



## DOCUMENT DELIVERABLE 29/02/2024

D3.2 - XR2Learn enablers  
WP3 – XR Technology PUSH  
February 2024

Author	UM
Work Package	WP3 – XR Technology PUSH
Delivery Date	08.03.2024
Due Date	29.02.2024
Classification	Public

## Status of deliverable

Action/role	Name	Date (dd.mm.yyyy)
Submitted by	Ioannis Chatzigiannakis (CNIT)	08.03.2024
Responsible (WP leader)	LS	29.02.2024
Approved by (internal reviewer)	Filisia Melissari (SYN) Ioannis Chatzigiannakis (CNIT)	29.02.2024

## Revision history

Date (dd.mm.yyyy)	Author(s)	Comments
04.12.2023	UM	Initial template
29.01.2024	LS, SUPSI, UM	Partner contributions
14.02.2024	LS, SUPSI, UM	Revisions
16.02.2024	UM	Version for the internal review
28.02.2024	UM	Preparing final version for submission

## Author(s) contact information

Name	Organization	E-mail
Enrique Hortal, Annanda Sousa, Bulat Khaertdinov	Maastricht University (UM)	<a href="mailto:enrique.ortal@maastrichtuniversity.nl">enrique.ortal@maastrichtuniversity.nl</a>

## - TABLE OF CONTENTS

<b>Table of Contents .....</b>	<b>4</b>
<b>List of Abbreviations.....</b>	<b>7</b>
<b>EXECUTIVE SUMMARY.....</b>	<b>9</b>
<b>1. INTRODUCTION.....</b>	<b>10</b>
<b>2. Authoring Tool: Enabler 1 .....</b>	<b>12</b>
2.1. INTERACT: Authoring Tool.....	12
2.1.1. Technical Documentation.....	12
2.1.1.1. Description .....	12
2.1.1.2. Prerequisites .....	13
2.1.1.3. Installation.....	13
2.1.1.4. Basic User Manual .....	14
2.1.1.5. Repository and license .....	19
2.2. Disclaimer.....	20
<b>3. Emotion Recognition Tools (Enablers 2-6).....</b>	<b>21</b>
3.1. Preliminary Research: Emotion Representation Learning and Emotion Recognition....	23
3.1.1. Speech Emotion Recognition .....	24
3.1.1.1. Speech Representations .....	24
3.1.1.2. Open-source Datasets and Experimental Setup.....	26
3.1.1.3. Evaluations .....	28
3.1.1.4. Challenges and Limitations .....	29
3.1.2. Emotion Recognition through Bio-measurements .....	30
3.1.2.1. Learning Representations from Bio-measurements.....	31
3.1.2.2. Open-source Datasets and Experimental Setup.....	31
3.1.2.3. Evaluations .....	32
3.1.2.4. Challenges and Limitations .....	34
3.1.3. Emotion Recognition using Body Tracking.....	35
3.1.3.1. Features and Classifiers.....	35
3.1.3.2. Open-source Datasets .....	36
Challenges and Limitations .....	39
3.2. Training Tools.....	40
3.2.1. Technical Documentation.....	40
3.2.1.1. Description .....	40
3.2.1.2. Prerequisites .....	42
3.2.1.3. Installation.....	42
3.2.1.4. Basic User manual .....	43
3.2.1.5. Open-source Code.....	44
3.2.2. Enabler 2: Emotion Representation Learning Tool .....	45
3.2.2.1. Description .....	45
3.2.2.2. Basic User manual .....	45
3.2.3. Enabler 3: Tools for Using Emotion Representations .....	46
3.2.3.1. Description .....	46
3.2.3.2. Basic User manual .....	47

