models effectively [5], but creating them can be a significant obstacle. Addressing this challenge calls for innovative data collection strategies and collaborative efforts to facilitate the development of more robust and sophisticated HGR systems.

3.2 Mixed Reality devices for hand gesture capture

In the field of Hand Gesture Recognition, a wide range of MR devices can be used [25], each offering unique capabilities to capture hand movements and joints. These devices mainly include depth-sensing cameras, Head-Mounted Displays, and specialized Mixed Reality gloves.

3.2.1. Comparative analysis of hand gesture capture devices

Depth-sensing cameras, such as Leap Motion[1] or Kinect[2], capture detailed 2D or 3D point clouds of the hand, providing high-precision data for gesture recognition. Their cost-effectiveness also makes them an attractive option for HGR applications, especially in budget-constrained scenarios [25].

On the other hand, Head-Mounted Displays (HMDs) offer a distinct advantage by incorporating built-in cameras and sensors that facilitate more natural and intuitive interactions. By eliminating the need for external tracking systems or markers, HMDs simplify the setup and enhance the user experience during gesture capture and MR interactions [7]. Their main disadvantages are that they are more expensive and sometimes uncomfortable, especially for prolonged use.

Another noteworthy option is Mixed Reality gloves, equipped with sensors like IMUs (Inertial Measurement Units) and flex sensors. These wearable devices enable precise hand articulation capture, offering high accuracy in tracking intricate hand movements and gestures. A potential disadvantage of these gloves is their impact on user comfort and ease of use, as they require users to wear additional equipment during interactions in MR environments [25].

In our study, we have chosen to utilize Head-Mounted Displays (HMDs) and, specifically, the Microsoft HoloLens 2[3] as the MR device for Hand Gesture Recognition. The decision was based on several factors, mainly due to the availability of the HoloLens 2 devices we already possessed, and because they allow us to seamlessly integrate the Mixed Reality applications and the capture of hand gestures with the headset's sensors. In addition, this device offers a great track record in the state of the art of Mixed Reality [32],

[1] https://leap2.ultraleap.com/leap-motion-controller-2
[2] https://azure.microsoft.com/es-es/products/kinect-dk
[3] https://microsoft.com/hololens

good support from Microsoft, and a well-established SDK for application development[4], making it a suitable choice for our research on developing an advanced HGR system that can be used by MR apps.

3.2.2. Microsoft HoloLens 2 description and specifications

The Microsoft HoloLens 2 is an advanced and cutting-edge Mixed Reality headset designed to seamlessly blend the digital and physical worlds, offering immersive and interactive experiences [32]. In this section, we present its technical characteristics[5] to be highlighted for the realization of this work.

Hardware specifications

The HoloLens 2 features a custom-built Microsoft Holographic Processing Unit 2.0 (HPU), specifically designed to handle complex holographic computations, real-time tracking, and gesture recognition. It also incorporates a Qualcomm Snapdragon 850 System-on-Chip (SoC), 4-GB LPDDR4x system DRAM, 64-GB UFS 2.1 of storage, Wi-Fi 5 (802.11ac 2x2), Bluetooth 5.0, and USB Type-C for connectivity.

Optics and display

The Microsoft HoloLens 2 features a display system that incorporates see-through holographic lenses, also known as waveguides, allowing users to overlay holographic content onto their physical environment. The device is equipped with 2K (2048 x 1080 pixels) 3:2 light engines, providing high-resolution visuals, and a holographic density of over 2.5k radiants, ensuring detailed holographic imagery. Additionally, the HoloLens 2 utilizes eye-based rendering technology to optimize the display for 3D eye position, dynamically adjusting the rendered content based on the user's eye movements for improved visual fidelity and a more natural Mixed Reality experience.

Sensors

The Microsoft HoloLens 2 is equipped with various sensors, including 4 visible light cameras for head tracking, 2 IR cameras for eye tracking, a 1-MP Time-of-Flight (ToF) depth sensor for depth perception, and an IMU consisting of an accelerometer, gyroscope, and magnetometer for motion tracking. Additionally, the device features an 8-MP camera for capturing still images and recording FullHD (1080p) at 30 FPS video.

Environment understanding

The Microsoft HoloLens 2 offers advanced environment understanding capabilities, including 6 DoF (Six Degrees of Freedom) tracking, enabling world-scale positional tracking for users' movements in physical spaces. The device also features spatial mapping, allowing real-time creation of an environment mesh, which aids in overlaying virtual holographic content onto the real world seamlessly.

Portability

The Microsoft HoloLens 2 offers favorable ergonomics and portability features. The device is designed to accommodate users who wear glasses, ensuring a comfortable

[4] https://microsoft.com/hololens/developers
[5] https://learn.microsoft.com/en-us/hololens/hololens2-hardware

fit for extended wear. With a weight of 566 grams, the HoloLens 2 remains relatively lightweight, contributing to its ease of use. Regarding battery life, the device provides approximately 2 to 3 hours of active use, allowing for extended periods of MR interaction before requiring recharging.

3.3 Previous research on HGR using DL and 3D joints

Hand Gesture Recognition using Deep Learning techniques and 3D joint representations has attracted increasing interest [7], but the exploration of this approach is relatively less extensive compared to Computer Vision (CV) methods. The focus on CV-based approaches has resulted in a more comprehensive body of research, while the potential of 3D joint representations remains relatively less explored [25].

Traditional methods such as Support Vector Machines (SVM), Random Forests, and dissimilarity-based classifiers have also been explored [33, 34]. Nevertheless, Neural Networks have become the prevailing approach in this domain with promising results. Recurrent Neural Networks, particularly LSTM units, have been popularly utilized for handling time-series data like hand pose streams, as demonstrated in [35]. The Deep Gesture Recognition Utility [36], employing stacked GRUs and a global attention model, has proven to be both efficient and effective [36].

Other approaches have noted that very simple 1D Convolutional Neural Networks with motion summarization modules can achieve state-of-the-art results with reduced computational complexity [31]. Despite these advancements, open challenges still exist in the domain of HGR, warranting further investigation into 3D joint-based Deep Learning methods.

Studies employing 3D joint data for HGR often utilize a variety of devices, especially those with lower cost, for data capture. However, this diversity in devices can lead to differences in the representation of hand gestures since each device may consider different key points in the hand [6, 37, 7]. As a consequence, achieving a standardized and consistent representation of hand joints becomes challenging, potentially affecting the generalizability of results across studies.

One notable domain where significant advancements have been made in HGR using 3D joints is in the SHREC (SHape REtrieval Contest) community[6]. Specifically, the "SHREC 2022 Track on Online Detection of Heterogeneous Gestures" [7] stands out as a relevant benchmark in the field. Although the track poses certain limitations due to the absence of publicly available test data[7], it serves as a valuable resource for understanding the methodologies used by participants to tackle the HGR problem.

The main advantage of focusing on this competition is that researchers employed the same capture device we use, the Microsoft HoloLens 2, and utilized also the same representation of hand joints for training (see Section 4.1.1 for joints' description) [7]. While we are not able to make any direct comparisons of the performance of the models due to the lack of labeled test data, studying the approaches employed by various participants provides valuable insights into the state of the art techniques and inspires potential improvements for developing advanced Hand Gesture Recognition systems in our research.

In the subsequent sections, we delve into the primary approaches used, all of which revolve around DL methods and 3D joint representations.

[6] https://www.shrec.net/
[7] https://univr-vips.github.io/Shrec22/#dataset

3.3.1. Spatial Temporal Graph Convolutional Networks for HGR

One of the proposed approaches to address Hand Gesture Recognition is through the utilization of the "Two-Stream" Spatial Temporal Graph Convolutional Network (2ST-GCN) [7]. This architecture is an extension of the Spatial Temporal Graph Convolutional Network (ST-GCN) [38], which is specifically designed for gesture detection in video data. The 2ST-GCN model employs a two-stream design to process both spatial and temporal information. The first stream focuses on understanding the spatial relationships between key hand points (i.e. the joints) within individual frames, while the second stream analyzes temporal dynamics by observing how these hand points move over time across consecutive frames. By incorporating graph convolution, the model effectively captures complex spatial and temporal patterns, enabling accurate and robust Hand Gesture Recognition.

3.3.2. Temporal Convolutional Networks for HGR

Another proposed approach to tackle Hand Gesture Recognition is to utilize Temporal Convolutional Networks (TCN) [7]. Specifically, the Causal TCN variant proposed is a specialized CNN designed for processing sequential data in a causal order. This architecture incorporates cascading convolutional layers to learn patterns of varying temporal lengths within the sequence, allowing for the modeling of long-term temporal dependencies. Unlike traditional Recurrent Neural Networks (RNN), Causal TCNs overcome limitations on the length of temporal dependencies they can capture, making them particularly advantageous for predicting long-term time series events.

3.3.3. Transformer Network + Finite State Machine for HGR

An additional strategy proposed to address Hand Gesture Recognition is the combination of a Transformer network with a Finite State Machine (FSM) [7]. The FSM consists of four states designed to detect the initiation, middle, and completion stages of the gesture, along with an additional state for verification. Complementing the FSM, the network comprises two main components: the first employs transformers to generate a gesture embedding, while the second utilizes a Fully Connected (FC) layer to classify the gestures into 17 distinct classes, including 16 specific gestures and a class for non-gestures.

3.3.4. 1D Convolutional Neural Networks for HGR

Finally, another proposed approach is to utilize the STRONGER (Simple TRajectory-based ONline GEsture Recognizer) network [7, 27], which is based on a modified version of the DDNet (Double-feature Double-motion Network) architecture [31]. DDNet is a Deep Neural Network designed for directional signal classification, such as images, audio, and sensor data, among others. DDNet's core concept is to leverage directional data that contains valuable information in multiple orientations. The network processes and combines input signals in various directions using directional filters, followed by convolutional and pooling layers that reduce the dimensionality of the learned features. DDNet's unique advantage lies in its ability to capture specific directional patterns that are challenging for conventional Neural Networks to capture effectively. This makes it particularly valuable for applications involving directional signals, such as gesture classification [7].

3.4 Datasets used in previous research

In this section, we explore the datasets used in the literature for Hand Gesture Recognition. We describe them and analyze their characteristics, sizes, limitations, and potential for improvement.

3.4.1. Characteristics and size of the datasets

SHREC22 dataset

The SHREC22 dataset[8] includes 288 sequences of hand gestures, each containing a variable number of gestures ranging from 3 to 5 per sequence. The dataset is divided into a training set, with provided annotations, and a test set, which lacks annotations due to its use in a competition. Hence, only 144 usable samples are available (576 total gestures). The data was recorded using Microsoft HoloLens 2, capturing 26 points of interest on the hand's joints [7].

Dynamic Hand Gesture 14/28 dataset

This dataset[9] consists of 1400 sequences where 14 hand gesture classes are performed in two ways (2800 in total): using one finger and using the whole hand. Each gesture is executed five times by 20 right-handed participants. The sequences include depth images and 22 joint coordinates in both 2D depth image space and 3D world space, forming a complete hand skeleton. The Intel RealSense short-range depth camera captures the dataset at 30 frames per second, with a resolution of 640x480 for depth images. The gesture lengths range from 20 to 50 frames and 22 points of interest on the hand joints are captured [6].

SHREC21 dataset

The SHREC21 dataset[10] comprises 180 sequences of hand gestures, carefully planned to include 3 to 5 gestures per sequence, supplemented with semi-random hand movements labeled as non-gesture. The original dictionary contains 18 gestures, categorized into static gestures characterized by a fixed hand pose, and dynamic gestures characterized by hand and joint trajectories. However, a gesture was later removed from the dataset due to potential conflicts, leaving 17 gesture classes. The dataset provides an annotated test sequence. Gesture trajectories were captured using LeapMotion sensors at 50 FPS, providing 20 points of interest with both positional coordinates and quaternions [37].

3.4.2. Limitations and potential improvements in existing datasets

One primary limitation in existing datasets is the scarcity of samples (all have 100 or less samples per class), which hinders the application of Deep Learning techniques effectively [5]. Additionally, the variation in the number and types of joints captured by each dataset poses a challenge for direct adoption in certain applications, requiring complex adaptations to maintain consistency in the gesture representation.

[8] https://univr-vips.github.io/Shrec22/#dataset
[9] http://www-rech.telecom-lille.fr/DHGdataset/
[10] https://univr-vips.github.io/Shrec21/#revision

As well as the fact that there are not too many datasets, and those that exist are quite small, we can also highlight the lack of pre-trained models available in the literature. While pre-trained models have become increasingly prevalent in various Computer Vision and Natural Language Processing tasks [8], we have not encountered any publicly available pre-trained models specifically tailored for Hand Gesture Recognition using 3D joint data or similar.

Future improvements could focus on expanding the datasets with a larger number of samples, allowing for a more suitable training of DL models. Efforts to standardize the representation of hand joints across datasets could also enhance cross-dataset compatibility and facilitate a more seamless integration of the data for robust and accurate Hand Gesture Recognition models.

3.5 Conclusions of the state of the art

3.5.1. Summary of key findings, gaps, and areas for improvement

Through our exploration of the state of the art in Hand Gesture Recognition, several key findings have emerged. We identified a greater focus on Computer Vision approaches using RGB images, while the utilization of 3D joint data is relatively less explored. In this latter field, the trend seems to be towards CNN-1D networks as they are more efficient, simpler and also achieve great results.

The availability of large and diverse annotated datasets still remains a challenge, hindering the application of Deep Learning techniques to their full potential. Additionally, the lack of pre-trained models tailored for HGR using 3D joint data is also a notable limitation.

Furthermore, we observed variations in the representation of hand joints among different datasets, which may complicate the direct adaptation of models across datasets. Hence, training with a specific dataset may pose challenges in adapting the model to Mixed Reality devices that utilize a different number of joints.

3.5.2. Relevance of the thesis within the current research context

In the context of the state of the art, our thesis is justified by several distinct objectives. Firstly, our research focuses on exploring the potential of Deep Learning with 3D joint data for Hand Gesture Recognition, an approach that has received relatively less attention compared to conventional Computer Vision techniques, although it offers significant advantages.

Secondly, we intend to address the limitation of existing datasets by creating a significantly larger and more diverse dataset specifically tailored for HGR using 3D joint data. This dataset will serve as a valuable resource for training and evaluating Deep Learning models, ultimately contributing to the advancement of gesture recognition systems.

Lastly, the relevance of our thesis is underscored by our emphasis on designing the Hand Gesture Recognition system with compatibility for Mixed Reality devices. We envision a system that can be readily integrated into MR glasses, enabling natural and intuitive interactions with virtual environments. By aligning our research with the constraints and requirements of Mixed Reality devices, we aim to facilitate the practical implementation and real-world applicability of our gesture recognition solution.

CHAPTER 4
Dataset creation

In this chapter, we present the process of creating the dataset for Hand Gesture Recognition (HGR). The necessity for developing a new dataset arises from the limited availability of datasets in the current state of the art, particularly those with a substantial number of samples. Given the utilization of Deep Learning techniques, a large amount of data is essential to achieve optimal model performance and generalization.

This chapter provides a comprehensive overview of the dataset creation process, covering various crucial aspects. Firstly, we detail the data capture process using a HoloLens 2 Mixed Reality device. Thereafter, we discuss the different classes of the dataset and the automatic labeling of gestures during the data capture process. In addition, we provide an analysis of the demographics of the participants who took part in the data capture.

Furthermore, the dataset undergoes a detailed cleaning process to remove any noise or erroneous data, ensuring its quality and reliability. We outline the applied cleaning techniques and the identification and handling of noisy data to achieve a high-quality dataset. Finally, we present some key statistics on the dataset, including the distribution of gestures among the various categories, ensuring a balanced representation of the classes.

4.1 Data collection

In this section, we delve into the various aspects of data collection for the HGR dataset. We explore the key elements related to the data capture process using the Microsoft HoloLens 2 Mixed Reality device, which enables the real-time capture of 3D hand joint points.

4.1.1. Joints' description

Joints, also known as points of articulation, are three-dimensional data points that represent the position (x, y, z) of various key articulation points in the hands. By capturing these joints, we can effectively track and analyze hand movements and gestures, facilitating the development of accurate gesture recognition models.

Figure 4.1 presents the different joints captured, where each joint location on the hand can be seen. Each identifier is cross-referenced to those in Table 4.1, where a description of each joint is provided.

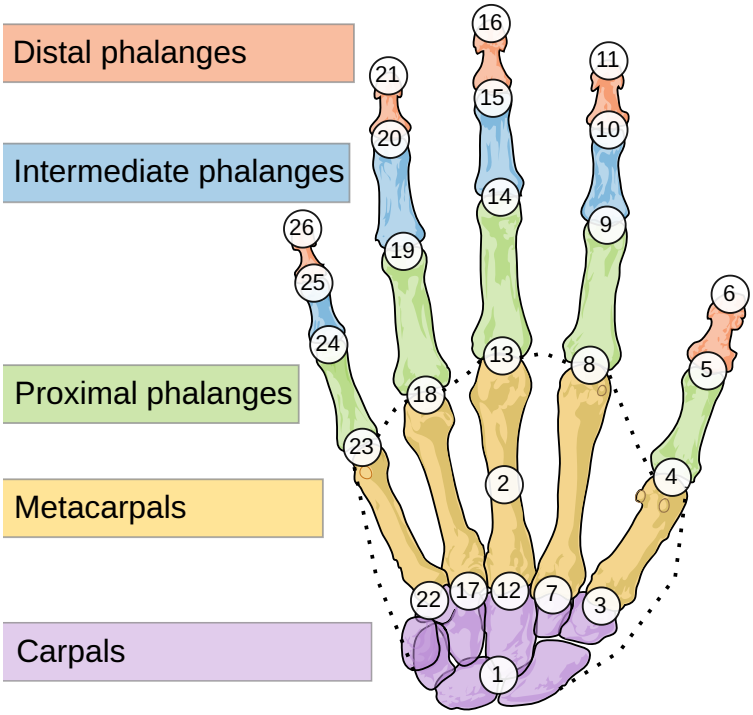


Figure 4.1: Visualization of captured joints in skeletal hand representation. The figure presents a skeletal hand visualization, with each joint numbered according to the corresponding identifiers in Table 4.1.

Joint name	ID	Description
Wrist	1	The wrist.
Palm	2	The palm.
ThumbMetacarpalJoint	3	The lowest joint in the thumb (down in your palm).
ThumbProximalJoint	4	The thumb's second (middle-ish) joint.
ThumbDistalJoint	5	The thumb's first (furthest) joint.
ThumbTip	6	The tip of the thumb.
IndexMetacarpal	7	The lowest joint of the index finger.
IndexKnuckle	8	The knuckle joint of the index finger.
IndexMiddleJoint	9	The middle joint of the index finger.
IndexDistalJoint	10	The joint nearest the tip of the index finger.
IndexTip	11	The tip of the index finger.
MiddleMetacarpal	12	The lowest joint of the middle finger.
MiddleKnuckle	13	The knuckle joint of the middle finger.
MiddleMiddleJoint	14	The middle joint of the middle finger.
MiddleDistalJoint	15	The joint nearest the tip of the middle finger.
MiddleTip	16	The tip of the middle finger.
RingMetacarpal	17	The lowest joint of the ring finger.

Continued on next page

RingKnuckle	18	The knuckle of the ring finger.
RingMiddleJoint	19	The middle joint of the ring finger.
RingDistalJoint	20	The joint nearest the tip of the ring finger.
RingTip	21	The tip of the ring finger.
PinkyMetacarpal	22	The lowest joint of the pinky finger.
PinkyKnuckle	23	The knuckle joint of the pinky finger.
PinkyMiddleJoint	24	The middle joint of the pinky finger.
PinkyDistalJoint	25	The joint nearest the tip of the pinky finger.
PinkyTip	26	The tip of the pinky.

Table 4.1: Description of captured joints. The table provides an overview of the joints captured during the data collection process using the HoloLens 2 device. The identifiers are cross-referenced with Figure 4.1, offering a visual representation of each joint and its location on the hand. Adapted from [39].

4.1.2. 3D hand joint capture process

To ensure an adequate amount of data for reliable training, we set the target of acquiring at least 500 samples per gesture class. With 16 classes in total, this led us to the goal of gathering 8,000 samples for the entire dataset. To achieve this, we recruited 25 different participants to perform the data capture process. Dividing the target of 500 samples per class by the number of participants, we arrived at an allocation of 20 samples per person per gesture class. With 16 gesture classes, each participant contributed 320 samples in total[1].

Prior to data collection, each participant received detailed instructions and a clear explanation of the gestures they were required to perform. Emphasis was placed on maintaining natural and spontaneous hand movements, enabling the dataset to encompass a wide range of realistic gestures.

Regardless of the participants' dominant hand, all users were required to use their right hand for the data capture process. This decision was made to simplify and optimize the data processing pipeline on the HoloLens 2 device, which focused on detecting and capturing only the right hand's movements.

For the data capture interface, a custom-built MR application was developed. The application makes use of the HoloLens 2 sensors to accurately capture the 3D positions of the hand joints. Participants were performing the gestures while the HoloLens 2 recorded the positional data of the hand joints at a sampling rate of 30Hz. The application was designed to minimize distractions during the data capture sessions, ensuring that participants could focus solely on performing the gestures with ease. For complete details of the application please refer to section 4.1.3.

Given that recording all 320 gestures in one continuous session would have been exhausting for the participants, the data collection was organized into four separate sessions, each containing 80 gestures and with an approximate duration of 15 minutes. Within each session, participants performed a total of 5 gestures from each gesture class in a randomized order. This approach of individual sessions not only ensured a more manageable and comfortable data capture experience for the participants, but also allowed us to distribute the effort and maintain consistency throughout the entire dataset.

[1] In fact, we recorded a few more gestures from each user, as a precaution against potential data loss during the cleaning process.

Upon completion of each data capture session, the application generated a data file containing the recorded 3D hand joint positions. Each line of this file represents a specific time frame, with the coordinates of the 26 joints. Each joint is therefore characterized by 3 floats (x, y, z position), so each row has 79 elements ($1 + 26 \times 3$) and it is encoded as:

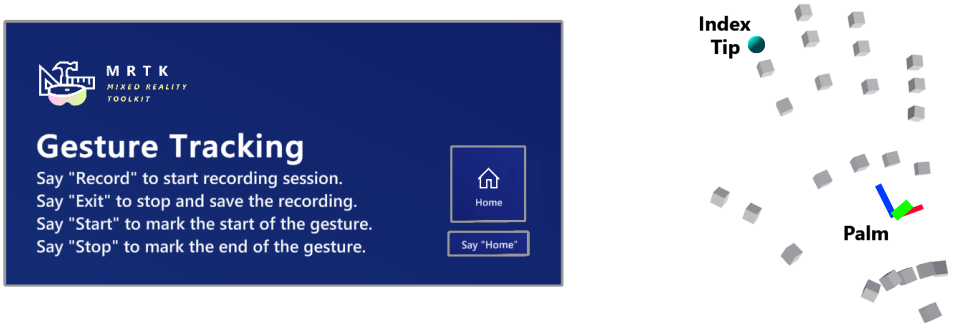
```
Frame ; Joint1_x ; Joint1_y ; Joint1_z ; Joint2_x ; Joint2_y ; Joint2_z ; ...
```

4.1.3. Process of automatic labeling during data capture

Given the substantial number of gestures to be captured, 8000 in total, the implemented system follows a structured workflow to ensure accurate labeling without the need for manual intervention during data acquisition. The two keys of the process are the following:

- **To show the user which gesture to make**: To avoid bias and maintain diversity, the app randomly presents the 80 gestures to be recorded, but always ensuring that those 80 are five of each class.
- **Voice-activated Start and Stop commands**: Since the application informs the user of the next gesture to perform, we can directly label the captured samples with the corresponding gesture identifier. The user's vocal cues of "start" and "stop" enable us to accurately mark the temporal boundaries of each gesture, thus facilitating the automatic labeling of the recorded data.