3.2.4. Enabler 4: Unimodal Emotion Classification Tools .....	47
3.2.4.1. Description .....	47
3.2.4.2. Basic User manual .....	47
3.3. Inference Tools.....	48
3.3.1. Technical Documentation.....	48
3.3.1.1. Description .....	48
3.3.1.2. Prerequisites .....	49
3.3.1.3. Installation.....	49
3.3.1.4. Basic User Manual .....	49
3.3.1.5. Open-source Code.....	51
3.3.2. Enabler 5: Multimodal Fusion Tools .....	51
3.3.2.1. Description .....	51
3.3.2.2. Basic User Manual .....	52
3.3.3. Emotion Classification and Model Evaluation.....	52
3.3.3.1. Description .....	52
3.3.3.2. Basic User Manual .....	52
3.4. Personalization Tool .....	53
3.4.1. Technical Documentation.....	53
3.4.1.1. Description .....	53
3.4.1.2. Prerequisites .....	54
3.4.1.3. Installation.....	54
3.4.1.4. Basic User Manual .....	55
3.4.1.5. Open-source code.....	55
3.4.2. Enabler 6: Personalization tool.....	56
3.4.2.1. Description .....	56
3.4.2.2. Basic User Manual .....	56
3.4.3. Demo UI .....	57
3.4.3.1. Description .....	57
3.4.3.2. Basic User Manual .....	58
3.5. Command Line Interface (CLI).....	59
3.5.1. Technical Documentation.....	59
3.5.1.1. Description .....	59
3.5.1.2. Prerequisites .....	60
3.5.1.3. Installation.....	60
3.5.1.4. Basic User Manual .....	60
3.5.1.5. Open-source Code.....	61
<b>4. Data Acquisition .....</b>	<b>64</b>
4.1. Magic XRoom: Data Collection Tool .....	64
4.1.1. Technical Documentation.....	64
4.1.1.1. Description .....	64
4.1.1.2. Prerequisites .....	65
4.1.1.3. Installation.....	65
4.1.1.4. Basic User Manual .....	66
4.1.1.5. Open-source Code.....	77
4.1.1.6. Known Issues.....	78
4.2. Data Collection .....	80
4.2.1. Data Collection Protocol for Magic XRoom .....	80

4.2.2. Data Collection Pilots .....	83
<b>5. CONCLUSION .....</b>	<b>85</b>

## - LIST OF ABBREVIATIONS

BA	Beacon application
CNN	Convolutional neural network
EdTech	Educational Technologies
KPI	Key performance indicator
MFCC	Mel-frequency cepstral coefficients
MLP	Multilayer perceptron
SSL	Self-supervised learning
WP	Work package
XR	Extended reality
<b>Partners' names and acronyms</b>	
CNIT	CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI
F6S	F6S NETWORK IRELAND LIMITED
MAG	MAGGIOLI SPA
LS	LIGHT AND SHADOWS
SYN	SYNELIXIS SOLUTIONS SA
SUPSI	SCUOLA UNIVERSITARIA PROFESSIONALE DELLA SVIZZERA ITALIANA
UM	UNIVERSITEIT MAASTRICHT
HOU	HELLENIC OPEN UNIVERSITY

EADTU	EUROPEAN ASSOCIATION OF DISTANCE TEACHING UNIVERSITIES
EITM	EIT MANUFACTURING SOUTH SRL



---

## - EXECUTIVE SUMMARY

---

This deliverable, D3.2 XR2Learn enablers, outlines the development of innovative tools designed to foster the creation and integration of Extended Reality (XR) learning applications enriched by affective computing. The contributions of the proposed enablers are two-fold: to reduce the workload required for the development of XR learning applications and to promote the personalization and enhancement of the learning experience in an effortless and seamless manner. In the context of Task 3.2, the following tools have been developed:

- Enabler 1: The Authoring Tool, a key component that simplifies the creation of XR applications specifically tailored for educational purposes. This tool allows educators and developers to easily build immersive learning environments.
- Enablers 2-5: These enablers focus on the development of tools for automatic emotion recognition, utilizing various input data modalities. With the aim of facilitating the use of not only public datasets but also "in-house" data in an effortless manner, the Enablers 2-5 include:
  - Self-Supervised Learning: operates without the need for labeled data to pre-train Deep Learning models, allows to train models on emotions with less annotated data and resources.
  - Supervised Learning: requires labeled data and provides a structured approach to identify user emotions from input modalities.
- Enabler 6: This enabler represents the last step in the affective computing pipeline, effectively integrating the capabilities of the automatic emotion detection components (enablers 2-5) and facilitating the use of its output as a source for the adaptation of the learning material. Its primary function is to utilize the detected emotions of the user to suggest appropriate learning activities. This personalization aspect ensures that the learning experience is optimized for each individual, making it more engaging and effective.
- Magic XRoom: This innovative feature serves as a tool for collecting data. This data is crucial for the evaluation of the enablers, as it provides the necessary input for emotion detection algorithms.

In conclusion, the deliverable presents a set of novel enablers that can be utilized to accelerate the development of educational XR applications. Moreover, by integrating XR applications with required equipment, data collection modules and emotion recognition enablers, it paves the way for a more immersive, personalized learning experience that is adaptable to the emotional states of the users, thereby enhancing the overall effectiveness of the educational process.

---

## 1. INTRODUCTION

---

This deliverable reports the progress achieved in Task 3.2, "XR2Learn Enablers" during the first 14 months of the XR2Learn project. This covers the developments made during the first sub-phase of Phase B described in Section 1.2.2 of the proposal document. Being a part of Work Package 3 XR Technology Push, Task 3.2 is focused on designing, implementing and delivering novel enablers for XR applications. Thus, the concept of enablers is central to this task. Essentially, they are building blocks or components that can be used to speed up the development of XR-based applications. This, in turn, makes developing XR applications more manageable and cost-effective, turning these applications more attractive and accessible.

The XR2Learn project introduces two types of XR enablers, namely the authoring tool (**Enabler 1**; Section 2) and affect detection, or emotion recognition, enablers (**Enablers 2, 3, 4, 5, 6**; Section 3). As highlighted in the proposal document, the former is a *"tool for designing more efficiently 3D spaces (e.g. construction/manufacturing environments) under physical space constraints"*, whereas the latter aims to *"infer user perception such as affect detection, emotion recognition and AI based learner progress assessment"* using three modalities, namely bio-measurement, speech and/or body tracking signals. This document specifies the enablers and describes the progress of their development within the first phase of the task. The current progress in emotion recognition enablers is showcased using one modality, namely audio, acquired from open-source data. In the subsequent phase of the project other modalities (e.g. bio-measurements, body-tracking) are planned to be integrated with the enablers following the data collection required for these modalities.

We also report the work conducted on components and functionalities beyond the project proposal with regards to Task 3.2. They were identified by partners contributing to the task as necessary for the development of the enablers based on the conducted preliminary research and drafting of the initial system design. These components include:

- **Magic XRoom** (Section 4.1) is a data collection tool which is vital for acquiring training data for emotion recognition enablers in education settings. The preliminary research on open-source datasets, conducted in the first months of the project, identified multiple challenges that are faced when building models using these data. A common challenge for all modalities is that there is a lack of data corresponding to educational settings annotated with the appropriate emotional models that are focused on user engagement. Nevertheless, training Machine and Deep Learning models for emotion recognition requires some amounts of annotated data, which we propose to collect through Magic XRoom.
- **Command Line Interface** (Section 3.5) is an automated interface that provides users with a single entrypoint for Enablers 2-6 allowing them to execute the whole emotion recognition pipeline. Currently, CLI is integrated with Enablers 2-6, which supports audio modality provided from open-source data.
- **Demo User Interface (Demo UI)** (Section 3.4.3) is a web-based application implemented to display the communication between Inference and Personalization tools (Enabler 6), and the XR Unity application.