As shown in Figure 4.2, the labeling process is aided by a couple of auxiliary elements of the app interface. First, a banner that shows the instructions and voice commands for the user to remember. Second, the app superimposes the real-time visualization of hand joints detected by the HoloLens 2 over the user's hand, providing visual feedback and enhancing the user experience.



(a) Voice commands **(b)** Hand joint holograms

Figure 4.2: App interface elements to assist the user. On the left, it can be seen the sign with the voice commands. On the right, the holograms representing each joint.

The app workflow comprises five distinct stages (see Figure 4.3):

(A) Initially, the app remains inactive (Figure 4.3a), awaiting the user to put on the HoloLens 2 device and position the interface elements comfortably (e.g., placing the instructions banner on a wall).

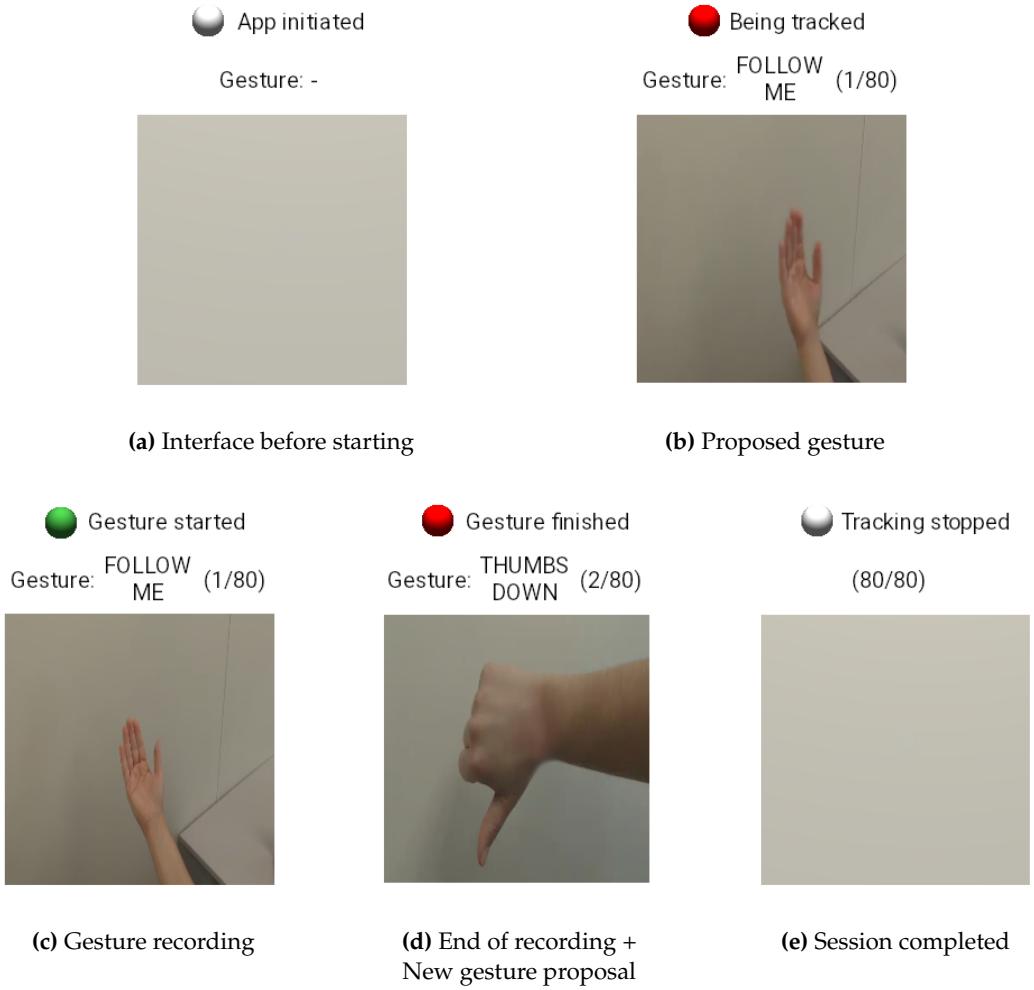
(B) Once the user is ready, he/she says "record" to start the capture session. From this moment on, joints are captured at 30Hz continuously (Figure 4.3b). The application

shows the user which gesture to perform in two ways: by indicating the name of the gesture via text and by showing a panel with a video demonstration of the gesture[2].

(C) Before beginning the gesture, the user says "start" (Figure 4.3c) to mark the gesture's start from that frame onwards.

(D) Similarly, when the user finishes performing the gesture, he/she says "stop" (Figure 4.3d) to mark its end, and the app proceeds to display the next gesture to be performed.

(E) After completing all 80 gestures, the user says "exit" (Figure 4.3e) to finalize the joint capture and store the generated data file in the HoloLens 2 memory.



(a) Interface before starting

(b) Proposed gesture

(c) Gesture recording

(d) End of recording + New gesture proposal

(e) Session completed

Figure 4.3: Workflow of the data capture app: The figure illustrates the sequential steps of the app during the data capture process. Note that the user's actual hand is not represented in this sequence, only the interface visible to the user is shown.

Furthermore, to provide real-time feedback about the app's status during the data capture process, the app incorporates a small sphere that changes color accordingly: it remains gray when not capturing data, it turns red between gestures, and it becomes green while capturing a gesture (between the "start" and "stop" commands).

[2]Important note: It is essential to emphasize that users were explicitly instructed not to imitate the gesture demonstrated in the video. The video merely served as a visual reference to inform participants of the required gesture, while they were encouraged to perform it in a manner that felt natural and comfortable to them.

4.1.4. Ergonomics considerations in data collection

In our data collection process, we placed a strong emphasis on accommodating participants' natural hand movements to achieve the highest level of realism and authenticity in gesture representation. With the objective of capturing diverse and authentic hand gestures, participants were encouraged to perform the gestures in a manner that felt most natural and comfortable to them. This approach allowed for the spontaneous execution of gestures, mirroring how they would naturally perform them in real-life scenarios.

To ensure maximum flexibility and ease of execution, participants were not restricted in their body posture during data capture. The majority of participants chose to perform the gestures while standing, which allowed for full-body engagement and a wide range of hand movements. However, as the data capture sessions required performing a considerable amount of gestures, some participants opted to execute the gestures while seated, offering them more comfort and endurance during the extended recording periods. Additionally, a few participants even chose to support their elbow on an elevated surface while performing certain gestures to reduce fatigue and maintain gesture accuracy.

By providing this flexibility, we ensured that the captured dataset represents a diverse and genuine set of hand movements. These ergonomics considerations were crucial in achieving a dataset that closely mimics real-life scenarios, enhancing the applicability and practicality of the Hand Gesture Recognition models trained on the dataset. Moreover, by prioritizing participants' comfort and preferences during the data capture process, we encouraged a positive environment, resulting in a more engaging and rewarding data collection experience for all participants involved.

4.1.5. Privacy considerations in data collection

Throughout the data capture process, ethical considerations were a priority. Prior to participation, all potential subjects were provided with information about the purpose of the study, the data capture procedures, and the intended use of the recorded data, including any potential data publication. Participants provided their informed consent for data collection and were explicitly informed of their right to withdraw from the study at any point without consequence.

In accordance with established privacy guidelines and regulations, strict measures were implemented to safeguard the confidentiality and anonymity of participants. Personal identifying information was carefully separated from the captured data, ensuring that all individuals remained anonymous throughout the research process.

4.2 Dataset classes

This Hand Gesture Recognition dataset comprises a total of 16 distinct classes, each representing a specific hand gesture. These classes encompass a wide range of gestures, showcasing the versatility and complexity of hand movements. In Figure 4.4, each class is visually represented using a scatter 3D visualization of the captured hand joints. This representation illustrates the precise positioning and spatial arrangement of the hand joints during the execution of the corresponding gesture.

On the other hand, in Table 4.2, each gesture is identified by its unique ID and categorized into one of the four generic types: static (ST), single-dynamic (D), fine-grained dynamic (FG), or dynamic-periodic (P). Moreover, each gesture is accompanied by a brief description, providing insights into its intended meaning or practical applications.

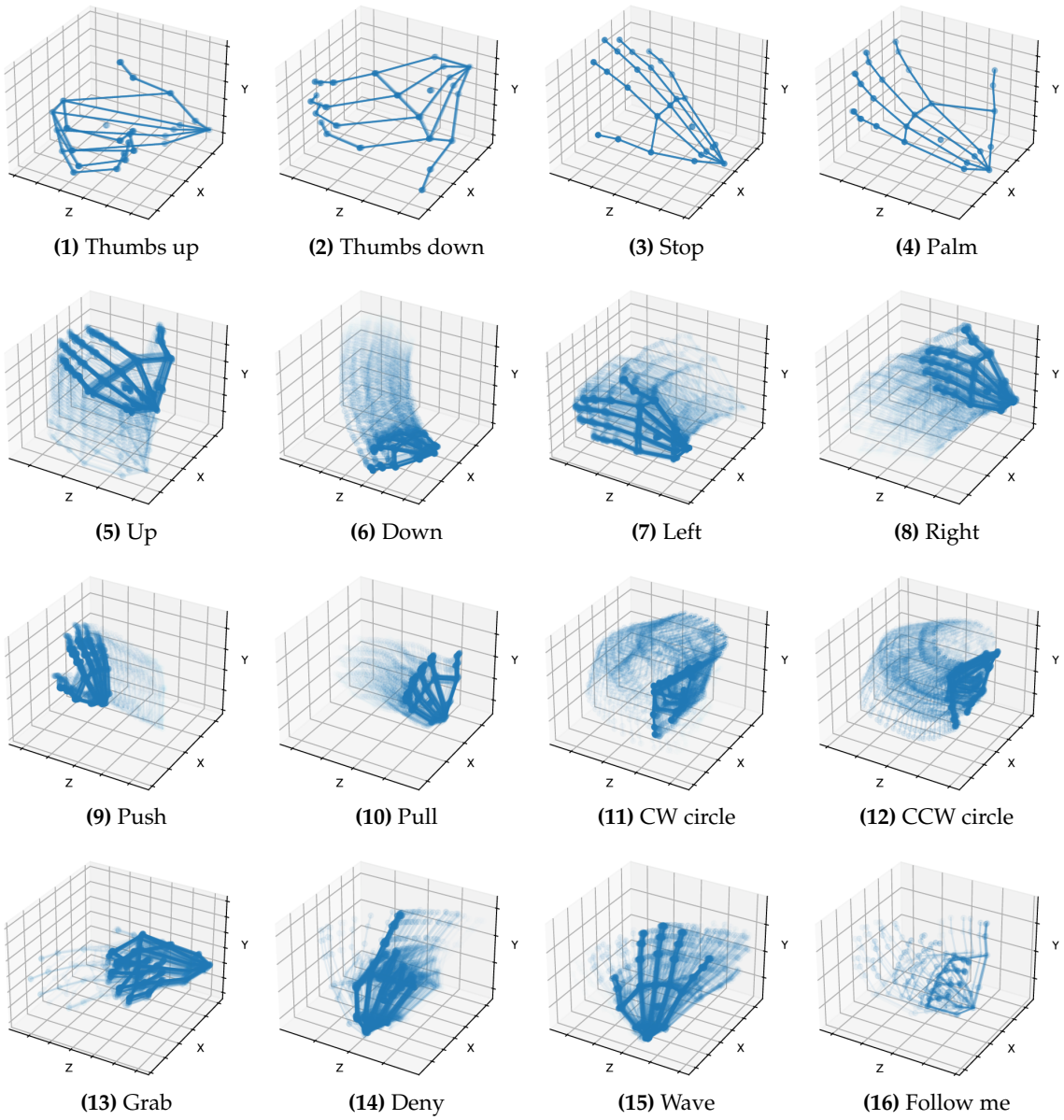


Figure 4.4: Visualization of all gesture classes in the dataset. The figure displays a graphical representation of each class of the dataset, captured using a HoloLens 2 device. The first four gestures belong to static categories, while the remaining gestures are dynamic. In the dynamic ones, the early frames are depicted with lighter shades, gradually transitioning to darker shades for the later frames, representing the temporal flow of the gesture.

ID	Gesture name	Type	Description
01	Thumbs up	ST	Thumb up, rest of the hand closed.
02	Thumbs down	ST	Thumb down, rest of hand closed (inverse of 01).
03	Stop	ST	Palm open, back of hand facing the user.
04	Palm	ST	Palm open, palm facing the user (inverse of 03).

Continued on next page

05	Up	D	(Y-axis) Hand with open palm, moves from the bottom to the top.
06	Down	D	(Y-axis) Hand with open palm, moves from the top to the bottom (inverse of 05).
07	Left	D	(X-axis) Hand with open palm, moves from right to left.
08	Right	D	(X-axis) Hand with open palm, moves from left to right (inverse of 07).
09	Push	D	(Z-axis) Hand with open palm, moves outward, as if pushing something.
10	Pull	D	(Z-axis) Hand with open palm, moves towards the user, as if pulling something (inverse of 09).
11	CW circle	D	(XZ plane) Index finger draw a circle in the air, rest of hand closed. Start at the bottom and clockwise (if it were a clock: $6, 7, 8, ..., 12, 1, 2, ..., 4, 5, 6$).
12	CCW circle	D	(XZ plane) Index finger draws a circle in the air, rest of hand closed. Start at the bottom and counterclockwise (if it were a clock: $6, 5, 4, ..., 1, 12, 11, ..., 8, 7, 6$) (inverse of 11).
13	Grab	FG	Fingers are clamped together.
14	Deny	P	Index finger extended and rest of hand closed. Moving the hand to one side and to the other to indicate denial of something. Repeat several consecutive times.
15	Wave	P	Greet with an open hand. Repeat several consecutive times.
16	Follow me	P	Intuitive gesture to follow. Extend hand forward with palm up and fingers together, then move hand back and forth. Repeat several consecutive times.

Table 4.2: Gesture classes descriptions. The table presents an overview of the 16 distinct gesture classes in the Hand Gesture Recognition dataset presented. The identifiers are cross-referenced with Figure 4.4, offering a visual representation of each class.

4.2.1. Description of gesture categories

This section describes the four generic categories of gestures mentioned above. Each one represents a distinct type of hand movement, enabling a systematic organization of the dataset.

Ensuring the inclusion of at least one gesture from each of the four significant types is desirable to ensure the diversity of the dataset. Each gesture type represents a distinct category of hand movements with unique characteristics, and by incorporating examples of each kind, we enhance the dataset's applicability.

Static gestures contribute to a robust representation of distinct hand postures and symbolic gestures. On the other hand, dynamic gestures capture straight hand motions, making the dataset suitable for recognizing continuous movements and directional instructions. Fine-grained dynamics focus on intricate finger articulations, ensuring the dataset covers gestures with precise finger control. Finally, dynamic-periodic gestures add repetitive patterns to the dataset, which is relevant for recognizing cyclic actions.

By covering these diverse gesture types, we ensure that our dataset captures a broad spectrum of hand movements encountered in real-world scenarios. In the following, we provide a detailed description of each kind, highlighting the specific features that differentiate them.

Static gestures

Static gestures are characterized by a pose that is kept fixed for a minimum duration of 0.5 seconds. Participants are instructed to hold a specific hand configuration without any significant movement. These static gestures often represent specific hand signs or symbols that convey meaning or indicate commands. The specific gestures are those denoted as type ST in Table 4.2.

Single-dynamic gestures

Single-dynamic gestures involve a continuous and smooth trajectory of the hand. Participants are encouraged to execute a single, fluid motion with their hand to represent these gestures. The dynamic nature of these gestures makes them well-suited for conveying directional instructions, such as swiping or tracing gestures. The specific gestures are those denoted as type D in Table 4.2.

Fine-grained dynamic gestures

Fine-grained dynamic gestures focus on the movement of individual fingers during hand movements. Participants perform gestures that involve precise finger movements and configurations. These gestures often emphasize the dexterity and versatility of hand articulations, and they are valuable for applications requiring precise finger tracking and control. The specific gestures are those denoted as type FG in Table 4.2.

Due to their specific nature, these gestures have been less included in the dataset. This category encompasses actions like “grab” and “pinch”, which are constrained by the Microsoft SDK and are associated with particular actions. We have introduced at least one in the dataset to check that it can indeed be recognized and used with our system, but from an MR app development point of view, they are the least attractive gestures as they already have a function in the SDK.

Dynamic-periodic gestures

Finally, dynamic-periodic gestures involve the repetition of the same fingers’ motion pattern at least three times. Participants execute these gestures with a rhythmic and periodic pattern, which makes them suitable for representing actions that have a repetitive nature or need to be performed in cycles, such as waving or saying no with the hand. The specific gestures are those denoted as type P in Table 4.2.

4.3 Analysis of participant demographics

This section presents an analysis of the demographics of the participants involved in the data capture process for the HGR dataset. A diverse group of individuals was recruited for this study, including work colleagues, friends, and family members, who generously volunteered their time and participation without any financial compensation or incentives. Their invaluable contributions enabled the creation of a robust and complete dataset, reflecting a real-world representation of gestures across different demographics.

4.3.1. Participants' age and gender

The gender and age composition of participants in the data capture process plays a significant role in the dataset's diversity and applicability. The dataset comprised 15 male and 10 female participants (there were no individuals who identified with another gender), indicating a relatively balanced gender representation. This inclusion of both genders ensures that the Hand Gesture Recognition models are not biased towards one gender and can generalize well across different users.

In terms of age distribution, the participants were categorized into three main groups. The majority of the participants fell into the young-adult category, aged between 18 to 29 years, with 16 individuals. This age group is particularly relevant as it represents a significant proportion of the potential user base for HGR applications in various domains. The middle-aged group, consisting of 6 participants aged between 30 to 49 years, provides insights into how hand gesture patterns may differ across different life stages. Finally, the dataset also includes 3 older adult participants aged 50 years and above. This age group is essential to capture the nuances in hand movements that may occur with age-related factors, such as reduced dexterity or flexibility.

4.3.2. Participants' dominant hand

All participants in the data capture process were right-handed. Nevertheless, it should be remembered that the capture process was reserved for the use of the right hand, so that even though this may restrict the dataset's diversity in hand dominance, it aligns well with the typical user interaction scenario, where users typically interact predominantly with their dominant hand.

Still, it is important to acknowledge that Hand Gesture Recognition systems should ideally be designed to accommodate both left-handed and right-handed users. However, due to practical constraints and the limited availability of left-handed volunteers, the current dataset predominantly represents gestures performed by right-handed participants. Despite this limitation, the dataset remains a valuable resource for developing and evaluating Hand Gesture Recognition models. Moreover, it also serves as a foundation for future extensions, where efforts can be made to incorporate left-handed gesture data or to simulate left-hand gestures by performing mirror operations, ensuring a more diverse dataset.

4.3.3. Participants' previous experience with MR

Understanding participants' prior experience with Mixed Reality (MR) technologies is essential to measure their familiarity and potential influence on the data capture process. Out of the total participants, 14 individuals had no prior experience with Mixed Reality headsets or exposure to holograms. For these participants, the data capture process served as their first encounter with MR technologies, ensuring that the dataset reflects a diverse range of experience levels.

Additionally, five participants had some prior exposure to MR technologies, having experimented with Mixed Reality headsets on a limited basis before. This group's previous experience may have influenced their interaction with the data capture app and their comfort level with performing gestures in Extended Reality environments.

Furthermore, six participants were already familiar with MR technologies and regularly used the HoloLens 2 in their daily lives. Their proficiency with these technologies may have enabled them to adapt quickly to the data capture process, potentially leading to more refined and accurate hand gestures.

4.4 Dataset cleaning

The dataset cleaning process is a critical step to ensure the data's quality and reliability, ultimately impacting the performance of Hand Gesture Recognition models. In the following sections we will discuss the different measures applied and the approach to the management of outliers.

4.4.1. Applied cleaning processes

During the dataset cleaning process, we implemented specific strategies to address data artifacts and ensure the dataset's integrity. One essential step involved eliminating gestures that users themselves identified as incorrect during the data capture process. Users were given the option to flag any incorrectly performed gestures, which were subsequently removed from the dataset to maintain data accuracy.

Additionally, we performed a thorough search for empty gestures, i.e., those with non-recording of joints between the "start" and "stop" commands. Empty gestures could result from tracking errors or unintentional user actions. As a consequence, we identified one user whose majority of gestures were empty. To rectify this issue and achieve the minimum required 20 gestures per class, we conducted additional data capture sessions with this user.

Moreover, for the rest, a total of eight users (approximately 1/3 of the total participants) recorded some empty gestures, though in a significantly smaller proportion. Specifically, we found a total of 49 empty gestures (less than 1% of the total number of recorded gestures). Among these users, two recorded more than ten empty gestures, while six had only a few (ranging from 1 to 5 empty gestures).

It is important to note that out of the 49 empty gestures, a staggering 45 (92%) belonged to the gesture `Thumbs down`. This finding raises concerns regarding the gesture's recognition accuracy, as it may pose challenges for the HoloLens 2 in capturing the gesture accurately. Further investigation is needed to better understand the reasons behind this observation and ensure reliable recognition for all gestures.

In addition to the aforementioned cleaning processes, we have also applied outlier elimination techniques to further refine the dataset. These outlier removal methods are intended to identify and exclude extreme or erroneous data that could negatively affect the model's training and performance. Further details on the outlier elimination process will be discussed in Section 4.4.2, where we delve into the specific methodologies used to ensure data integrity and reliability.

4.4.2. Identification and handling of outliers

Firstly, in order to rigorously analyze outliers within the dataset, we calculated the quartiles for the duration of the recorded gestures. Specifically, the first quartile (Q1) was found to be 54 frames, the second quartile (Q2 or median) was 76 frames, and the third quartile (Q3) was 95 frames. Additionally, the Interquartile Range (IQR), which represents the spread of the middle 50% of the data, was determined to be 41 frames.

On the one hand, after examining the duration of the gestures, we identified 54 quasi-empty gestures out of 8653 samples. These gestures exhibited an extremely short duration, consisting of fewer than 20 frames, making it highly improbable that any significant hand movement was captured. In addition, it is worth mentioning that most of these

quasi-empty gestures were associated with the same users who had recorded entirely empty gestures, thus reinforcing the theory that they were not valid gestures.

Therefore, we removed these 54 quasi-empty gestures from the dataset. The short duration of these gestures could be attributed to users placing their hands too far away from the HoloLens 2 sensors, resulting in an incomplete gesture detection.

On the other hand, some excessively long gestures have also been detected, which can certainly be considered outliers. By using the IQR Method of Outlier Detection [40], we observed that any gesture with a duration exceeding 157 frames could be considered an outlier. However, we opted to set a slightly higher threshold and remove gestures with a duration greater than 250 frames. This decision was made based on the understanding that some inexperienced users with MR technology might take longer than the majority to perform certain gestures.

In total, 10 gestures were removed from the dataset due to their excessively long durations. Prolonged gestures could potentially result from difficulties in accurately detecting the "stop" voice command, leading to segments of quasi-constant hand position or even a completely relaxed arm, thus compromising the authenticity of the gesture representation.

4.5 Dataset statistics

In this section, we compile and present the statistics of the final version of the dataset, which will be utilized for experimentation in the following chapter. We analyze the duration of the gestures, their distribution across classes, and the partitioning into training, development, and testing sets. Additionally, we provide individual statistics for each partition, offering a complete overview of the dataset's characteristics.

4.5.1. Gesture distribution after cleaning

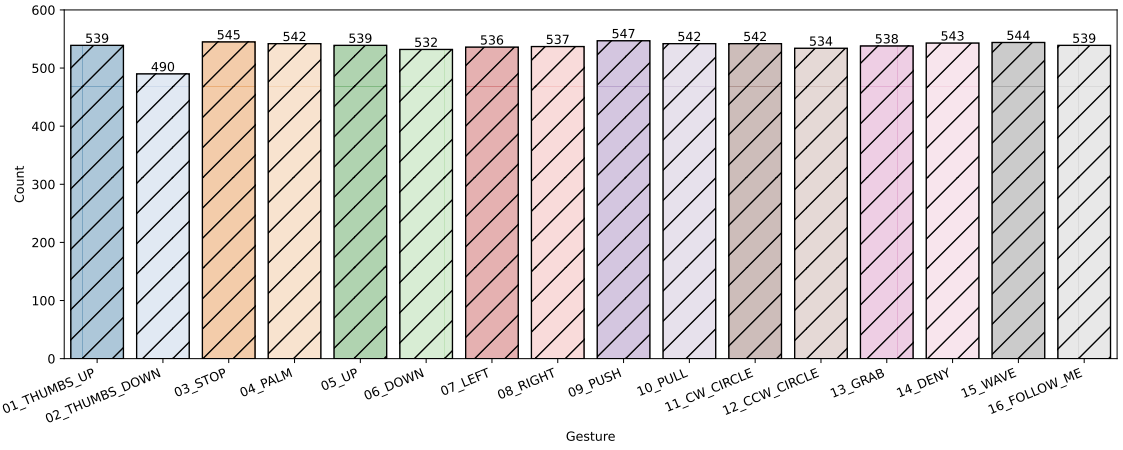


Figure 4.5: Gesture distribution by class in the final dataset after cleaning, showcasing a well-balanced dataset with a minor imbalance in the `Thumbs down` class.

The distribution of gestures after the cleaning process is visualized in Figure 4.5. As expected, all classes contain approximately 500 gestures, reflecting the dataset's balanced nature. However, a slight imbalance is evident, where all classes have over 530 gestures, except for the `Thumbs down` class with 490 gestures.

This imbalance is because most empty or quasi-empty gestures are associated with the `Thumbs down` class (see Section 4.4.1), affecting their representation in the dataset. Nevertheless, the overall impact of this imbalance is minimal and it does not pose a significant concern, as the percentage differences are relatively small.

The distribution analysis further reveals that, even though the potential challenges in capturing the `Thumbs down` gesture accurately, the Mixed Reality device successfully records the majority of these `Thumbs down` gestures, as the imbalance remains marginal.

4.5.2. Gesture duration after cleaning

The analysis of gesture duration is of utmost importance as it plays a crucial role in determining the optimal window size for the developed gesture classifier. As shown in Figure 4.6, the histogram of gesture durations (in frames) exhibits two distinct modes: a primary mode centered around 76 frames and a secondary mode around 60 frames.

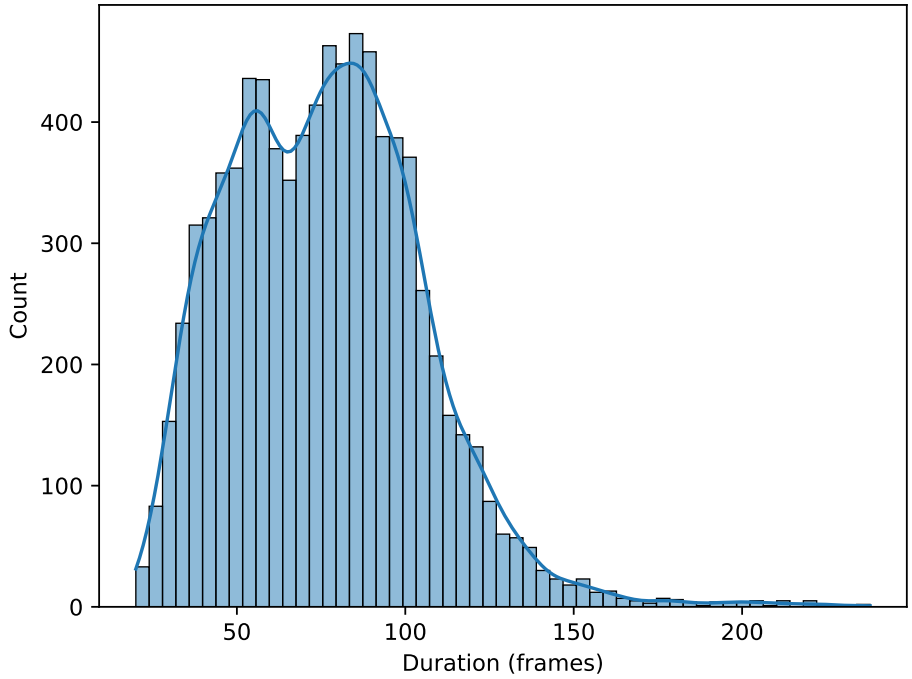


Figure 4.6: Histogram illustrating the distribution of gesture durations in frames.

A deeper examination of the duration by gesture type (Figure 4.7) reveals clear patterns. The first four gestures, which correspond to static hand poses, have a lower average duration, around 60 frames, consistent with the secondary mode observed in the histogram. On the other hand, dynamic gestures have an average duration of approximately 76 frames, aligning with the primary mode in the distribution. The exception to this pattern is seen in the circular gestures (`CW circle` and `CCW circle`), which exhibit a longer average duration of approximately 93 frames.

The increased duration of circular gestures is justified, as they are conceptually and execution-wise more complex compared to other gestures. During data capture, it was noted that circular gestures posed challenges for accurate detection by the HoloLens 2. As a result, users performed these gestures deliberately more slowly to ensure successful capture.

Altogether, the shorter duration of the static ones is compensated by the longer duration of the circular ones, so that the overall mean and median are still around 76 frames.

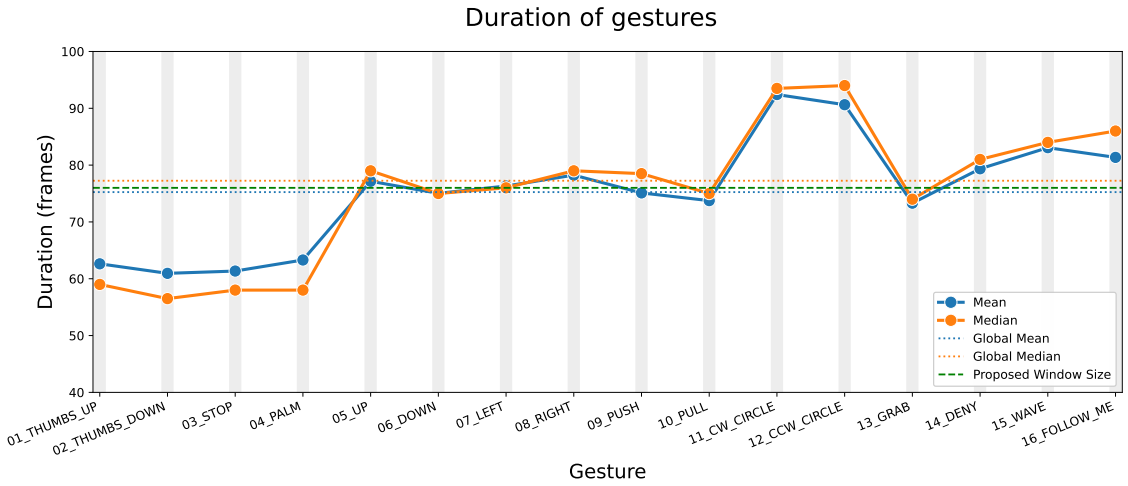


Figure 4.7: Average and median durations of gestures for each gesture class.

4.5.3. Training, development, and test set split

Given a total of 8589 samples captured from 25 unique users, it was crucial to perform user-based partitioning to ensure data independence among the sets and prevent any potential contamination. Accordingly, 17 users were allocated to the training set, 3 users to the development set, and 5 users to the test set.

During the selection process for the test set users, particular attention was given to ensuring diversity in terms of gender, age groups and previous experience with the HoloLens 2. This diverse composition encompassed both users familiar with the technology and those without prior experience, further enhancing the dataset's representativeness for various potential users.

Metric	Training	Development	Test
Number of users	17	3	5
Number of gestures	5887	992	1710
Duration mean	76.228	79.225	75.412
Duration std	29.854	24.070	27.966
Min duration	20	21	23
Duration 25% (Q1)	52	64	56
Duration 50% (Q2)	76	78	75
Duration 75% (Q3)	97	90	92
Max duration	236	222	238

Table 4.3: Statistics of the dataset partition into the training set, development set, and test set. Duration in frames. The partitioning was performed on a per-user basis to ensure data independence and avoid cross-contamination among the sets.

The complete statistics of each dataset partition are provided in Table 4.3. The partitioning was designed to achieve a balanced distribution of gestures among the various classes, thus providing a solid foundation for training and evaluating the Hand Gesture Recognition model.

CHAPTER 5
Experimentation

In this chapter, we present a detailed description of our experimental activities, which aim to develop an efficient and accurate Hand Gesture Recognition system, based on the groundwork established in previous chapters. Through a rigorous experimentation, we examine the trade-offs between model architectures, data transformations, and classification strategies, with the final goal of obtaining an optimized solution suitable for a seamless real-time integration in Mixed Reality applications.

5.1 Proposed approach

As discussed in Section 3.1.2, our gesture recognition strategy is based on the use of three-dimensional positional information of hand joints captured by the HoloLens 2 device. The proposed approach is centered around designing and implementing a robust gesture classification system. Given a temporal window consisting of multiple frames, our primary classifier is designed to determine the specific gesture within the window.

In addition, to lighten the computational load on this primary classifier and ensure real-time processing, we propose the introduction of a preliminary binary classifier. The purpose of this binary classifier, also referred to as detector, is to rapidly determine the presence or absence of a gesture within a temporal window. As a result, we enable an efficient filtering mechanism that prioritizes more complex analysis only when a gesture is detected, enhancing the overall responsiveness of the system.

Taking into account the findings of the reviewed state of the art literature (see Section 3.3), we have concluded that the utilization of one-dimensional Convolutional Neural Networks is promising for our gesture recognition approach. The efficiency inherent to CNN-1D architectures aligns well with our goals, making them a suitable choice for our experimentation.

Among the one-dimensional network options within the gesture recognition context, we note the success of the STRONGER network [27], particularly due to its initial transformations applied to the joint data. These transformations appear to enhance the discriminative capabilities of the classifier, even though being a simple (and efficient) model. Our research will delve into these transformative techniques, with a focus on their adaptation and integration into various network architectures.

5.1.1. Experimental setup and methodology

In this section, we describe the systematic experimental setup and methodology employed to conduct our investigation.

Data preprocessing

In order to maintain uniformity within each training batch, it is necessary that all temporal windows have the same dimensions. To address this, we make the simplifying assumption that all windows contain a fixed number, K, of frames. After the dataset cleaning process, we conducted the following preprocessing step: for a given gesture window, we resized it to precisely contain K frames. If the window had fewer frames, we padded it with frames from the borders. Conversely, if the window exceeded K frames, we retained only the central K frames, as these frames are more likely to capture the essence of the gesture due to the inherent noise associated with gesture labeling.

Temporal window definition

The choice of an optimal value for K, the number of frames within each temporal window, is a critical factor. After careful consideration, we propose $K = 76$ as the ideal value. This choice is motivated by our observations in Section 4.5.2, where the median and mean duration of gestures converged around this size (see Figure 4.7).

Training configuration

During the training process, several techniques were employed to optimize model performance. The training configuration included a batch size of 32 and ran for 100 epochs. Adaptive learning rate adjustment was implemented through a *reduce on plateau* scheduler, with a factor of 0.2 and a patience of 5 epochs, ensuring efficient convergence. Additionally, a custom learning rate schedule was defined, gradually decreasing the learning rate from 0.001 to 0.00001 as the epochs progressed. The training process was further refined by incorporating an early stopping method with a patience of 5 epochs to prevent overfitting. Finally, Adam optimizer was used with an initial learning rate of 0.001.

Evaluation metrics

The performance evaluation of our proposed approach relies on accuracy (the ratio of correct predictions to the total predictions) as the primary metric. For the main gesture classifier, we assess the accuracy of correctly identifying the specific gesture in each window. For the detector (referred to as the binary classifier), we evaluate the accuracy of detecting the presence or absence of a gesture in each window.

On the other hand, in the context of the complete system, we focus on the GER metric (see Section 5.8.2) to determine the overall quality of the system, in which all types of errors have the same importance. It is worth highlighting that in both the complete system and the binary classifier, each window may or may not contain a gesture. However, if a gesture is present, the window contains one, and only one, gesture.

Hardware and software configuration

Our experiments were conducted on two distinct hardware configurations. When referring to CPU-based computations, the configuration encompassed a machine equipped with a 13th Gen Intel i7-1360P (16 cores) @ 5.000 GHz, 31726 MiB of RAM, Ubuntu 22.04.1, and Python 3.10.12. For GPU-based computations, we utilized a setup featuring a 7th Gen Intel i7-7700 (8 cores) @ 4.200 GHz, an NVIDIA GeForce RTX 3070 with 8192 MiB of VRAM, 15926 MiB of RAM, Ubuntu 20.04.6, and Python 3.8.10. If the machine is not explicitly mentioned, the one with GPU was used.

Results and experimental replication

The reported experimental results are based on an average of ten independent training runs, ensuring the reliability and consistency of our proposed approach. This strategy not only mitigates the impact of random variations but also enhances the robustness of our findings. In addition to presenting the mean values, we include the corresponding 95% Confidence Intervals (CIs) to provide an understanding of result stability and variability.

5.2 Custom transformations for Hand Gesture Recognition

In this section, we delve into the motivation and implementation of the custom transformations for enhancing Hand Gesture Recognition. An effective joint-sequence representation should encompass both global motion information and location-viewpoint invariance. However, it is challenging to satisfy both requirements in one feature [31]. Leveraging insights from the DDNet and STRONGER approaches [27, 31, 7], we adopt the strategy of integrating distinct transformations at the top of the model. With this, we seek to combine location-viewpoint invariant features, global orientation features, and a two-scale global motion feature. In the following, we present each of the transformations applied on the raw 3D joint positions.

5.2.1. Joint Collection of Distances (JCD)

The first transformation, the Joint Collection of Distances, focuses on capturing the Euclidean distances between hand joints. For each frame, the JCD transformation calculates the L2 distance matrix among every pair of hand joints, resulting in a symmetric matrix with a full size of $26 \times 26 = 676$. However, exploiting its symmetrical nature and the fact that the diagonal distances are always zero (reflecting the distance of a joint from itself), we can simplify this matrix to its lower triangular form, which can be flattened into a single vector of size:

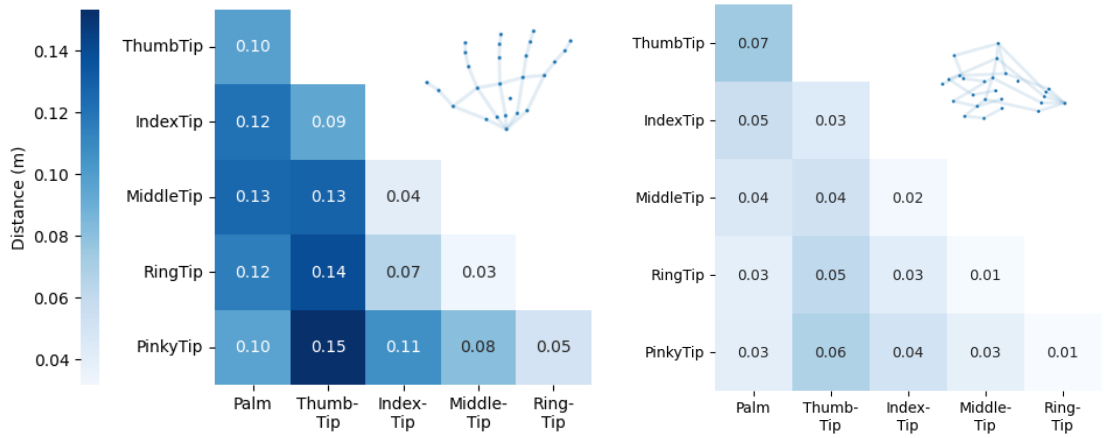
$$\binom{26}{2} = \frac{26!}{2! \cdot (26-2)!} = \frac{26!}{2! \cdot 24!} = \frac{25 \cdot 26}{2} = 325$$

Thus, from the original set of 26 three-dimensional joint points (which flattened form a vector in $\mathbb{R}^{78}$), the JCD transformation yields a vector in $\mathbb{R}^{325}$. This representation offers a remarkable advantage: it is inherently location-viewpoint invariant. It eliminates the need for translations or rotations to position the hand in a canonical orientation.

Formally, let K be the total number of frames in a window, and let N be the total number of joints (in our case $K = 76$ and $N = 26$). At frame $k \in [1, K]$, the 3D Cartesian coordinates of joint $n \in [1, N]$ is represented as $J_n^k = (x, y, z)$. The formula for calculating the JCD feature of a frame k is:

$$JCD^k = \begin{pmatrix} \left\| \overrightarrow{J_2^k J_1^k} \right\|_2 & & & \\ \vdots & \ddots & & \\ \vdots & \cdots & \ddots & \\ \left\| \overrightarrow{J_N^k J_1^k} \right\|_2 & \cdots & \cdots & \left\| \overrightarrow{J_N^k J_{N-1}^k} \right\|_2 \end{pmatrix}$$

where $\left\| \overrightarrow{J_j^k J_i^k} \right\|_2$ denotes the Euclidean distance between J_i^k and J_j^k, with $1 \leq i < j \leq N$.



(a) `Stop` gesture **(b)** `Grab` gesture

Figure 5.1: Illustration of a reduced JCD transformation for distinct hand poses.

As an illustrative example, consider Figure 5.1, which provides a simplified representation of the Joint Collection of Distances transformation for two hand poses. To facilitate visualization, we have focused on a subset of joint distances by selecting six key joints: the palm and the tips of each finger. The left portion of the figure (Figure 5.1a) showcases the JCD representation for an open hand gesture, corresponding to the `Stop` gesture. Here, it is evident that the distance between the thumb and the pinky is notably large (15 cm).

In contrast, the right side of the figure (Figure 5.1b) displays the JCD for a closed hand, symbolizing the ending of a `Grab` gesture. In this case, we observe that the distances between joints are generally much smaller (6 cm between the thumb and the pinky tips), reflecting the proximity of the fingers to one another. This visual insight into the JCD transformation highlights how it captures the relative spatial relationships among hand joints, facilitating the discrimination among gestures.

However, while JCD provides viewpoint invariance, it lacks the ability to capture global hand motion trajectories or orientations, making it insufficient for recognizing dynamic or symmetric gestures on its own (e.g., `Thumbs up` and `Thumbs down` would generate the same JCD). This underscores the necessity of the additional transformations of the network input.

5.2.2. Joint Pairs' Directions (JPD)

The second transformation, referred to as Joint Pairs' Directions, aims to derive a set of representative directions from specific pairs of hand joints. These directions are calculated as the vector difference between the positions of the involved joints. Specifically, the following seven directions are considered:

- Direction from the palm to the thumb tip.
- Direction from the palm to the index finger tip.
- Direction from the palm to the middle finger tip.
- Direction from the index finger tip to the thumb tip.

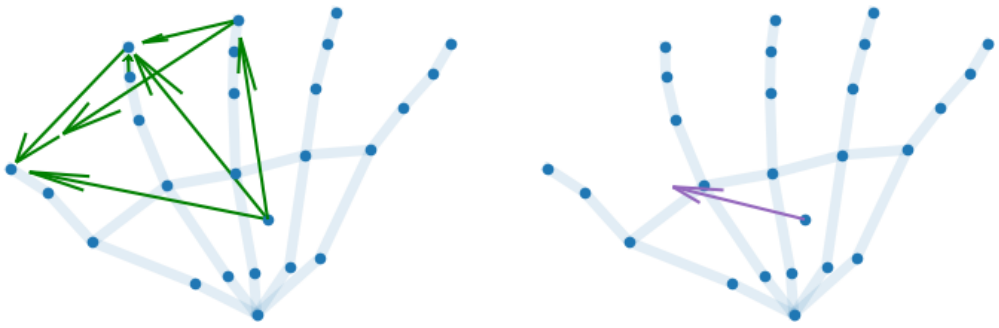
Continued on next page

- Direction from the middle finger tip to the thumb tip.
- Direction from the middle finger tip to the index finger tip.
- Direction from the distal joint of the index finger to the index finger tip. (It represents where the index finger actually points to)

Figure 5.2a illustrates this collection of directions within an open hand. These vectors capture essential geometric relationships among different parts of the hand and represent global orientation features. The incorporation of these directions into the hand representation enhances the information provided to the Neural Network and contributes to its gesture discrimination capabilities.

5.2.3. Palm Orientation (PO)

The third transformation is simply the orientation of the palm, i.e., the normal that leaves the palm of the hand. It is calculated as the unit normal vector to the plane defined by the index knuckle, the pinky knuckle, and the palm. This transformation serves as an additional global orientation feature, capturing the hand basic alignment. Figure 5.2b shows the representation of this vector in an open hand.



(a) JPD transformation **(b)** PO transformation

Figure 5.2: Illustration of the Joint Pairs' Directions and Palm Orientation transformations in an open hand (`Stop`) gesture.

5.2.4. Two-scale motion features: Mslow and Mfast

The final transformations, Mslow and Mfast, cover joint velocities computed at two distinct scales: slow and fast. These transformations capture temporal differences (i.e., velocity) between consecutive joint positions to derive global motions that are location-invariant. As global motions for the same action may exhibit varying scales, some can be faster and others slower, we need a robust global motion feature accounting for both fast and slow movements.

To achieve this, we compute both fast and slow global motions. This approach draws inspiration from the concept of two-scale optical flows proposed for RGB-based action recognition [41]. SlowFast networks have demonstrated remarkable efficacy in classifying videos within vast and challenging datasets such as Kinetics-400 and Kinetics-600 [41]. Their successful application extends to other video recognition tasks, including action detection and object localization within RGB videos [41].

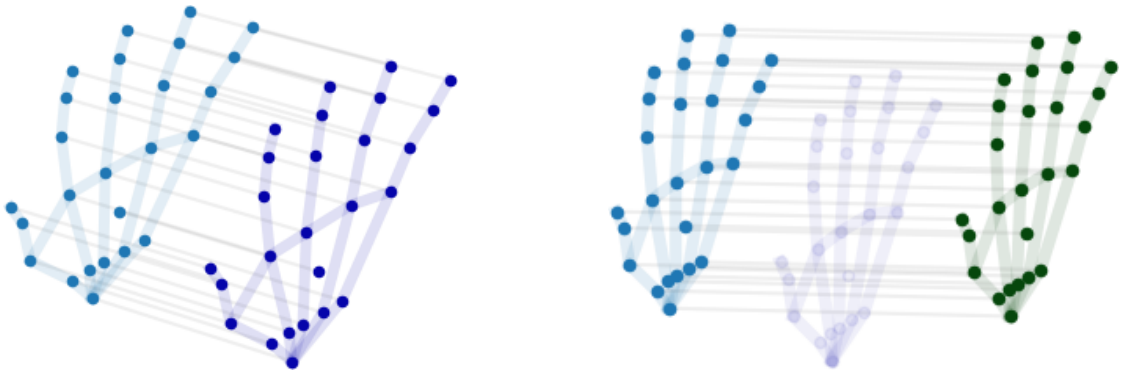
Formally, given a window of K frames, and $S^k = \{J_1^k, J_2^k, \ldots, J_N^k\}$ being the set of joint positions in frame k, the motions are computed as:

$$\begin{aligned} M_{slow}^k &= S^{k+1} - S^k\,, \quad k \in \{1,2,3,\ldots,K-1\} \\ M_{fast}^k &= S^{k+2} - S^k\,, \quad k \in \{1,3,5,\ldots,K-2\} \end{aligned}$$

$$\begin{aligned} Mslow &= \left\{ ms \;\middle|\; \forall\, k \in \{1,2,3,\ldots,K-1\}\,,\; ms = M_{slow}^k \right\} \\ Mfast &= \left\{ mf \;\middle|\; \forall\, k \in \{1,3,5,\ldots,K-2\}\,,\; mf = M_{fast}^k \right\} \end{aligned}$$

Notice that the final motion calculation is done per window and not per frame.