The contributions of this deliverable to other tasks and deliverables in Work Package 3 can be summarized as follows:

- **Task 3.1; Deliverable 3.1:** the design of the tools and components facilitate the integration of enablers with Beacon Apps. In particular, the Personalization Tool (Enabler 6) exploits the Publisher/Subscriber messaging protocol that is

intended to provide asynchronous communication between the tool and XR applications in Unity. Furthermore, a user-friendly Demo UI has been implemented to showcase this communication and workflow of the emotion recognition enablers.

- **Tasks 3.3, 3.4; Deliverable 3.3:** the implemented enablers are delivered as open-source code repositories with consistent documentation style and thorough README files to guide users through exploiting the enablers. These guidance documents will be further extended into the Wiki pages concerning XR2Learn enablers.

This document splits the reporting of technical work in five main sections that can be summarized as follows:

- **Section 2** describes the Authoring Tool (Enabler 1) starting with a description of this Enabler's motivation and main functionalities. This section then contains prerequisites, installation and basic user manual information.
- **Section 3** presents the technical design and development of emotion recognition tools, or Enablers 2-6. This section begins with preliminary research conducted on emotion representation learning and emotion recognition in audio, bio-measurements, body tracking modalities. The conducted analysis explores the effectiveness and efficiency of the state-of-the-art emotion recognition techniques with the specified modalities. Besides, it comments on challenges and limitations that can be faced when developing these models within Enablers 2-6. Furthermore, it describes each enabler and additional components developed to support the enablers' functionalities, including their application, prerequisites, installation instructions, a basic user manual and details on open-source licenses.
- **Section 4** introduces Magic XRoom as the suggested and developed data collection tool. This section also follows a similar structure summarizing the prerequisites, the instructions to set it up along with the required equipment, as well as the technical documentation of XRoom.
- **Section 5** summarizes the developments made and progress achieved in Task 3.2 up to Month 14 of the project. Furthermore, this section outlines a list of actions and tasks considered to achieve the two main objectives of the second sub-phase of Phase B: improving enablers and integrating them with Beacon applications.

---

## 2.AUTHORING TOOL: ENABLER 1

### 2.1. INTERACT: AUTHORING TOOL

#### 2.1.1. Technical Documentation

---

##### 2.1.1.1. Description

To meet the evolving demands of learning environments and educational training needs, there is a growing requirement for authoring tools to allow rapid design and development of XR training scenarios built upon established frameworks. It is, therefore, crucial to develop authoring tools that enable the swift creation of training scenarios supporting a broader range of features beyond basic 3D object manipulation and avatar navigation in virtual environments. These tools should address aspects such as ergonomics, advanced physics for objects, and scenario creation.

To achieve this goal, the first enabler provided to the XR2Learn community is an authoring tool called INTERACT, delivered as a Unity plugin that can significantly reduce the time needed to develop intelligent tutoring systems (ITS). The plugin is a no-code (or low-code) generic tool for creating physics-based VR training scenarios. INTERACT is based on a cutting-edge physics engine, allowing realistic interactions such as collision detection and ergonomic evaluations. The plugin empowers users to create physically realistic VR simulations for diverse applications, including training in heavy industry, education, and energy sectors, using 3D data (such as CAD or point clouds) imported into the authoring tool. We demonstrated the practical application of INTERACT by developing Beacon Application 1 (consult Deliverable D3.1 for further information), a training program for a laser cutting machine. The resulting XR application offers a virtual reality training scenario focusing on machine maintenance tasks. Users are guided through a step-by-step process of using a laser cutting machine, with their performance validated through a final score.