An intuitive explanation of these flows can be visualized with Figure 5.3. Mslow computes the hand's movement between consecutive frames, essentially measuring the distance covered by subtracting the positions of the same joint in both frames. Conversely, Mfast carries out a similar operation but with a stride of 2, computing the displacement between frame i and frame $i+2$. In the illustration, the initial frame is depicted in blue, the subsequent frame in purple, and the frame two steps ahead in green. This approach captures different aspects of hand motion patterns, capturing both slower and faster features of movement for a more complete representation.



(a) Mslow **(b)** Mfast

Figure 5.3: Illustration of Mslow and Mfast flow fundamental calculation.

5.2.5. Integration of transformations within the model architecture

Once we have defined the five input transformations, we can proceed to examine their incorporation and utilization within the architecture of the Deep Learning model. As previously mentioned, our approach involves employing CNN-1D, where a window of K frames serves as the input, and the model's objective is to predict the corresponding gesture class based on this input sequence (or detect if there is a gesture in the binary case).

In essence, upon receiving the input, the network initiates five distinct pathways, applying a specific transformation to each through a lambda layer. Subsequently, the network performs a series of convolutions with residual connections (for specific details,

please refer to Appendix A) to obtain a form of sub-embedding for the gesture. These sub-embeddings are then combined in a concatenation layer, resulting in a full gesture embedding.

Thus, during training, the network will receive tensors of size: $batch\_size \times K \times A$, where K is the number of frames contained in a window and A is the number of components in each frame. The size of these variables in each phase of the block of transformations can be seen in Table 5.1.

Stage	K	A	Comments
Input	76	78	26 joints $\times$ 3 spatial components = 78
JCD	76	325	$\binom{26}{2} = \frac{26!}{2!\cdot(26-2)!} = 325$
JPD	76	21	7 directions $\times$ 3 spatial components = 21
PO	76	3	1 normal $\times$ 3 spatial components = 3
Mslow	75	78	The window becomes $K-1$
Mfast	37	78	The window becomes $K/2-1$
Output of Transf. block (concatenate)	340	Last conv. filter size (128)	$K_{concat} = 76+76+76+75+37 = 340$ Since all paths have passed through convolutions before concatenation, the size of A is the filter size (assuming uniformity across all paths for concat.)

Table 5.1: Output tensor sizes after each stage of the transformations block.

Once concatenated, the network proceeds as if the input was the produced embedding rather than the raw joint data. The concept of this transformation block at the top of the network is illustrated in Figure 5.4.

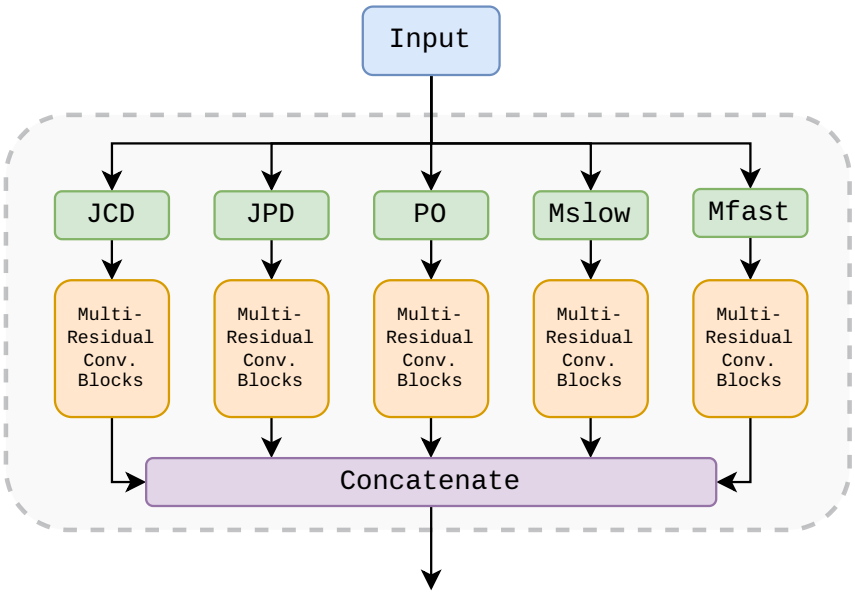


Figure 5.4: Illustration of the initial transformations block in the model architecture, showcasing the application of the five custom transformations.

5.3 Translational layer: normalizing hand positions

In this section, we explore the Translational layer whose function is to normalize the coordinates of the hand joints captured by the HoloLens 2 device.

5.3.1. Coordinate system and stationary frame of reference

In 3D graphics applications, including everything related to HoloLens 2, Cartesian coordinate systems are employed to define the positions and orientations of virtual objects [42]. These systems establish three perpendicular axes: X, Y, and Z. Each object in a scene, or in our case, each hand joint, is assigned an XYZ position in its respective coordinate system. This system of reference is expressed in meters and forms the basis for rendering holographic content.

Holographic rendering dynamically adjusts the app's presentation of holograms as users move, ensuring that virtual objects align with their predicted head movements. For seated-scale experiences within a game engine like Unity (which is the one we use for the hand-capture apps), a stationary frame of reference defines the engine's "world origin". Objects situated at specific world coordinates are positioned using this frame of reference, allowing them to remain stable in the user's view, even as the user moves [42].

In our context, the coordinates of hand joints are represented in this stationary frame of reference, with the origin set at the user's initial head position and orientation. In other words, the $(0, 0, 0)$ will be the position of the user's head at the moment he/she opened the MR application.

5.3.2. Addressing coordinate shifts

Given the nature of Mixed Reality, users are not stationary but rather move within the physical environment. As a result, coordinates received by the deployed application can differ significantly from those captured during dataset collection and used for training the Neural Network. To address this challenge, we introduce the Translational layer, which implements a normalization process for hand positions.

The Translational layer performs a translation normalization, achieved through joint displacement within a window of frames. For this purpose, we calculate the mean position along each axis using a moving average that varies with each window. With this value, we translate all the hand's coordinates to the coordinate system's origin.

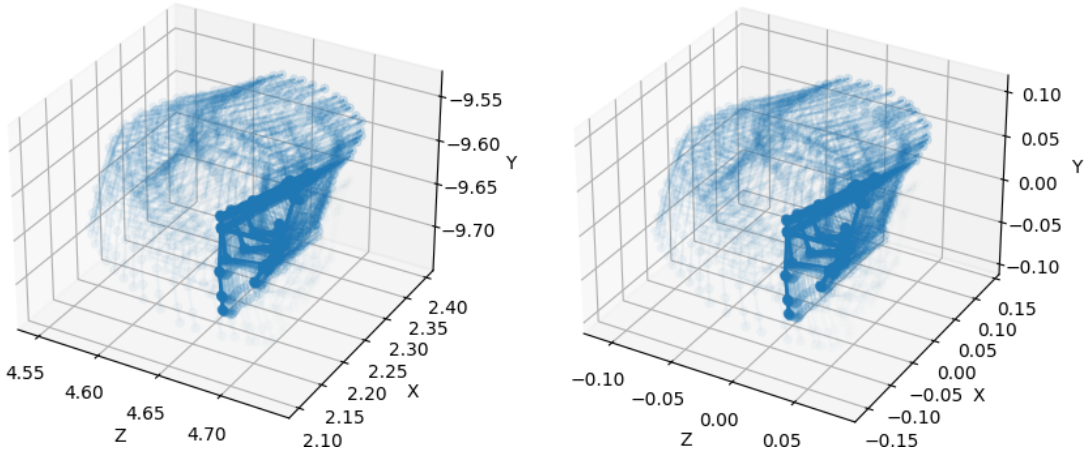
Formally, let A be the total number of components of a frame, that is, $N \times 3$ (number of joints times coordinates per joint). Given a window of size $(K \times A)$, we reorder the tensor to a size of $(\frac{K \cdot A}{3} \times 3)$ and we calculate the global mean per axis: μ_x , μ_y , μ_z. Once the means have been calculated, we construct the translation matrix T of size $(K \times A)$:

$$T = \begin{pmatrix} \mu_x & \mu_y & \mu_z & \mu_x & \mu_y & \mu_z & \cdots & \mu_x & \mu_y & \mu_z \\ \mu_x & \mu_y & \mu_z & \mu_x & \mu_y & \mu_z & \cdots & \mu_x & \mu_y & \mu_z \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mu_x & \mu_y & \mu_z & \mu_x & \mu_y & \mu_z & \cdots & \mu_x & \mu_y & \mu_z \end{pmatrix}$$

And finally, all the points of the window W are translated by subtracting these averages: $W' = W - T$.

Our ultimate goal is to enable direct input of captured HoloLens 2 windows into the Neural Network. To achieve this, we have integrated the normalization process directly into the network's architecture. A lambda layer, the Translational layer, is positioned at the network's top, even before the application of the transformations block. This layer is responsible for the initial translation of positions to the origin before further calculations are performed, ensuring consistency between the input data and the training context.

It is crucial to emphasize that translation is applied per window, rather than per frame. This approach preserves the concept of motion, allowing the hand to maintain its dynamic properties. An example of window translation is illustrated in Figure 5.5.



(a) Original hand motion window

(b) Translated hand motion window

Figure 5.5: Illustration of the Translational layer's normalization process on a `CW circle` hand gesture window.

5.4 Data Augmentation techniques

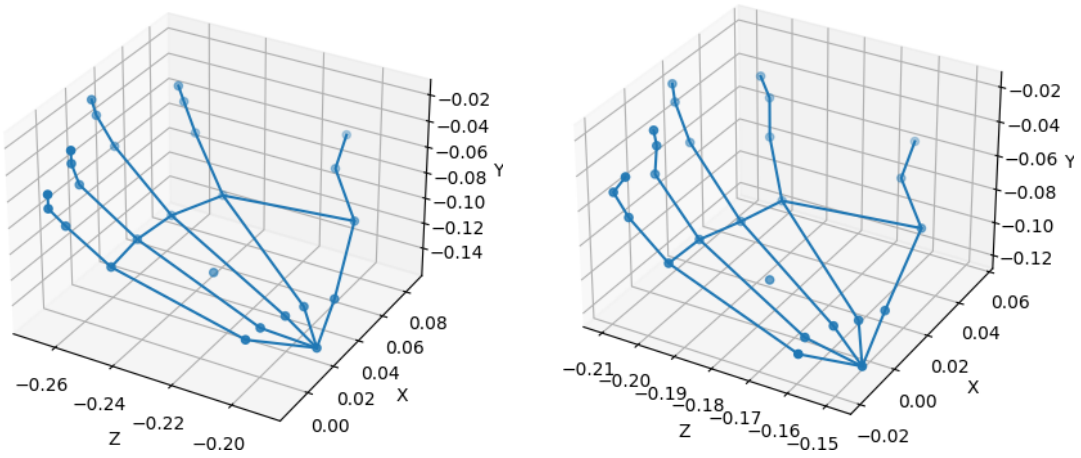
In order to introduce additional variability to the dataset and enhance the model's ability to generalize, we have implemented a simple Data Augmentation strategy. This augmentation process involves applying perturbations to the joint positions, allowing the network to encounter a broader range of hand configurations during training. Specifically, the following augmentation techniques have been incorporated into the data preprocessing pipeline:

- **Gaussian Noise (GN)**: We add Gaussian Noise with a mean of zero and a standard deviation of 0.001 to the joint positions. This noise is applied independently to each frame in a window with a 50% probability. As a result, the hand positions within the window suffer subtle fluctuations.
- **Rotations**: Rotations are applied around any of the axes (chosen randomly) within the range of [-20 degrees, +20 degrees]. These rotations are applied uniformly to all frames within a window, introducing minor variations in hand orientation.
- **Scaling**: Uniform scaling is performed along all three axes simultaneously, within the range of [0.75, 1.25]. This ensures that the hand's overall size can vary slightly while maintaining proportionality.

Notably, no translations are applied during this augmentation process, as hand positions are normalized within each window using the Translational layer. Additionally, the mean of the introduced noise is set to zero to avoid altering the overall hand position.

To implement this augmentation strategy, we utilized the `ImageDataGenerator` provided by the Keras library. While initially designed for image data, we adapted it for our purpose by introducing a dummy dimension to the data. By working with tensors of size $(batch\_size \times K \times A \times 1)$, we effectively utilize the `preprocessing_function` argument of the generator. Within this custom preprocessing function, transformations were defined manually using NumPy. Subsequently, the network's input layer handles the reshaping and removal of the extra dummy dimension, ensuring compatibility with the rest of the architecture.

By incorporating these augmentation techniques, we seek to enrich the dataset with diverse hand configurations and motions, enabling the model to become more robust and generalize effectively. An illustrative example of this Data Augmentation is presented in Figure 5.6 applied to a `Palm` hand gesture frame.



(a) Original hand

(b) Augmented hand

Figure 5.6: Data Augmentation example with Gaussian Noise, scaling ($\times 0.78$), and rotation around the Z-axis ($+20°$).

5.5 Hand gesture classification models

5.5.1. Model comparisons and implementations

As we have already discussed in Chapter 3, since Yang et al. [31] observed that a very simple network architecture based on 1D convolutions fed with features derived from the hand joint sequence can provide state-of-the-art results with reduced computational complexity, they are, together with MLPs for baseline, the option where we will mainly focus. In the following, we present the different models studied.

Baseline model: MultiLayer Perceptron (MLP)

The MultiLayer Perceptron serves as our baseline model for hand gesture classification. In this approach, the entire gesture sequence within a window is flattened into a vector of size $(K \cdot A)$. The architecture of the MLP consists of four Fully Connected (FC) layers. The ultimate layer is a softmax classifier, totally connected to the previous layer's output, featuring a number of neurons equivalent to the dataset's class count. Within the hidden layers, each one is composed of 500 neurons, with ReLU as the activation function, and with dropout layers incorporated before the dense layers.

While the MLP offers simplicity and ease of implementation, its design inherently limits its ability to capture intricate temporal patterns in the gesture sequences. In subsequent sections, we will explore more advanced models that aim to address this limitation and improve classification accuracy.

Residual Network 1D (ResNet-1D)

The Residual Network 1D architecture is structured similarly to a traditional ResNet [8] but employs 1D convolutions. Comprising a total of 11 layers, the network consists of three distinct residual blocks, culminating in a Global Average Pooling (GAP) layer and a final softmax classifier. The number of neurons in this classifier is equal to the number of classes in the dataset.

Each residual block starts with three consecutive convolutions, the output of which is added with the residual block's initial input. This summation is then directed into the subsequent layer. All convolutions utilize a fixed number of filters: 64 for the first block and 128 for the second and third blocks. The Rectified Linear Unit (ReLU) activation function is employed, preceded by a Batch Normalization operation. Within each residual block, the filter length is set at 8, 5, and 3 for the first, second, and third convolution, respectively. This configuration of ResNet-1D allows for the capture of intricate temporal patterns [15] which should help in the classification of dynamic gestures.

STRONGER network

For the STRONGER architecture, we implemented the model based on the description provided in the paper [27], as we did not find an available implementation. It begins with the application of the transformation block (for simplicity, we have adopted our block which is similar to the one they propose). After the concatenation of transformed features, the model proceeds with four additional convolutional layers. Subsequently, a Global Average Pooling (GAP) layer is utilized to flatten the resulting tensor. The flattened representation is then passed through a Fully Connected (FC) layer comprising 128 neurons, and finally, through another dense layer with as many neurons as the number of classes together with a softmax activation function.

Although it is not exactly the original network of the paper (we have also placed the Translational layer on top of it), it is very similar. Since it has already shown a favorable performance in gesture recognition tasks, we expect it to also hold for our dataset.

Enhanced models: MLP and ResNet-1D with transformations block

With these models, we explore the impact of integrating the proposed transformations into the two architectures already seen. The core concept behind these enhanced models is introducing the transformation block, encompassing the five distinct paths at the top of these Neural Networks. By doing so, we aim to assess whether including this block positively influences the performance, incorporating domain-specific transformations as an initial processing step.

ParaRed: Parameter-Reduced model

With the objective of achieving real-time gesture recognition, we also developed the ParaRed network, a parameter-reduced version of our best-performing model (see Table 5.2). By lightening the architecture through the removal and simplification of some convolution blocks, we sought to strike a balance between computational efficiency and gesture classification accuracy.

5.5.2. Results analysis of gesture classification models

Once the architectures have been described, we proceed to evaluate their performance. We have analyzed the models with and without Data Augmentation. This analysis aims to identify the best models, as well as to evaluate the effectiveness and added value of the transformations.

Model	Accuracy (± CI)
MLP	0.896 ± 0.019
DA + MLP	0.915 ± 0.017
ResNet-1D	0.968 ± 0.011
DA + ResNet-1D	0.969 ± 0.011
STRONGER	0.970 ± 0.011
DA + STRONGER	0.977 ± 0.009
Transf + MLP	0.878 ± 0.020
DA + Transf + MLP	0.895 ± 0.019
Transf + ResNet-1D	0.984 ± 0.008
DA + Transf + ResNet-1D	**0.985 ± 0.008**
DA + ParaRed	0.981 ± 0.008

Table 5.2: Classification results of the different models on our dataset. It is shown the mean accuracy and the 95% Confidence Intervals (CI) of ten independent training runs.

In Table 5.2, we can see the results of the accuracy of each classifier. The baseline MultiLayer Perceptron already achieves remarkable accuracy, reaching a precision of 0.896. Moving on, the introduction of ResNet-1D demonstrates excellent progress in dealing with temporal sequences, resulting in a substantial accuracy boost up to 0.968. This result reaffirms the suitability of residual architectures for temporal data processing, aligning with their proven success in similar 1D contexts [15].

We expected good results from the STRONGER, and so it has been, obtaining an accuracy of 0.977 in the Data Augmentation (DA) variant. This result reinforces the idea that incorporating complex transformations at the top of the network holds promise for improving gesture recognition.

Turning our attention to the models integrated with the transformations mentioned above, some divergences in performance emerge. The MLP has lost accuracy, probably due to the inherent flattening operation, which, together with the transformations, seems to aggravate the loss of positional information. As a result, its accuracy decreases to 0.878, even behind the plain MLP. In contrast, the ResNet-1D positively integrates the transformations. It achieves the highest accuracy of 0.985 in the DA configuration, showcasing the compatibility of these transformations with residual architectures and their efficacy in capturing intricate temporal relationships.

Furthermore, our proposed ParaRed, a parameter-reduced version with five times fewer parameters (see Section 5.7.1) than the best ResNet (DA + Transf + ResNet-1D), emerges as a strong candidate. Despite the limitations imposed by this parameter reduction, it maintains a competitive accuracy of 0.981, reaffirming the robustness of its design and its potential as an efficient real-time gesture recognition model.

In addition, the visualization of the confusion matrices of these models can be found in Appendix B, which provides an overview of the classification model results on specific gestures. Although the obtained accuracy results already indicate generally good performance, it is interesting to highlight specific areas of confusion. In particular, the MLP model shows confusion between `Wave` and `Stop` gestures, as well as between `Follow me` and `Palm`. Also, we can see how the Transf + MLP model fails mainly because of significant confusion between the right and left gestures (both those of a straight gesture and circles) and the DA version directly classifies all `CW circle` as `CCW circle`.

Finally, it should be noted that Data Augmentation is advantageous in all our models. Although its impact may vary depending on the model, it is undeniable that it contributes to improving the overall performance, which further underlines its importance in enhancing model generalization.

5.6 Binary gesture detection

5.6.1. Motivation

The motivation for employing binary gesture detection includes several reasons. First and foremost, the adoption of a detection approach aligns with established practices in the domain, particularly usual in the realm of RGB video-based gesture recognition [29, 30]. This approach is based on the concern to improve computational efficiency and real-time processing, a common goal shared across HGR literature.

Furthermore, our dataset does not have a specific `no-gesture` class. In situations where a gesture is not present, a method is required to detect such instances effectively. While one potential solution involves employing softmax probabilities as a means of decision-making, it is essential to acknowledge the inherent limitations of this approach. Neural Networks often exhibit a tendency toward overconfidence in their predictions, yielding predicted probabilities that may not accurately reflect true confidence levels [43].

Consequently, the implementation of more advanced calibration techniques [43, 44] to address this disparity between predicted and true probabilities becomes a viable consideration. However, the binary gesture detection approach offers us an intuitive and interpretable framework that avoids the complexities associated with probabilistic uncertainty.

5.6.2. Data capture for no-gesture

Defining and capturing samples of non-gesture instances was conceptually more challenging because of the actual definition of the no-gesture class itself. Unlike the well-defined categories of gestures, no-gestures lack a clear and predefined form, making their capture a complex task. To meet this challenge, we adopted an imaginative approach which uses the same application employed for gesture capture (see Section 4.1.3).

Specifically, we asked users to perform simple, non-gestural movements while the application captured their hand joints. Emphasis was placed on ensuring that no recognizable gestures from the 16 predefined classes were performed during the recording sessions. This deliberate avoidance of gestures aimed to create a distinct dataset of no-gestures, representing a wide range of natural and spontaneous non-gestural movements. Special attention was given to diversifying this no-gesture dataset to encompass various orientations, hand placements, and viewpoints.

The methodology consisted of recording the user's movements during the entire session, encompassing the period between the "record" and "exit" commands. Interactions involving the "start" and "stop" commands were deliberately omitted, as each window extracted from a session could be labeled directly as a `no-gesture` instance, avoiding the need for additional annotations.

A subset of the participants were used for this task, 7 in particular, who recorded a total of 5,592 samples of non-gestures. To prevent any cross-contamination, the dataset was partitioned on a per-user basis, distributed among the training (4,970 samples), development (262 samples), and testing (360 samples) sets. Furthermore, an equivalent number of gesture samples were included in each set, although these samples were labeled with a generic `gesture` label rather than the specific gesture class. As a result, the binary dataset encompassed a total of 9,940 training instances, 524 development instances, and 720 testing instances.

5.6.3. Binary gesture detection models

In this section, we explore a variety of models designed for the binary classification task of distinguishing between gestures and no-gestures. This task is crucial for efficient and fast gesture detection. Our goal is to strike a balance between model simplicity and performance, evaluating both lightweight models suited for fast decision making and the higher performing models from the previous section, to assess their suitability for this specific detection task. In the following, we delve into the details of each model.

Previous models adapted for binary gesture detection

These models share the same architectures as discussed in Section 5.5.1. However, a modification is applied to their final layers for the binary gesture detection task. Instead of the previous Fully Connected layer with neurons corresponding to the number of gesture classes and a softmax activation function, the final layer is substituted with a single-neuron FC layer featuring a sigmoid activation function. This change reconfigures the models into binary classifiers, allowing them to differentiate between gesture and non-gesture instances effectively. Specifically, the models evaluated were:

- MultiLayer Perceptron (MLP)
- Residual Network 1D (ResNet-1D)
- ResNet-1D with transformations block (Transf + ResNet-1D)
- Parameter-Reduced (ParaRed)

SimpleDetectGestureNet: a simple and lightweight model

Moreover, we introduce a model named SimpleDetectGestureNet, specifically designed for efficient and effective binary gesture detection. The core structure of the SimpleDetectGestureNet is composed of a 1D convolutional layer with 32 filters, followed by a 1D max-pool operation to capture essential features. The resulting feature maps are then flattened into a one-dimensional vector, subsequently fed into a Fully Connected layer containing 32 neurons, activated by the ReLU function. These hyperparameters have been selected after a brief search for optimal settings. Finally, the output of this layer is connected to a single-neuron FC layer with a sigmoid activation function. This simple architecture aims to balance computational efficiency and classification performance, making it a suitable candidate for fast binary gesture detection.

Transfer learning for detection: adapting the best classification model

Finally, we also explored the transfer learning approach. Building on the success of our most efficient gesture classification model, we adapted it to the binary detection task. This adaptation was carried out in two distinct phases.

In the first phase, we loaded the pre-trained classification model (DA + Transf + ResNet-1D), removed its final layer containing neurons equal to the number of classes with a softmax activation, and replaced it with a single neuron Fully Connected (FC) layer with a sigmoid activation. The weights of all layers except the newly added FC layer were frozen to prevent major changes to the already learned features. This phase involved training for 50 epochs using dynamic learning rate scheduling, which ranged from 0.01 to 0.0001.

Subsequently, in the second phase, we unfroze all layers of the model and conducted a complete fine-tuning process. This phase encompassed 100 epochs of training, employing a much lower learning rate of 0.000001 in order to facilitate a gradual adaptation of existing weights and prevent abrupt changes that might compromise the model's prior knowledge and generalization ability.

5.6.4. Results and insights of detection models

After analyzing the different models, we evaluated them for the gesture detection task. Each model was trained using Data Augmentation (DA) since, as discussed in Section 5.5.2, it has been proven that it helps to improve the performance of these classifiers.

Model	Accuracy (± CI)
DA + MLP	0.93 ± 0.02
DA + ResNet-1D	**0.96 ± 0.01**
DA + Transf + ResNet-1D	0.91 ± 0.02
DA + ParaRed	0.88 ± 0.02
DA + SimpleDetectGestureNet	**0.96 ± 0.01**
DA + Pre-trained	0.92 ± 0.02

Table 5.3: Detection results of the different models on our binary dataset. It is shown the mean accuracy and 95% Confidence Intervals (CI) of ten independent training runs.

Table 5.3 shows the results of each detector's accuracy. Interestingly, models incorporating transformations did not manage as well in this context. The DA + Transf + ResNet-1D and ParaRed models showed comparatively weaker results. Although promising, the pre-trained transfer learning model reached a midpoint with an accuracy of 0.92. Given the performance of the DA + Transf + ResNet-1D model, we expect a similar result for this pre-trained model, though it improves it slightly.

Proving the effectiveness of simplicity, the MLP achieved a commendable accuracy of 0.93, underlining that simple models can work satisfactorily at this particular task. The best-performing models were ResNet-1D and SimpleDetectGestureNet, both with an accuracy of 0.96. Once again, ResNet-1D models demonstrate their skill in capturing temporal dynamics, while SimpleDetectGestureNet shows that meticulous hyperparameter selection can improve performance even in modestly designed models.

5.7 Efficiency analysis

In this section, we delve into the efficiency review of the developed models, including gesture classification and binary gesture detection. We focus on two key aspects: model parameters and inference time in both CPU and GPU, followed by a discussion about the real-time applicability of these models in Mixed Reality (MR) applications.

5.7.1. Model parameters and inference time

To evaluate the efficiency of our models, we analyze their model complexity in terms of the number of parameters and inference time. Model parameters enclose both trainable and non-trainable elements. Inference time, crucial for real-time performance, is estimated for a single input window ($batch\_size = 1$). For larger batches, times are similar thanks to parallelism, but being a model that will work in an online environment, we should consider the worst case where we only receive one window at a time.

In addition, it is vital to note that the first inference always takes much longer than the rest because the predict function is created during the first (and only the first) call to `predict_on_batch`[1]. Therefore, these results are the median (and not the mean) of 1000 inferences (by averaging the last 999 inferences, the resulting times are essentially identical). The results can be seen in Tables 5.4 (gesture classification) and 5.5 (binary gesture detection).

Model	# parameters	Time in CPU (ms)	Time in GPU (ms)
MLP	3,473,516	**6.7035**	**2.9768**
ResNet-1D	552,976	8.0527	4.9331
STRONGER	1,839,604	13.2428	8.8700
Transf + MLP	45,571,792	31.2486	12.7916
Transf + ResNet-1D	1,308,176	22.8349	13.8903
ParaRed	**262,992**	11.7466	12.9396

Table 5.4: Efficiency results for gesture classification models. The reported times represent the median of 1000 inferences.

Model	# parameters	Time in CPU (ms)	Time in GPU (ms)
MLP	3,466,001	6.4169	**2.9565**
ResNet-1D	551,041	7.8323	4.8938
Transf + ResNet-1D	1,306,241	25.7132	14.0536
ParaRed	262,497	11.8502	12.7840
SimpleDetectGestureNet	**169,313**	**5.3105**	3.0947
Pre-trained	1,306,241	23.9999	14.3335

Table 5.5: Efficiency results for gesture detection models. The reported times represent the median of 1000 inferences.

[1]For more details, see `make_predict_function` in the source code.

As it can be seen, in general, MLP models emerge as the most efficient in terms of inference speed, thanks to their simplicity and low computational complexity. Despite their memory-intensive nature due to the high number of parameters, MLPs achieve great times in both CPU and GPU execution. However, their performance trade-off becomes evident as their accuracy remains behind other models, particularly in gesture classification tasks. While MLPs offer competitive speeds, alternatives with notably higher precision are worth considering if temporal constraints allow.

Regarding gesture classification, models with transformations exhibit increased computational demands compared to the ResNet-1D, leading to slower inference times. Nevertheless, the ParaRed model balances accuracy and efficiency, achieving competitive speeds while outperforming the ResNet-1D in classification accuracy (see Table 5.2).

For binary gesture detection, the results of the same models are naturally similar, as the main distinction lies in the complexity of their final layers. The pre-trained model, built upon the DA+Transf+ResNet-1D architecture, has a pretty analogous performance, making it of no interest due to better options, both in accuracy and inference speed. On the other hand, the SimpleDetectGestureNet has achieved its intended objective, being the one with the least parameters, even less than the ParaRed. Moreover, it is the fastest on CPU, surpassing the MLP and closely rivaling GPU speeds.

Finally, a remarkable thing about the ParaRed is worth mentioning: it is the only one that shows superior CPU performance compared to GPU execution, potentially due to the CPU-GPU information transfer overhead.

5.7.2. Real-time applicability in MR applications

The efficient deployment of gesture recognition models within Mixed Reality environments presents a significant challenge, as real-time performance is essential to deliver seamless and immersive user experiences. While deploying these ML models onto dedicated servers rather than the HoloLens 2 itself mitigates device resource concerns, the critical factor lies in the inference speed.

In addressing real-time feasibility, two key components deserve close consideration: the collective inference time of the chosen models and the potential delay introduced by the communication between the HoloLens 2 and the server. Although we do not have data on the exact communication delay because it may depend on many factors, its impact on the overall system's latency underscores the need to consider it.

Opting for the hierarchical modeling strategy promises good results. Using the fast and lightweight SimpleDetectGestureNet for detection (5.3 ms on CPU) and ParaRed for classification (11.7 ms on CPU) yields a combined processing time of approximately 17 ms per gesture inference. And note that this situation represents the worst-case scenario. In an MR application context, the analyzed windows are more likely to comprise non-gesture instances primarily. Consequently, the detector can quickly discard these windows without needing to be processed by the classifier. This inherent bias toward non-gestural instances further improves the feasibility of the real-time system.

Nevertheless, it should be noted that this time is without considering the delay of the transmissions. Given that the HoloLens 2 can sample the hand joints at 30Hz, corresponding to a time between frames of 33.33 ms, we still have a margin of about 16 ms for the potential communication overhead. Therefore, maintaining real-time throughput that matches the 30 Hz capture rate of HoloLens 2 so that windows overlap up to a single frame separation appears achievable.

5.8 Combined gesture recognition system

In this section, we delve into the development and evaluation of our combined gesture recognition system, which integrates both the binary gesture detection model and the specific gesture classification model in a hierarchical manner. This approach aims to optimize the trade-off between real-time performance and accurate gesture recognition within Mixed Reality applications.

5.8.1. Decision-making process

As we have already mentioned throughout this chapter, the decision-making process of our combined gesture recognition system is a crucial aspect that involves two sequential stages: gesture detection and gesture classification. This hierarchical approach ensures that predictions are fast enough, while maintaining accurate recognition. The basic workflow is illustrated in Figure 5.7.

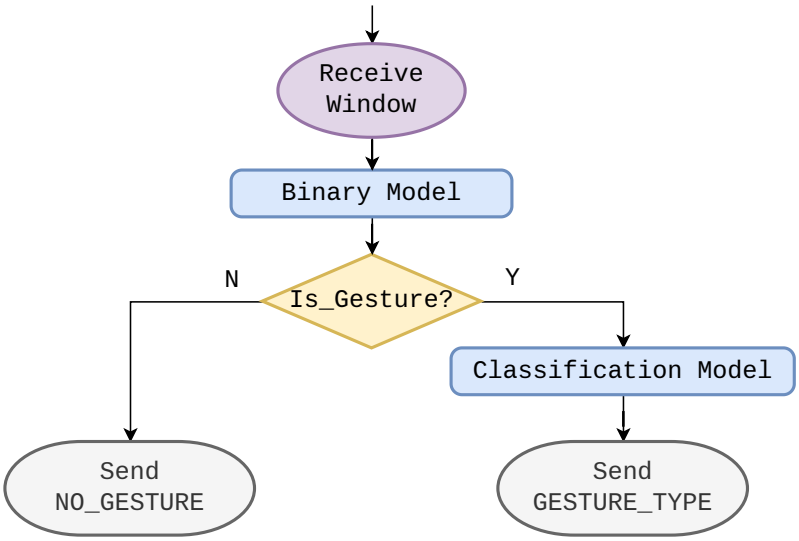


Figure 5.7: Basic hierarchical workflow of the combined gesture recognition system.

During the gesture detection stage, the incoming windowed data is first analyzed using the binary gesture detection model. If a gesture is detected, the data proceeds to the gesture classification stage. At this point, the specific gesture is identified using the gesture classification model. In addition, further complexity can be introduced by incorporating a confidence threshold based on the softmax probability of the classifier. This threshold determines whether the prediction of a gesture is reliable enough to classify it as such or not, even if the detector initially identifies it as a gesture. This approach takes advantage of the tendency of Neural Networks to exhibit overconfidence in their predictions [43] so that if it is low, it is plausible that it is a non-gesture, thus increasing the overall robustness of the system.

5.8.2. System evaluation and threshold analysis

To evaluate the system's performance as a whole, we must first establish the value of its hyperparameters. Assuming the approach that considers low softmax probabilities from the classifier as non-gesture predictions, we have to determine the value of two thresholds: those of the detector and the classifier.

For the detector, we performed an analysis of the Receiver Operating Characteristic (ROC) curve derived from our best model on the test set, setting the binarization threshold where the False Positive Rate (FPR) equals the False Negative Rate (FNR). This critical equilibrium point occurs at a threshold of 0.5516, with FPR = FNR = 0.0333, as depicted in Figure 5.8.

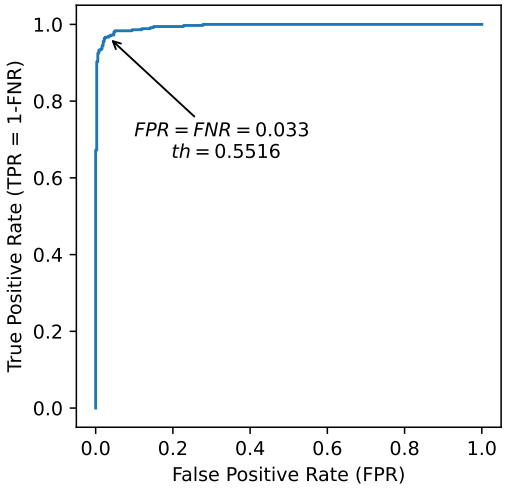


Figure 5.8: ROC curve for the binary gesture detection model.

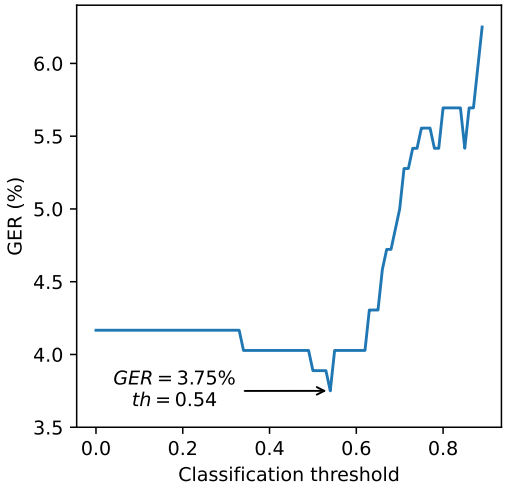


Figure 5.9: Optimal threshold analysis for the gesture classifier model.

On the other hand, for the gesture classifier, we study the optimal value of the threshold according to the improvement achieved in the system's overall performance. To achieve this goal, we formulated a metric called Gesture Error Rate (GER) to evaluate the effectiveness of the entire system. When analyzing a given data window, five different situations can occur, each corresponding to different outcomes:

- False Positive (FP): Occurs when a no-gesture is erroneously detected as a gesture.
- False Negative (FN): Arises when a gesture is mistakenly classified as a no-gesture.
- Classification Error (CE): Represents the misclassification of a particular gesture (e.g., identifying Gesture_X as Gesture_Y).
- True Positive (TP): Entails the accurate detection and classification of a gesture.
- True Negative (TN): Denotes the precise detection of a non-gesture.

The Gesture Error Rate (GER) is thus defined as the ratio of the sum of False Positives, False Negatives, and Classification Errors against the total number of windows processed. Mathematically, it can be expressed as:

$$GER = \frac{FP + FN + CE}{total\_windows\_processed}$$

This metric takes into account the impact of all possible errors, providing a solid assessment of the performance of the complete, combined gesture recognition system.

Hence, to determine the optimal threshold for the gesture classifier, we analyzed its GER in relation to different threshold values. This evaluation allowed us to understand how the GER changes with varying thresholds, representing the trade-off between the

different types of errors. Figure 5.9 presents the resulting curve depicting this relationship. As it can be seen in the figure, the optimal threshold was identified as 0.54, yielding a notable system-wide performance over test with a GER of 3.75%.

This result proves that considering gestures with low softmax probabilities as non-gestures can improve the overall performance of the combined gesture recognition system. Not setting it would be equivalent to setting it to zero, and, as it can be seen in Figure 5.9, this would give us a worse GER of 4.17%.

5.9 Deployment: proof-of-concept MR app

In this section, we delve into the deployment aspects of our system, outlining the communication framework and introducing the custom Mixed Reality application designed to validate the functionality of the complete gesture recognition system.

5.9.1. System deployment

Enabling efficient and smooth real-time gesture recognition within the MR environment involves a carefully considered deployment strategy. In our case, we chose not to perform inference directly on the HoloLens 2 device, although it is possible with some challenges [45]. This decision was primarily driven by the concern of potential performance degradation, where device saturation could lead to intermittent stuttering of the Mixed Reality interface, compromising the user experience. Instead, we designed an alternative approach.

In our deployment scheme, we set up a dedicated server responsible for conducting the inference. This architecture reduces the computational load on the HoloLens 2, as it just sends the joint information and awaits the class label response. This approach aligns with conventional communication patterns in MR applications and allows for a simple integration. Furthermore, this configuration also ensures model accessibility, as the server uses standard Python code with Keras models in the well-known `.h5` format.

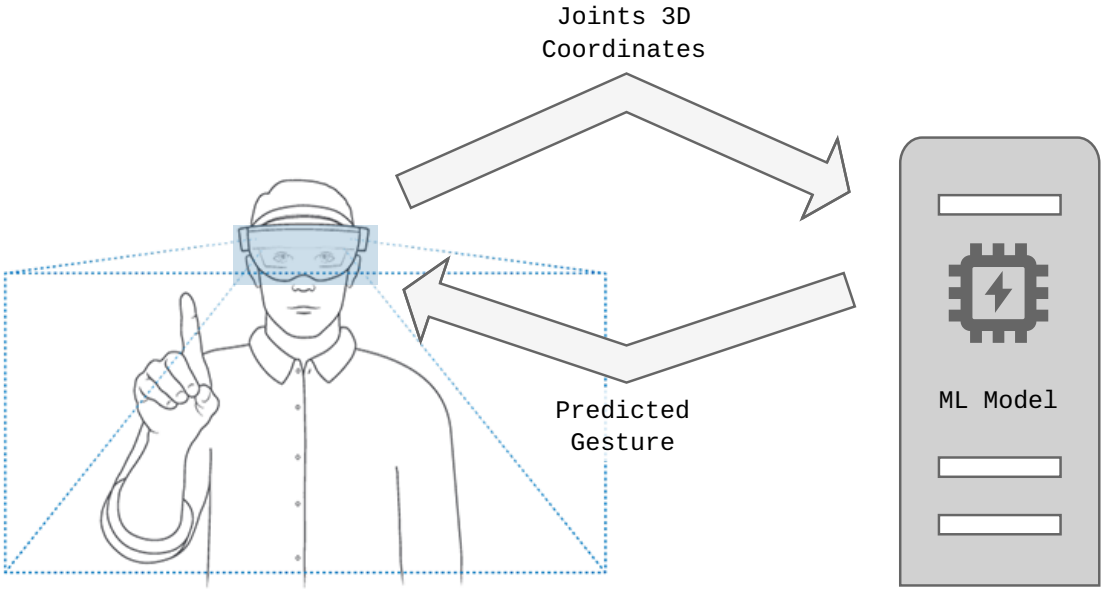


Figure 5.10: Communication architecture of the combined gesture recognition system.

In practice, the interaction proceeds as follows: the HoloLens 2 transmits joint data to the server, which then performs the hierarchical inference proposed in Section 5.8.1. The resulting label is returned to the application, enabling real-time responses such as scene transitions or game initiation. To facilitate this communication, we employed ZeroMQ (ZMQ) protocols over TCP packets designed for low-latency interactions. Figure 5.10 visually represents this interaction concept.

With this setup, the MR application can take advantage of the capabilities of the combined gesture recognition system without compromising the immersive experience. Furthermore, the modular design allows for easy adjustments and extensions to fit various scenarios, demonstrating the versatility and potential of the proposed solution.

5.9.2. Proof-of-concept MR app

Finally, we have developed a simple Mixed Reality application to serve as a practical demonstration of our proposed system. While not intended for final deployment or exhaustive evaluation, this app allows us to explore the feasibility and performance of our approach. The app design is minimalist, reusing the interface of our gesture capture app. Instead of providing a gesture proposal, it displays the inference results and argmax values of the probability distributions computed by our ML models.

The video preview window has been removed in this interface, and a small keyboard has been integrated to input the server's IP address. Figure 5.11 illustrates a Mixed Reality view, presenting a combination of real-world and holographic elements. The user's hand joints, detected by the HoloLens 2 device, are overlaid on their hand, and the outcome of the gesture recognition process is projected onto a nearby wall.

Figure 5.11: Mixed Reality view of the proof-of-concept application. We can observe an example of use, with a hand performing a `Thumbs up`, the overlapping of the detected joints, and the result of the inference of this gesture.

For the sake of simplicity in this proof-of-concept version, we have excluded the consideration of window overlapping (the overlap issue is discussed in Section 6.4). Hence, the HoloLens 2 device captures an entire window of 76 frames before transmitting the 76×78 tensor (26 joints per 3 space components each yields 78 floats per frame) to the server for inference. Upon completion, the inference results are sent back to the device for displaying them in the user's holographic view. This process continues iteratively, with non-overlapping windows being processed by the server until the user issues a "stop" command to conclude the capture.

5.10 Discussion and analysis of results

The results obtained from the experimentation provide a complete overview of the performance and efficiency of the proposed models for gesture detection and classification in Mixed Reality applications. Evaluating accuracy and efficiency becomes crucial, as both aspects are essential to ensure a smooth real-time experience in MR environments.

In the gesture classification section, it is observed that MLP models, despite their computational efficiency, reach a lower precision level than more complex architectures with the transformations block, like ParaRed and ResNet-1D. While the latter shows impressive performance (98.5% accuracy), the balance between precision and speed must be considered based on the specific requirements of the MR application.

Regarding the binary gesture detector, the implemented models demonstrated optimistic capabilities in distinguishing between gestures and non-gestures, with the best model achieving an accuracy of 96%. Furthermore, the hierarchical approach adopted for the combined detection and classification system yielded encouraging results, with a GER of only 3.75%. Integrating the SimpleDetectGestureNet architecture for detection and ParaRed for classification proved to be an effective combination in terms of performance-speed trade-off. This choice is based on SimpleDetectGestureNet's ability to quickly discard non-gestures before passing them to a more exhaustive classification by ParaRed.

Implementing this solution in an MR proof-of-concept application highlighted the feasibility of the proposed approach, with ample room for exploration of more complex aspects like window overlap and optimization of communication between the device and the server.

In summary, this chapter underlines the importance of model selection and efficient strategies for gesture detection and classification in MR applications. These findings will serve as a foundation for future developments and refinements in the search for an optimal solution for smooth Human-Computer Interaction in Mixed Reality environments.

CHAPTER 6
Conclusions and future work

In this final chapter, we conclude our exploration of Hand Gesture Recognition in Mixed Reality applications. Based on the learnings acquired during the research, we comment on the achievements of our proposed framework, its alignment with relevant course subjects, and explore possible future lines of research.

6.1 Conclusions

Throughout this work, a complete exploration of gesture recognition and classification in the context of Mixed Reality applications has been performed. The primary objective was to design, implement, and evaluate a robust and efficient HGR system for real-time interaction in MR scenarios.

The investigation began with an in-depth review of the existing literature, which showed us a wide range of methodologies designed to address the complexities of gesture recognition. Each approach had distinct advantages and limitations, which led us to evaluate carefully which to use. We deliberately chose 3D joint-based recognition over more conventional methods based on RGB images. This decision was motivated by the compact and potentially efficient nature of joint representations, although less well studied (which also gives us a margin for research and innovation).

Within this domain, diverse paradigms emerged, encompassing classical methods such as Support Vector Machines (SVM) alongside contemporary techniques like Convolutional (CNN) and Recurrent (RNN) Neural Networks. With this in mind, we decided to study one-dimensional Convolutional Neural Networks (1D-CNNs). This architecture has several advantages, mainly its simplicity and efficiency for one-dimensional signal processing such as audio, text, and time series [14, 15, 16]. In our case, the continuous signal of the 26 joints over time. Hence, this type of network was postulated as the best option in terms of precision-speed balance.

When studying this network paradigm, we observed the prevalence of architectures incorporating transformational layers at the top of the network. These layers served to capture rich gestural embeddings. Being intrigued by this concept, we further investigated the generalizability of these transformation blocks across different network architectures. Our goal was double: to discern the adaptability of these transformation elements in various models and to determine the model that offers optimal performance.

The main obstacle was the scarcity of large datasets required for effective training of Deep Learning models. In the existing literature, the available datasets were notably deficient, characterized by a limited number of samples and variable joint detection paradigms. Against this challenge, our response was to create our own extensive dataset. To this end, we relied on the collaboration of 25 volunteers, each participating in five recording sessions of approximately 10 minutes each. This joint effort allowed us to collect an extensive corpus consisting of 8,589 gesture samples distributed in 16 different classes, together with 5,592 representative instances of non-gestural activity.

Once the data was collected and cleaned, we implemented different gesture classification and detection models, as well as our own transformation block, a translation block, and a self-made Data Augmentation. ResNet-1D, with its block of transformations, performed the best for the classification task, with an accuracy of over 98%. This success demonstrates the transformation block's adaptability to various architectural paradigms and reinforces its effectiveness in the classification domain.

In addition, we also designed a reduced-parameter version of this best model to evaluate the trade-off between efficiency and accuracy. In particular, our empirical results revealed that the simplified version maintained almost all its predictive effectiveness while showing a higher speed, especially when used over CPU, where the inference time was nearly halved.

In the binary gesture detection task, the SimpleDetectGestureNet emerged as the optimal choice due to its efficient performance, with an accuracy of 96%. Our basic network demonstrated superior accuracy and processing speed despite exploring different and more complex model configurations, outperforming the adapted architectures designed for multi-class classification. Notably, transformation modules yielded limited benefits in the binary context, where simplicity prevailed over complexity for optimal results.

Efficiency emerged as a critical factor for seamless integration in MR environments. This led to a complete analysis that resulted in the strategic combination of ParaRed and SimpleDetectGestureNet. Our hierarchical decision-making design effectively discards non-gesture windows, ensuring fast processing and resource allocation. Furthermore, the inference process was strategically offloaded to an external server, alleviating the computational load on the HoloLens 2. In the proposed framework, the device focuses on running the MR app and detecting the joints, transmitting the data to the server, and waiting for the classification label.

This combined system was rigorously assessed using the Gesture Error Rate (GER) metric (see Section 5.8.2), achieving an impressive GER of 3.75%. Furthermore, we performed a proof-of-concept experiment that effectively demonstrated the viability and feasibility of the designed framework for real-world utilization in Mixed Reality applications.

In conclusion, this work has analyzed the complexities of gesture recognition and classification in Mixed Reality applications. The various state-of-the-art Deep Learning techniques studied, the efficient model design, and the careful optimization have paved the way for a robust and effective interaction paradigm. The learnings from this research provide helpful guidance for developing future gesture-based systems, offering a further step toward the successful fusion of human actions and digital experiences in immersive environments.

6.2 Achievement of objectives

Regarding the objectives established in section 1.2, the following accomplishments have been achieved:

- **Heterogeneous gesture recognition system:** The system has been successfully implemented, recognizing both static and dynamic gestures.
- **Neural architectures study:** A thorough exploration of Neural Network architectures resulted in developing a ResNet-1D model with a transformation block. This model achieved an accuracy exceeding 98% in gesture classification.
- **Dataset development:** Creating a comprehensive dataset comprising 8589 gesture samples and 5592 no-gesture samples enabled robust training and validation of the Deep Learning models.
- **Gesture classifier:** Implementing the ResNet-1D model with the transformation block demonstrated its efficacy in accurately recognizing and classifying specific gestures from input data.
- **Binary classifier:** The development of the SimpleDetectGestureNet, a lightweight architecture, efficiently determined the presence or absence of gestures in joint data windows, with an accuracy of 96%.
- **Real-time system:** Implementing a hierarchical system using the two models with the best accuracy-speed trade-off has enabled real-time gesture recognition capabilities, even over CPU.
- **Evaluation and validation:** Rigorous evaluation and validation processes finally obtained a Gesture Error Rate (GER) of 3.75%, affirming the system's accuracy and robustness.
- **MR application integration:** The integration of the developed system within a basic Mixed Reality application showcased its potential for enabling natural and intuitive interactions in real-world scenarios.

In summary, the successful achievement of these objectives underscores the viability and effectiveness of the developed work, offering promising potential for seamless integration into various Mixed Reality applications.

6.3 Relationship with course subjects

The content and development of this project intersect with a range of key subjects within the field of Computer Science, Artificial Intelligence, Pattern Recognition, and Digital Imaging. Virtual and Augmented Reality (*Realidad Virtual y Aumentada*) has played a crucial role in conceptualizing the design and use of Mixed Reality, emphasizing the practical utility of integrating gesture recognition into immersive environments.

The principles covered in Computer Vision (*Visión por Computador*) significantly influenced the DL aspects of the work, including the utilization of advanced Deep Learning architectures, Data Augmentation techniques, and preprocessing methods. Additionally, concepts from Pattern Recognition and Machine Learning (*Reconocimiento de Formas y Aprendizaje Computacional*) have been useful in improving the accuracy and efficiency of gesture recognition algorithms.

Artificial Neural Networks (*Redes Neuronales Artificiales*) provided a solid basis for exploring and implementing various Neural Network models, including the key role of Convolutional Neural Networks in gesture classification. The study of Biometrics (*Biometría*) highlighted the importance of user comfort, confidence, and interaction efficiency, guiding decisions regarding developing a user-friendly and intuitive interaction approach within Mixed Reality scenarios.

Efficiency and real-time performance, crucial aspects drawn from Computer Graphics (*Gráficos por Computador*) and Graphics Programming (*Programación Gráfica*), significantly influenced the development of the hierarchical decision-making system and the optimization of the combined gesture recognition system. Insights from Game Engines (*Motores de Videojuegos*) facilitated the integration of the gesture recognition framework into a MR application using Unity, enabling a tangible understanding of real-world applicability.

The foundations of Machine Learning and Pattern Recognition explored in Applications of Pattern Recognition (*Aplicaciones de Reconocimiento de Formas*) provided the essential basis for designing and fine-tuning the gesture recognition models. Finally, the principles of Data Visualization (*Visualización de Datos*) played an essential role in developing effective visual representations, including the elaborate 3D joint visualizations that contribute to understanding the whole gesture recognition process.

In essence, the multidisciplinary nature of this project has been enriched by insights drawn from these various subjects, collectively contributing to the successful realization of a robust and efficient gesture recognition system within the domain of Mixed Reality applications.

6.4 Future work

The future trajectory of our gesture recognition system involves several challenging directions to be studied, each offering opportunities to improve its performance, versatility, and real-world applicability. In the following, we present possible new research directions, which could not be covered in this work due to the limited time duration of the project.

A first option is to augment the dataset to include left-hand gestures. This could be achieved through specialized operations like specular transformations or recording a new dataset dedicated to left-hand gestures. Moreover, the adaptability of the models should be investigated, ensuring that they can effectively recognize gestures from both hands. Another more straightforward strategy might involve incorporating an initial lambda layer that determines the handedness and, if required, performs mirroring.

The exploration of overlapping windows emerges as another interesting aspect. By introducing overlapping windows into the data streaming process, the system could improve the speed of gesture detection and benefit from a richer temporal context, improving the system's overall performance. Enhanced decision mechanisms that consider overlapping, such as a voting system or a selective update approach, could optimize the real-time efficiency of the system. Notably, such mechanisms could notify the HoloLens 2 device only when a substantial change in the recognition occurs, avoiding the continuous sending of messages that would happen with the overlapping of windows.

The potential of early detection within the gesture recognition process presents another route for exploration. The system could achieve reduced latency by introducing larger windows and early detection mechanisms while maintaining gesture recognition

accuracy. Furthermore, the contextual analysis of a sequence of windows could yield a more precise context-aware interpretation of gestures.

Another future research approach involves exploring a multimodal gesture recognition system that combines 3D joint data with RGB camera input, as the HoloLens 2 also has cameras and simultaneous acquisition could be done. This approach could improve gesture understanding by simultaneously capturing spatial and visual cues. However, the concurrent data capture raises questions about computational demands and trade-offs in real-time processing, so careful investigation is needed to balance data richness and efficiency.

A different avenue worth exploring is the investigation of model calibration techniques. Calibrating the gesture recognition models could improve confidence estimates and more accurate predictions. The overall system's reliability and performance could be further optimized by aligning the model's output probabilities with the true likelihood of correct classifications (see Section 5.6.1). This work could involve exploring methods like Platt scaling or Isotonic regression [43], contributing to a refined and well-calibrated gesture recognition system for practical deployment in MR scenarios.

Continuing along the path of future directions, the concept of a unified multiclass model is also worth investigating. This model would incorporate all 16 gestures and the `no-gesture` class. While this approach offers the advantage of simplicity by employing a single model, it introduces challenges that deserve consideration. Evaluating the precision of this model will be crucial to determine if its accuracy improves or decreases compared to the hierarchical design. Moreover, the absence of a rapid discarding mechanism is a challenge, requiring a fast and efficient inference process.

Hardware deployment across diverse platforms, such as GPUs and Jetson devices, stands as another potential area for future investigation. Such deployment would offer insights into the scalability of the system and its adaptability to various computing environments. This analysis could show possible latency variations, especially in open networks. Additionally, exploring alternative communication protocols, such as UDP, could contribute to optimizing data transmission efficiency and reducing latency, further improving the real-time responsiveness of the system.

In addition, developing a fully featured Mixed Reality application represents a key future goal. This application could take full advantage of the gesture recognition system to enrich user experiences, including interactive games, immersive storytelling, or other unexplored forms of user participation. By designing and deploying a complete Mixed Reality application, the system's impact in real-world scenarios would be demonstrated.

Lastly, the potential for cross-domain applications should be noticed, as the gesture recognition system developed in this project could be adapted and integrated into various fields beyond Mixed Reality, including healthcare, education, and entertainment.

In conclusion, this work lays the foundation for future advancements in gesture recognition for Mixed Reality applications. While substantial progress has been made, numerous untapped opportunities remain for further improvements and innovations. As technology advances and research continues, these potential directions promise to refine the proposed system, enhance its capabilities, and ultimately contribute to the seamless integration of intuitive and immersive interactions in Mixed Reality environments.

Acknowledgments

I would like to express my sincere gratitude to my parents, aunt, and entire family for always supporting, loving, and motivating me throughout this journey. My closest friends deserve special mention for their constant presence, offering both support and laughter, even during the most challenging times. I am also thankful for the friendly relationships shared with my classmates, whose collaborative spirit has sweetened my learning experience.

My genuine appreciation extends to the users who generously participated in the data collection for this research. Your willingness to contribute to the dataset recording was crucial to make this work possible. While data protection agreements prevent me from mentioning your names, please know that your involvement is deeply appreciated.

A special note of gratitude goes to the professors of the Master's Degree in Artificial Intelligence, Pattern Recognition, and Digital Imaging program at the *Universitat Politècnica de València* (UPV). Their dedication, enthusiasm, and kindness have greatly enriched my educational journey. I am particularly thankful to my UPV tutor for their feedback and guidance throughout this research, which has significantly contributed to the quality of this work.

I am also sincerely grateful to ValgrAI, whose commitment to promoting research and development on AI has been greatly helpful in the realization of these studies and my evolution as a professional in the domain.

Last but not least, my gratitude goes to the *Instituto Tecnológico de Informática* (ITI) for offering a collaborative platform for this project's development. I am thankful for the opportunity to contribute both as a dedicated intern and now as a proud employee. The collaboration with the ASTID department has been an enriching experience, and I sincerely appreciate the opportunity to develop this work jointly.

Among my coworkers at ITI, I want to express my sincere gratitude to both the HCI and SiDi departments, as well as to my ITI tutor, for their valuable insights and feedback. The indispensable help and friendly collaboration from my HCI colleagues in developing this work have been essential and an absolutely pleasant experience. Likewise, the unconditional support from my SiDi colleagues has been really invaluable. Working closely with both departments has enriched my knowledge and growth as a researcher and professional.

Each of these individuals and organizations has played a vital role in shaping my academic and personal development trajectory. Your contributions, no matter how big or small, have positively influenced this work, and for that, I am truly thankful.

Bibliography

[1] SkyQuest Technology. Global Mixed Reality Market Size, Share, Growth Analysis, By Component (Software and Hardware), By Application (Automotive and Aerospace, Healthcare) - Industry Forecast 2023-2030. Industry report, SkyQuest Technology, March 2023. Report ID: SQMIG45I2078.

[2] Biplab Chakraborty, Debajit Sarma, Manas Bhuyan, and Karl MacDorman. A Review of Constraints on Vision-based Gesture Recognition for Human-Computer Interaction. *IET Computer Vision*, 12:3–15, 11 2017.

[3] Jacob D. Benedict, Jacob D. Guliuzo, and Barbara S. Chaparro. The Intuitiveness of Gesture Control with a Mixed Reality Device. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 63(1):1435–1439, 2019.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017.

[7] Marco Emporio, Ariel Caputo, Andrea Giachetti, Marco Cristani, Guido Borghi, Andrea D'Eusanio, Minh Quan Le, Hai Dang Nguyen, Minh Triet Tran, Felix Ambellan, Martin Hanik, Esfandiar Nava-Yazdani, and Christoph von Tycowicz. SHREC 2022 track on online detection of heterogeneous gestures. *Computers & Graphics*, 107:241–251, 10 2022.

[8] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[10] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access*, 7:19143–19165, 2019.

[11] Tianmei Guo, Jiwen Dong, Henjian Li, and Yunxing Gao. Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724, 2017.

[12] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[13] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. CNN-based Segmentation of Medical Imaging Data. *CoRR*, abs/1701.03056, 2017.

[14] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 4 2021.

[15] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33:917–963, 7 2019.

[16] Yulan Li, Charlesetta Baidoo, Ting Cai, and Goodlet Kusi. Speech Emotion Recognition Using 1D CNN with No Attention. In *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, pages 351–356, 10 2019.

[17] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022.

[18] Kenneth Brezinski and Ken Ferens. Complexity-Based Lambda Layer for Time Series Prediction. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 2046–2052, 2021.

[19] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, 01 1994.

[20] Paul Milgram and Fumio Kishino. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12:1321–1329, 12 1994.

[21] Unity Technologies. XR Glossary. https://unity.com/how-to/xr-glossary#xr, Last accessed: July 2023.

[22] Ronald Tadao Azuma. *Predictive Tracking for Augmented Reality*. PhD thesis, University of North Carolina at Chapel Hill, USA, 1996. UMI Order No. GAX95-38370.

[23] Jannick P. Rolland, Richard L. Holloway, and Henry Fuchs. Comparison of optical and video see-through, head-mounted displays. In Hari Das, editor, *Telemanipulator and Telepresence Technologies*, volume 2351, pages 293 – 307. International Society for Optics and Photonics, SPIE, 1995.

[24] Mariano Banquiero, Gracia Valdeolivas, Sergio Trincado, Natasha García, and M. Carmen Juan. Passthrough Mixed Reality With Oculus Quest 2: A Case Study on Learning Piano. *IEEE MultiMedia*, 30(2):60–69, 2023.

[25] Lin Guo, Zongxing Lu, and Ligang Yao. Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review. *IEEE Transactions on Human-Machine Systems*, 51(4):300–309, 2021.

[26] Rafiqul Zaman Khan and Noor Adnan Ibraheem. Hand gesture recognition: a literature review. *International journal of artificial Intelligence & Applications*, 3(4):161, 2012.

[27] Marco Emporio, Ariel Caputo, and Andrea Giachetti. STRONGER: Simple TRajectory-based ONline GEsture Recognizer. In Patrizio Frosini, Daniela Giorgi, Simone Melzi, and Emanuele Rodolà, editors, *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, pages 109–118. The Eurographics Association, 2021.

[28] Zuzanna Parcheta and Carlos-D. Martínez-Hinarejos. Sign Language Gesture Recognition Using HMM. In Luís A. Alexandre, José Salvador Sánchez, and João M. F. Rodrigues, editors, *Pattern Recognition and Image Analysis*, pages 419–426, Cham, 2017. Springer International Publishing.

[29] Okan Köpüklü, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource Efficient 3D Convolutional Neural Networks. *CoRR*, abs/1904.02422, 2019.

[30] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks. *CoRR*, abs/1901.10323, 2019.

[31] Fan Yang, Sakriani Sakti, Yang Wu, and Satoshi Nakamura. Make Skeleton-based Action Recognition Model Smaller, Faster and Better. *CoRR*, abs/1907.09658, 2019.

[32] Hung-Jui Guo and Balakrishnan Prabhakaran. HoloLens 2 Technical Evaluation as Mixed Reality Guide, 07 2022. 10.48550/arXiv.2207.09554.

[33] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. Hand Gesture Recognition with Jointly Calibrated Leap Motion and Depth Sensor. *Multimedia Tools Appl.*, 75(22):14991–15015, nov 2016.

[34] Ariel Caputo, Andrea Giachetti, Franca Giannini, Katia Lupinetti, Marina Monti, Marco Pegoraro, and Andrea Ranieri. SFINGE 3D: A novel benchmark for online detection and recognition of heterogeneous hand gestures from 3D fingers' trajectories. *Computers & Graphics*, 91:232–242, 2020.

[35] Danilo Avola, Marco Bernardi, Luigi Cinque, Gian Luca Foresti, and Cristiano Massaroni. Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures. *IEEE Transactions on Multimedia*, 21(1):234–245, jan 2019.

[36] Mehran Maghoumi and Joseph J. LaViola. DeepGRU: Deep Gesture Recognition Utility. In *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I*, page 16–31, Berlin, Heidelberg, 2019. Springer-Verlag.

[37] Ariel Caputo, Andrea Giachetti, Simone Soso, Deborah Pintani, Andrea D'Eusanio, Stefano Pini, Guido Borghi, Alessandro Simoni, Roberto Vezzani, Rita Cucchiara, Andrea Ranieri, Franca Giannini, Katia Lupinetti, Marina Monti, Mehran Maghoumi, Joseph J. LaViola, Minh Quan Le, Hai Dang Nguyen, and Minh Triet Tran. SHREC 2021: Skeleton-based hand gesture recognition in the wild. *Computers & Graphics*, 99:201–211, 10 2021.

[38] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *CoRR*, abs/1801.07455, 2018.

[39] Microsoft. Microsoft Mixed Reality Toolkit Documentation: TrackedHandJoint Enum. https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities.trackedhandjoint?view=mixed-reality-toolkit-unity-2020-dotnet-2.7.0, Last accessed: July 2023.

[40] Tarek A. Atwan. *Time Series Analysis with Python Cookbook: Practical Recipes for Exploratory Data Analysis, Data Preparation, Forecasting, and Model Evaluation*. Packt Publishing, 2022.

[41] C. Feichtenhofer, H. Fan, J. Malik, and K. He. SlowFast Networks for Video Recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6201–6210, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[42] Microsoft Learn. Microsoft Mixed Reality Documentation: Coordinate systems. https://learn.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems, Last accessed: August 2023.

[43] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

[44] Cameron R. Wolfe. Confidence Calibration for Deep Networks: Why and How? https://towardsdatascience.com/confidence-calibration-for-deep-networks-why-and-how-e2cd4fe4a086, Last accessed: July 2023.

[45] Léon Lazar. Neural Networks on Microsoft HoloLens 2, 2021. Bachelor Thesis, University of Stuttgart.

[46] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[47] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[48] Yoshua Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[49] Richard Skarbez, Missie Smith, and Mary C. Whitton. Revisiting Milgram and Kishino's Reality-Virtuality Continuum. *Frontiers in Virtual Reality*, 2, 2021.

[50] Ronald T. Azuma. A Survey of Augmented Reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355–385, aug 1997.

[51] Carol Manetta and Richard A. Blade. Glossary of Virtual Reality Terminology. *International Journal of Virtual Reality*, 1(2):35–39, Jan. 1995.

[52] Xianlun Tang, Zhenfu Yan, Jiangping Peng, Bohui Hao, Huiming Wang, and Jie Li. Selective spatiotemporal features learning for dynamic gesture recognition. *Expert Systems with Applications*, 169:114499, 5 2021.

[53] Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *CoRR*, abs/1610.02136, 2016.

[54] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. MIXUP: Beyond Empirical Risk Minimization. *CoRR*, abs/1710.09412, 2017.

[55] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-Based Dynamic Hand Gesture Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1206–1214, 2016.

[56] Graziano Fronteddu, Simone Porcu, Alessandro Floris, and Luigi Atzori. A dynamic hand gesture recognition dataset for human-computer interfaces. *Computer Networks*, 205:108781, 3 2022.

APPENDIX A

Detailed architectures of top-performing models

In this appendix, we present complete visual representations of the architectures for our top-performing models. These diagrams are intended to provide a clear overview of network structures, allowing a better understanding of their complexity and design.

Main classifier: DA + Transf + ResNet-1D

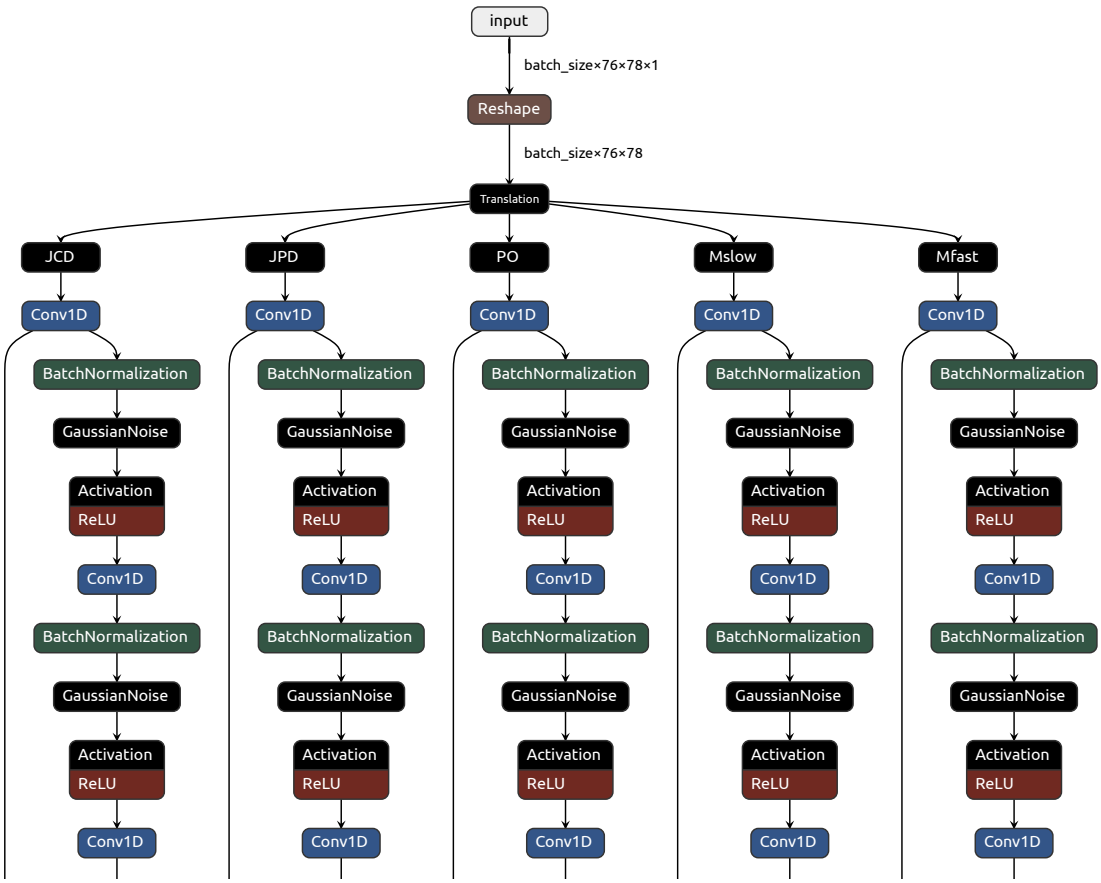


Figure A.1: Architecture of the main classifier (part 1).

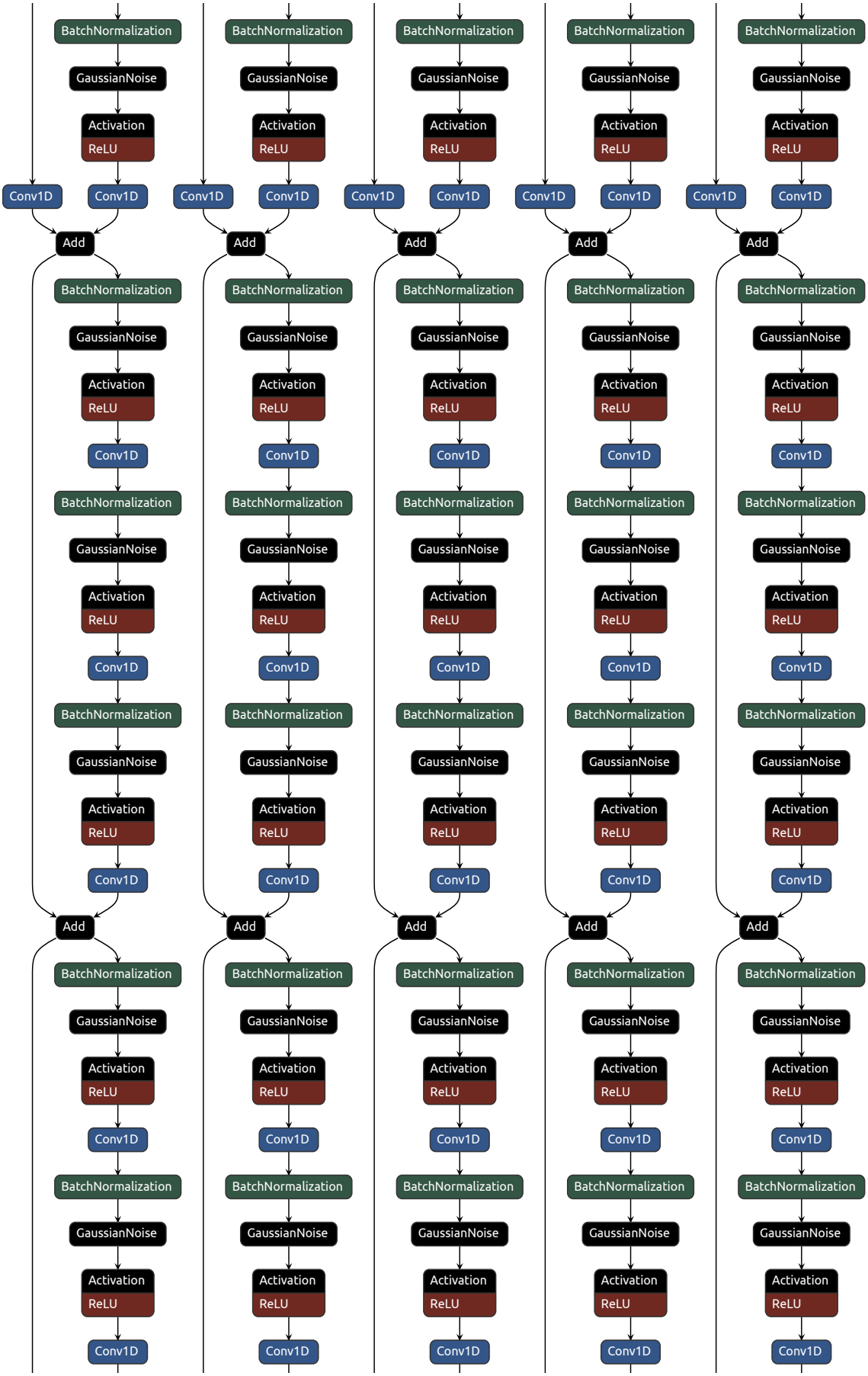


Figure A.2: Architecture of the main classifier (part 2).

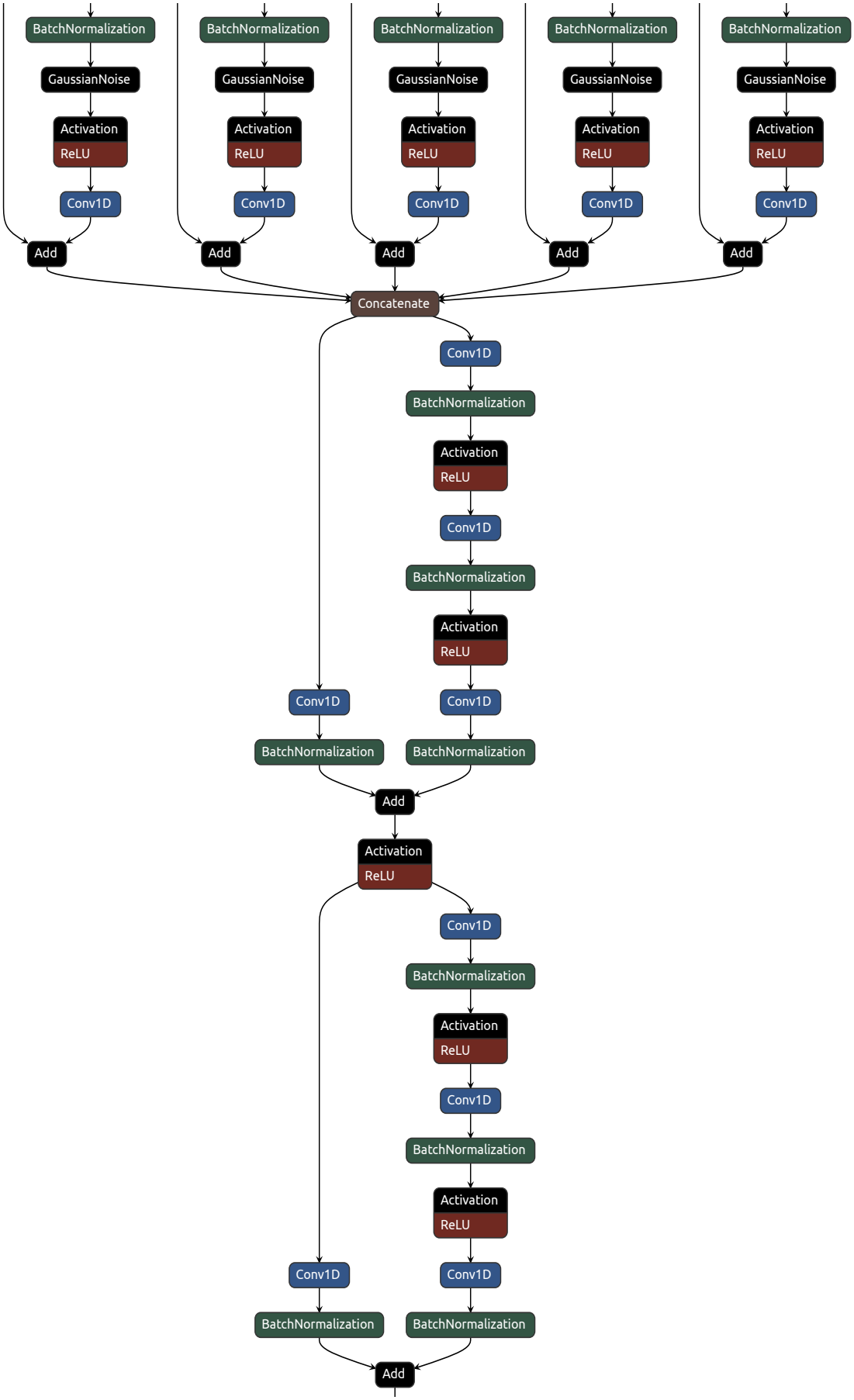


Figure A.3: Architecture of the main classifier (part 3).

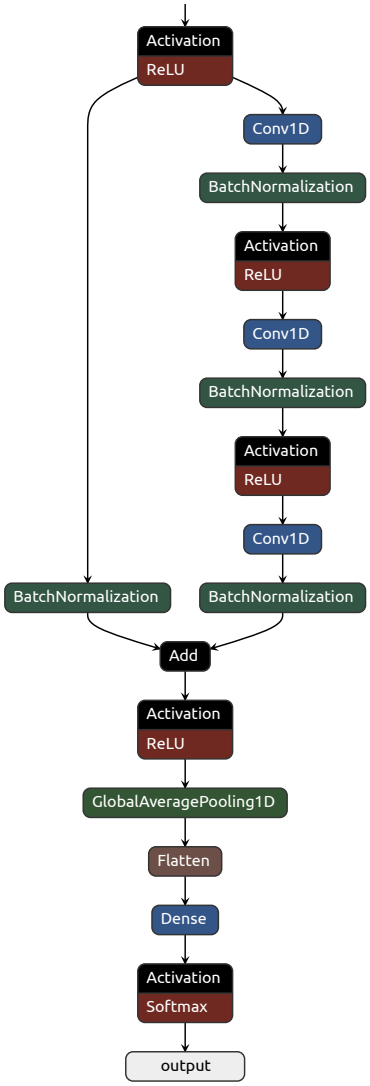


Figure A.4: Architecture of the main classifier (part 4).

Binary classifier: DA + SimpleDetectGestureNet

Figure A.5: Architecture of the binary classifier.

APPENDIX B

Confusion matrices of classification models

This appendix provides an overview of the confusion matrices obtained from evaluating the gesture classification models explored in this study. These matrices offer insight into the performance and accuracy of each model in correctly classifying gestures into different classes. The visual representation of these matrices is intended to aid in understanding which gestures the models can correctly classify, thus providing valuable context for comparative analysis. It should be noted that since the results in Table 5.2 are the mean of 10 independent runs, it is shown the confusion matrix of the model whose accuracy is more similar to the mean.

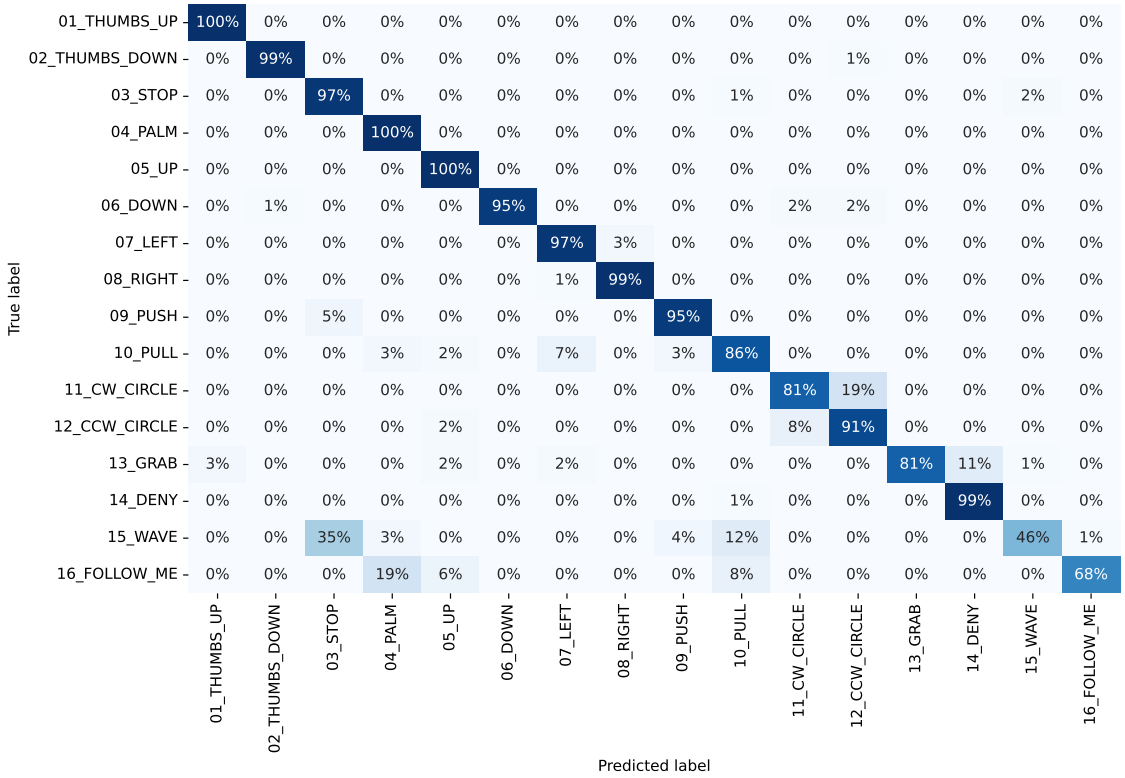


Figure B.1: Normalized confusion matrix for model MLP.

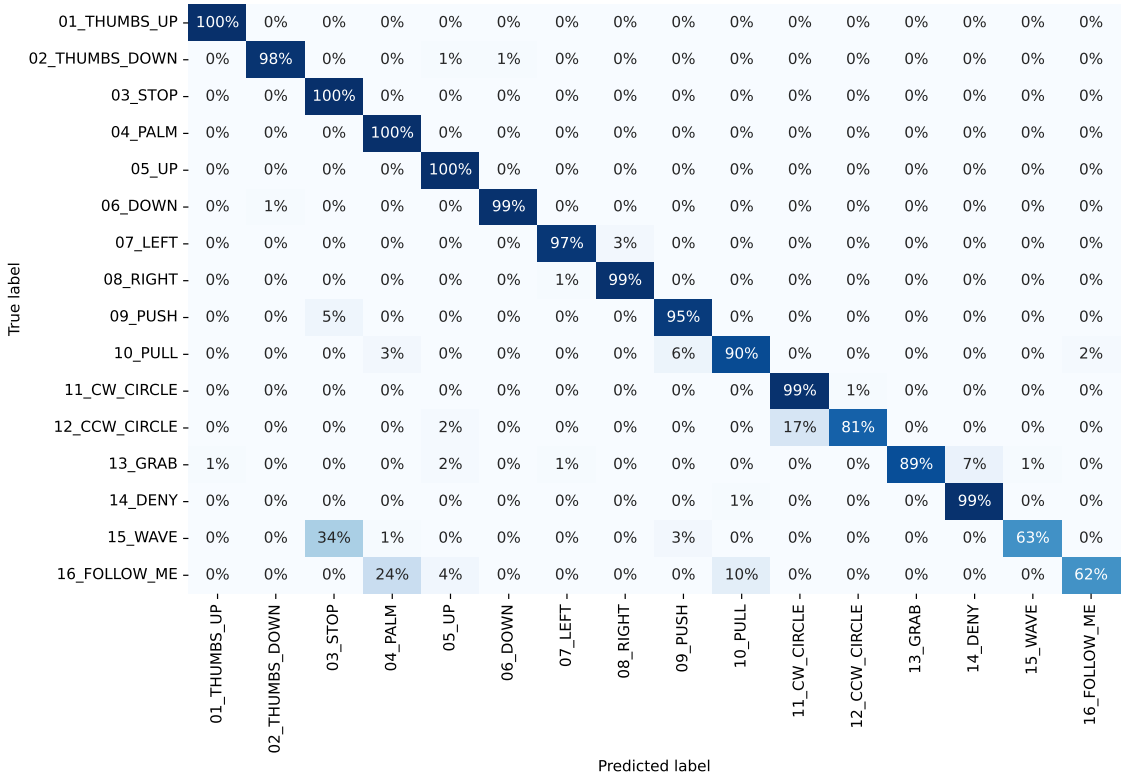


Figure B.2: Normalized confusion matrix for model DA + MLP.

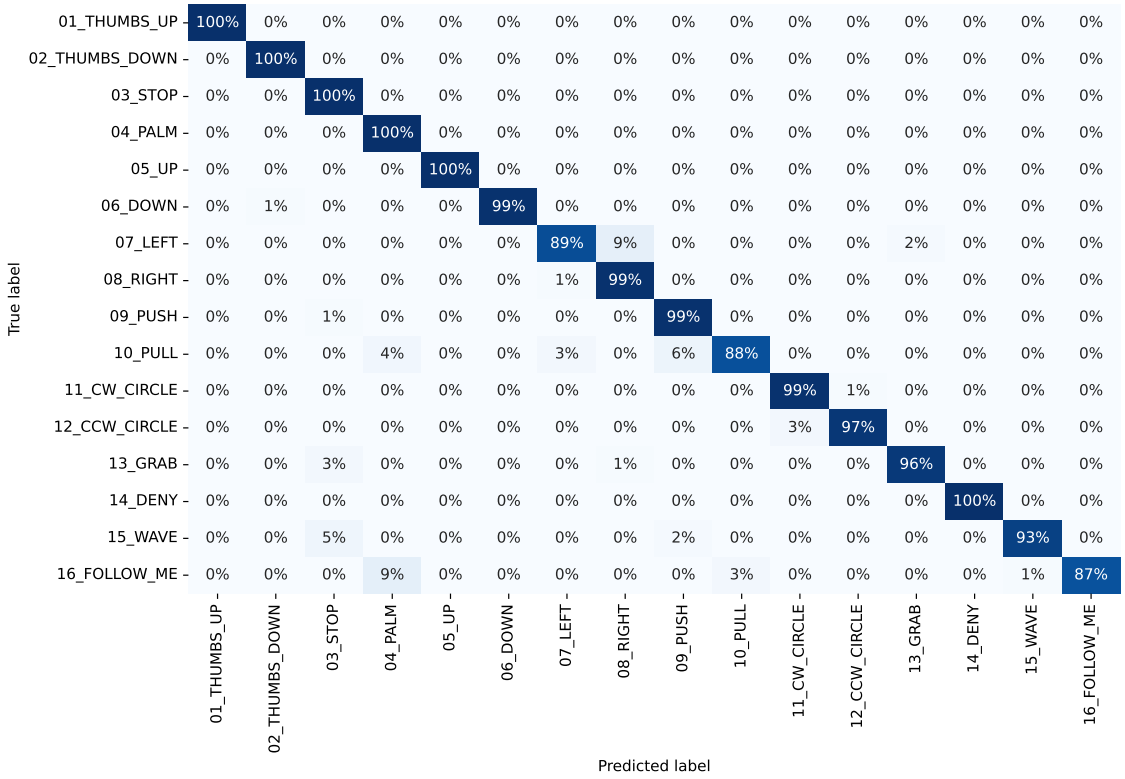


Figure B.3: Normalized confusion matrix for model ResNet-1D.

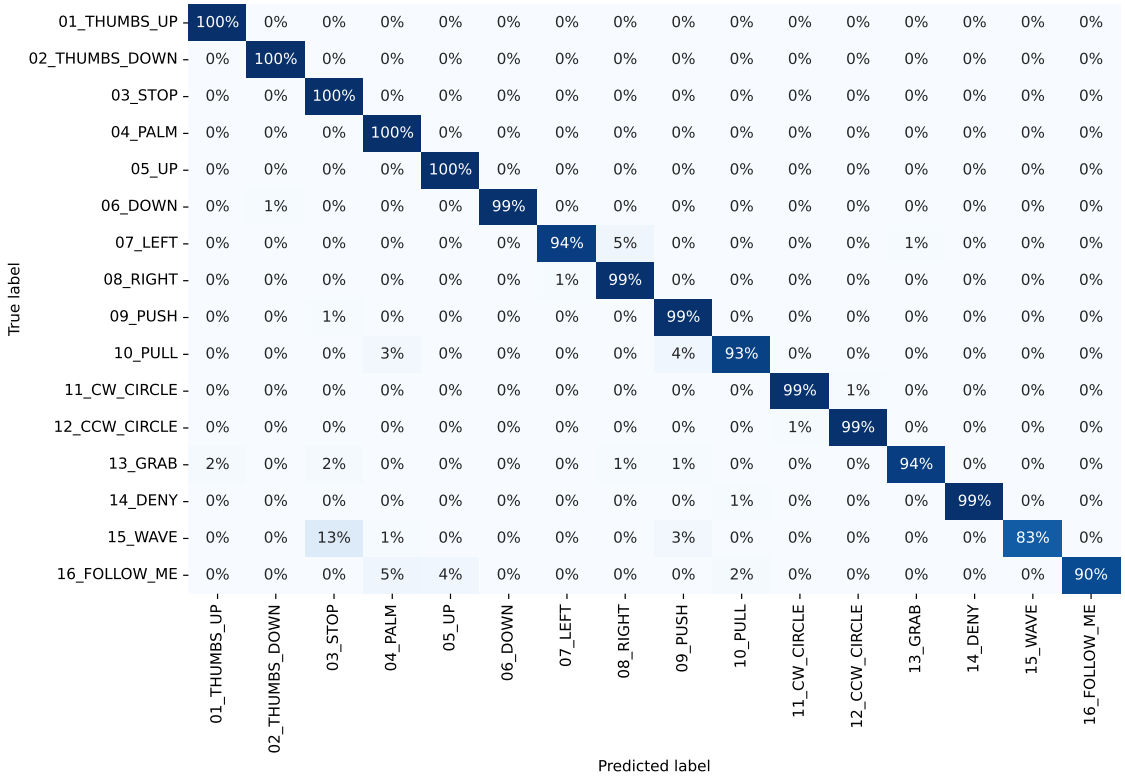


Figure B.4: Normalized confusion matrix for model DA + ResNet-1D.

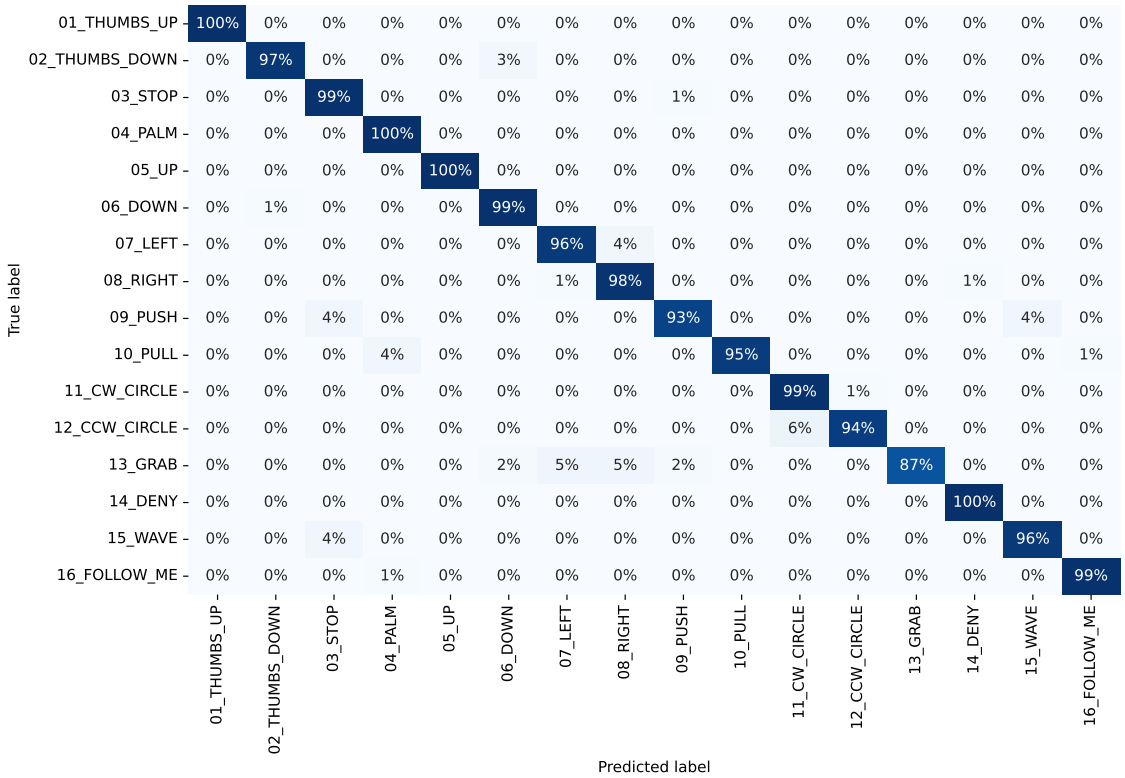


Figure B.5: Normalized confusion matrix for model STRONGER.

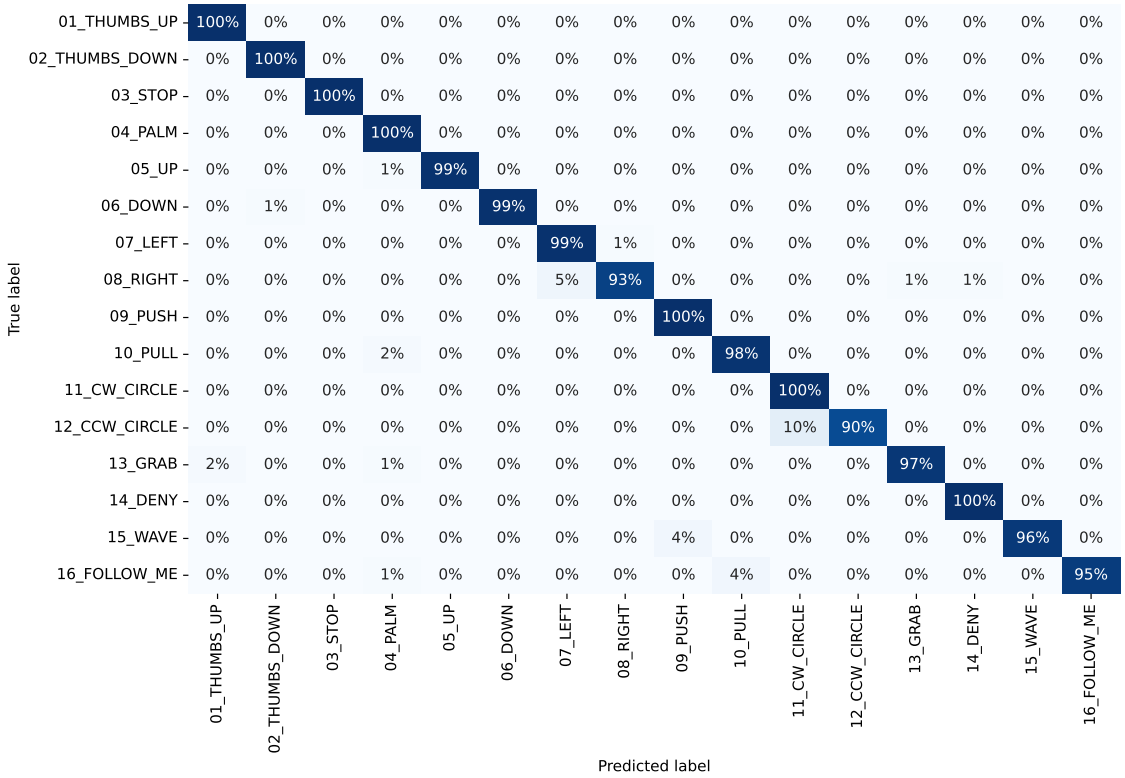


Figure B.6: Normalized confusion matrix for model DA + STRONGER.

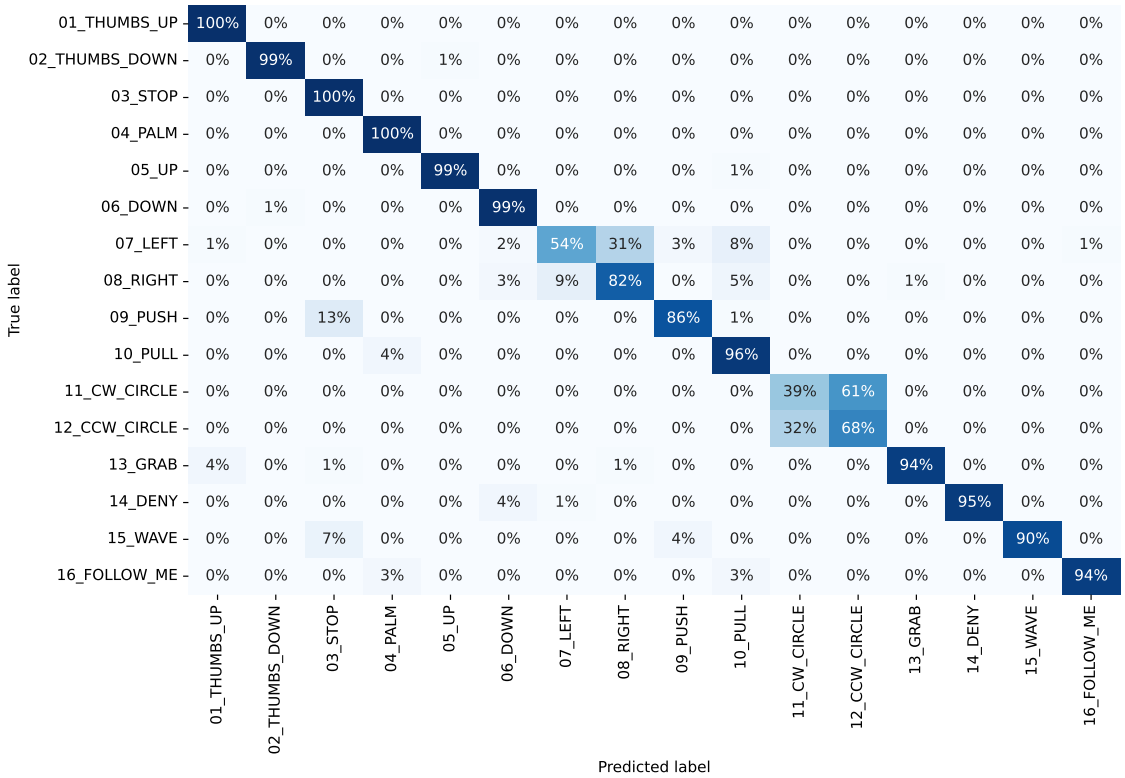


Figure B.7: Normalized confusion matrix for model Transf + MLP.

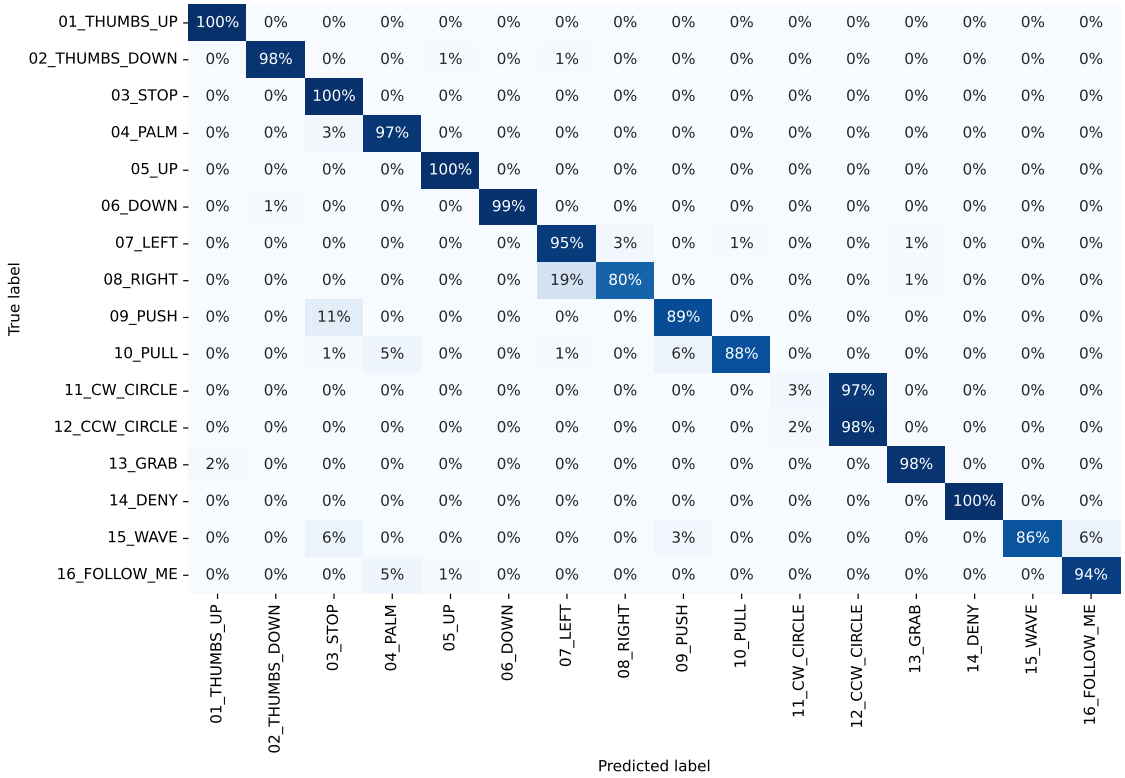


Figure B.8: Normalized confusion matrix for model DA + Transf + MLP.

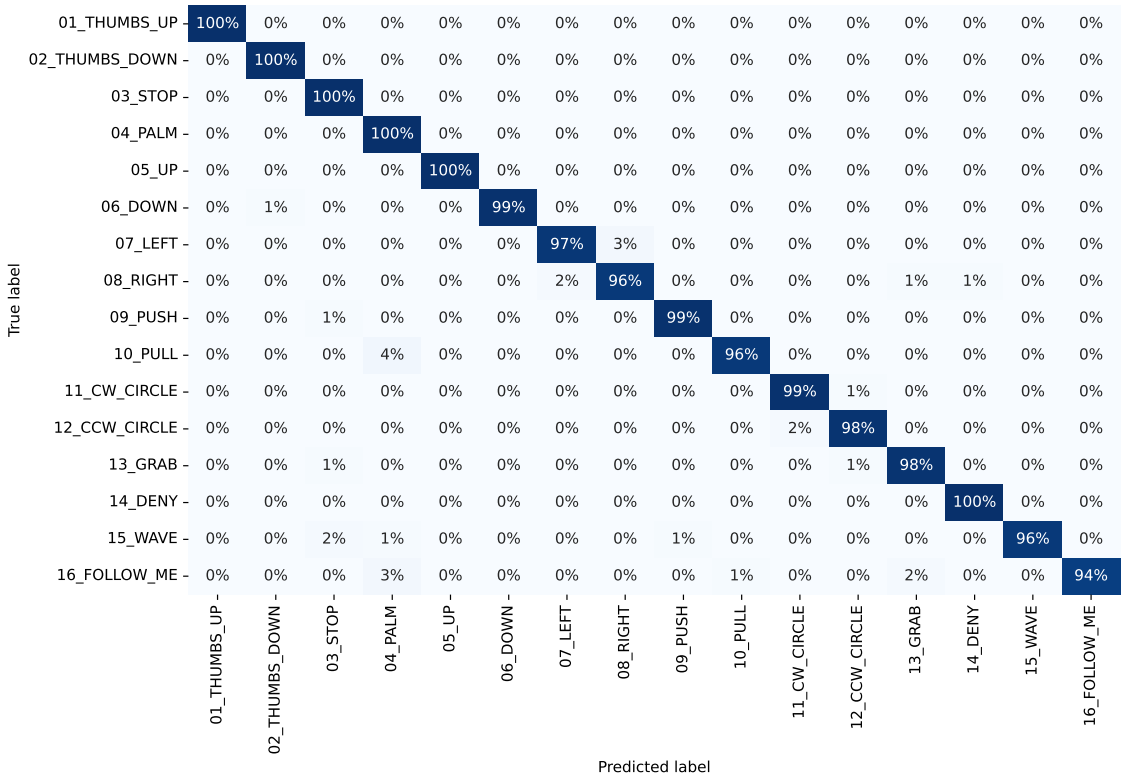


Figure B.9: Normalized confusion matrix for model Transf + ResNet-1D.

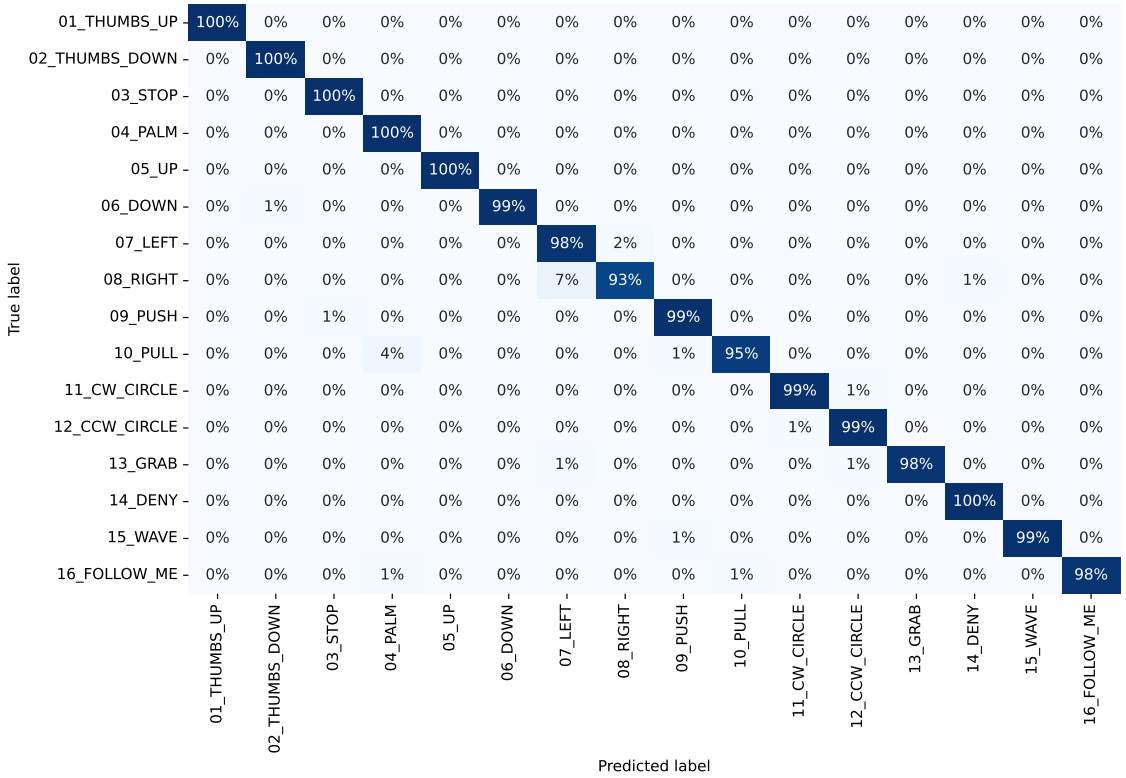


Figure B.10: Normalized confusion matrix for model DA + Transf + ResNet-1D.

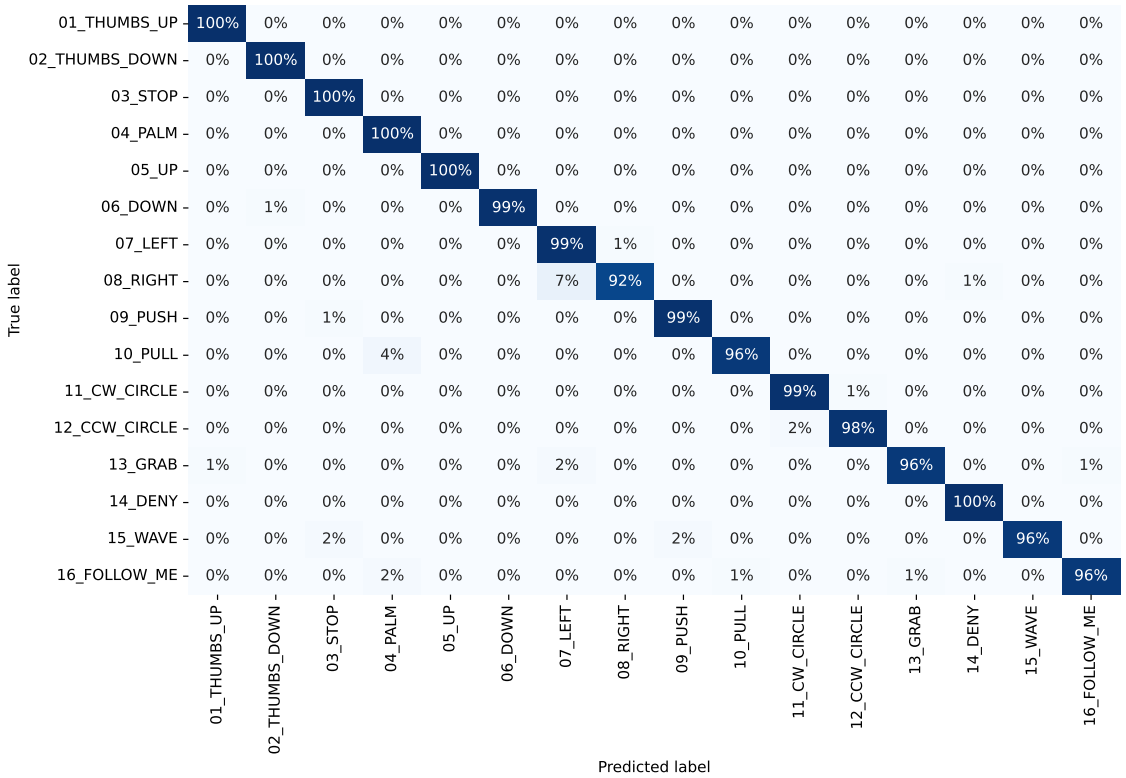


Figure B.11: Normalized confusion matrix for model DA + ParaRed.

APPENDIX C
Sustainable Development Goals

The United Nations' Sustainable Development Goals (SDGs) offer a complete framework for global sustainability. These goals cover a wide range of challenges, from poverty eradication to climate action. In the domain of technological research, our study intersects with some of these goals, as outlined in Table C.1. This alignment underscores the importance of our work in contributing to a more sustainable and inclusive future.

Sustainable Development Goals (SDGs)	High	Medium	Low	Not applicable
SDG 1. No poverty				✓
SDG 2. Zero hunger				✓
SDG 3. Good health and well-being	✓			
SDG 4. Quality education		✓		
SDG 5. Gender equality				✓
SDG 6. Clean water and sanitation				✓
SDG 7. Affordable and clean energy				✓
SDG 8. Decent work and economic growth		✓		
SDG 9. Industry, innovation, and infrastructure	✓			
SDG 10. Reduced inequalities	✓			
SDG 11. Sustainable cities and communities				✓
SDG 12. Responsible consumption and production			✓	
SDG 13. Climate action			✓	
SDG 14. Life below water				✓
SDG 15. Life on land				✓
SDG 16. Peace, justice, and strong institutions				✓
SDG 17. Partnerships for the goals				✓

Table C.1: Degree of relationship between the work and the United Nations Sustainable Development Goals.

In the first instance, we can examine the relationship between our project and SDG 9: *Industry, Innovation, and Infrastructure*. Our project inherently aligns with SDG 9, given its focus on designing and developing a gesture recognition system within the Mixed Reality (MR) domain. This system exhibits a clear potential for integration across various industrial sectors, acting as a catalyst for innovation and efficiency. In fact, one of our project's intended applications lies in robotic control in a warehouse.

Let us consider the scenario of warehouse operations, where our developed gesture recognition system has a strategic role. In this context, an operator equipped with HoloLens 2 would have the ability to interact seamlessly with a robotic agent in charge of tasks such as transporting boxes. Through the MR interface, the operator obtains a holographic overlay of pertinent information about the physical packages, clarifying their contents or indicating specific destinations within the warehouse. The operator's gestures, recognized by our system, emerge as an intuitive and effective means of communicating with the robotic entity through the glasses. Employing gestures, the operator can orchestrate a whole series of actions, from ordering the robot to rotate or move towards a specific direction. This dynamic combination of Mixed Reality and gesture-based interaction not only exemplifies innovation but also substantiates the practical synergy between our project and the realm of Industry 4.0, as championed by SDG 9.

Likewise, this work aligns with Sustainable Development Goal 4: *Quality Education*. Through the fusion of Mixed Reality and gesture recognition, our project offers promising ways to improve educational experiences. Imagine a classroom scenario where students equipped with HoloLens 2 devices participate in a captivating educational journey. Integrating our gesture recognition system empowers educators to navigate digital content, seamlessly enabling interactive and immersive learning. Students can see complex concepts come to life through holographic visualizations, while instructors can employ intuitive gestures to control the learning environment, ensuring a more attractive and dynamic pedagogical approach. By promoting technological literacy and building innovative educational tools, our work promotes equitable and quality learning opportunities, as advocated by SDG 4.

Continuing this trajectory, our project also intersects with Sustainable Development Goal 10: *Reduced Inequality*. By deploying Mixed Reality interfaces that facilitate natural and inclusive interactions, we aspire to bridge the gap between individuals with varying degrees of digital ability. Benefiting from gesture recognition technology, our system enables a diverse range of users, including those facing challenges in traditional interfaces, to interact effortlessly with digital environments. This inclusivity ensures that technology becomes an empowering force, breaking down barriers and reducing inequalities in accessing and benefiting from cutting-edge advancements. In this way, we contribute to the broader vision of creating a more inclusive and equitable digital world, aligning with the principles of SDG 10.

Moreover, our gesture recognition system is expected to be integrated into MR serious game applications to assist neurodivergent individuals. In scenarios like these, where maintaining focus on the therapeutic aspects of the game is crucial, traditional interfaces might prove distracting. The innate naturalness of gestures could play a key role, allowing users to interact with the game environment without being taken out of the immersive experience. By facilitating an unobtrusive and intuitive interaction mode, our system could significantly improve the efficacy of serious games for neurodivergent users, contributing to their concentration and overall well-being. This aligns with the principles of inclusivity and equality promoted by SDG 10 and furthers the goal of SDG 3: *Good health and well-being*, by trying to improve mental health and well-being for diverse populations.

In addition, our work also intersects with SDG 3 in other manners. While our primary focus has been on gesture recognition for MR applications, the implications of our technology extend to health-related scenarios. With the potential to minimize physical contact with devices and interfaces, our gesture-based system could reduce the spread of infections, especially in settings where shared touchscreens are prevalent. This aligns with the broader goal of safeguarding public health by offering alternative interaction methods prioritizing hygiene and well-being. By enhancing user experiences and ensuring safer interactions, our work aligns with the aspiration of SDG 3 to promote healthier lives and well-being for all.

Furthermore, this work holds relevance for SDG 8: *Decent work and economic growth*. Deploying gesture-based interaction systems like the one developed in this project could facilitate innovative applications in various industries. For instance, in manufacturing and logistics, as previously mentioned, integrating our system into robot control scenarios could improve efficiency and reduce the physical stress on workers by enabling intuitive and remote interaction. This could improve productivity and safer working conditions, aligning with sustainable economic growth and job creation goals. By exploring new ways of efficient human-machine interaction, we are addressing the dynamic needs of modern workplaces and advancing toward the objectives set by SDG 8.

Finally, our project intersects with SDG 12: *Responsible consumption and production* and SDG 13: *Climate action*. By developing gesture recognition models, we contribute to reducing electronic waste through a reduced dependency on physical interfaces and devices. This aligns with the principles of responsible production and consumption by promoting more sustainable patterns of technology utilization. Simultaneously, the pursuit of efficiency in our models fits with climate action objectives, as simple and optimized technology processes inherently consume less energy. This dual alignment underscores our commitment to advancing technology that enables environmentally aware practices.

In closing, the inherent alignment between our project and several Sustainable Development Goals highlights the long-term implications of our work. The potential to empower industries, improve accessibility for diverse populations, and support sustainable technological progress coincides with the global vision of the SDGs. By embracing these objectives, our study not only offers gesture recognition technology but also contributes to a broader, more sustainable future in which technology is used as a force for positive change.