The main features are summarized as follows:

- Embedded physics engine: handling multi-body dynamics, collision detection, friction, and kinematics, providing realistic behavior for objects in a 3D environment.
- Advanced collision detection: This feature allows accurate and efficient detection of collisions between objects, even when dealing with complex models.
- Cables: The software can simulate cables and flexible beams using finite element analysis, providing realistic representations of these elements in the 3D environment.
- Grab: This feature allows users to manipulate 3D objects directly with their hands, providing a more natural and intuitive way to interact with the virtual environment whether using VR controllers or hand-tracking systems.
- Scenarization: This module is designed for assembly training, which allows creating and editing complex assembly scenarios. This module supports gamifying scenes and creating a pedagogical scenario through a node-based graphical interface.



- Figure 1. List of modules and features inside the INTERACT authoring tool

### 2.1.1.2. Prerequisites

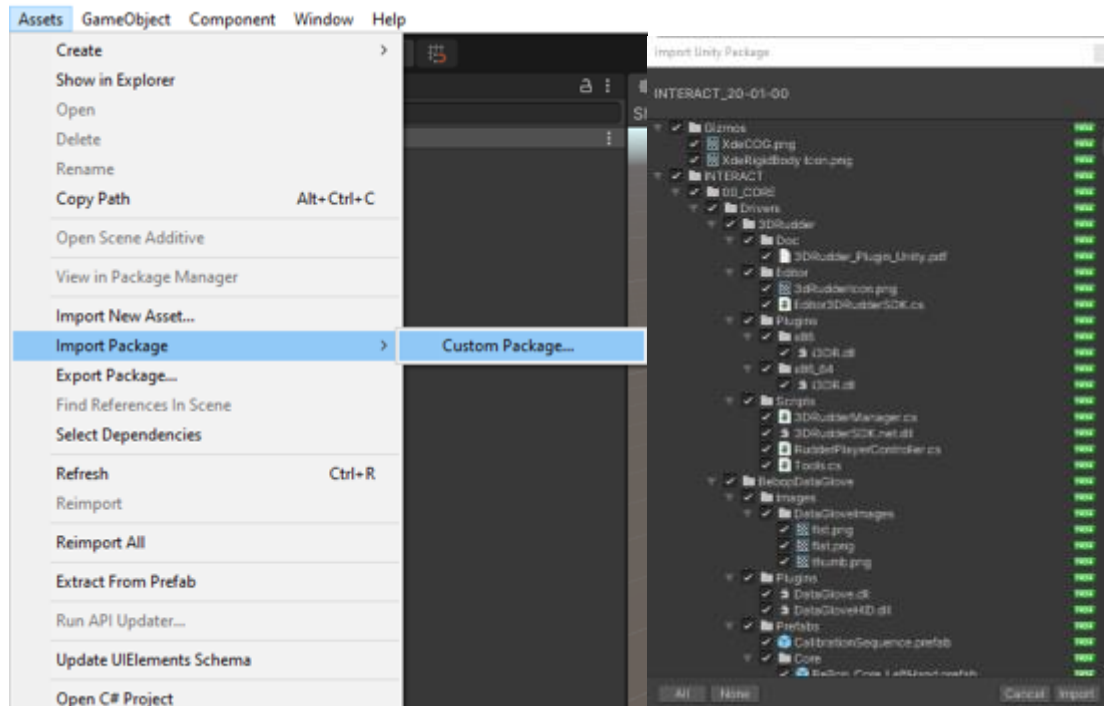
To use Enabler 1, INTERACT, the following requirements must be met:

- A VR-ready computer (appropriate graphics card: [see FAQ](#))
- Latest version of SteamVR installed (minimum recommended version 2.1)
- Unity version 2022.3 LTS (minimum and recommended) installed with a valid Unity Pro license

### 2.1.1.3. Installation

INTERACT is distributed as a Unity package. The installation of the INTERACT package in a new Unity project simply consists of importing the \*.unitypackage into the project.

You can do it by dragging and dropping the unitypackage in the Project tab, or using the menu Assets > Import Package > Custom Package... :



- Figure 2. INTERACT installation steps

### 2.1.1.4. Basic User Manual

#### Creating an INTERACT scene and configuring the XR avatar

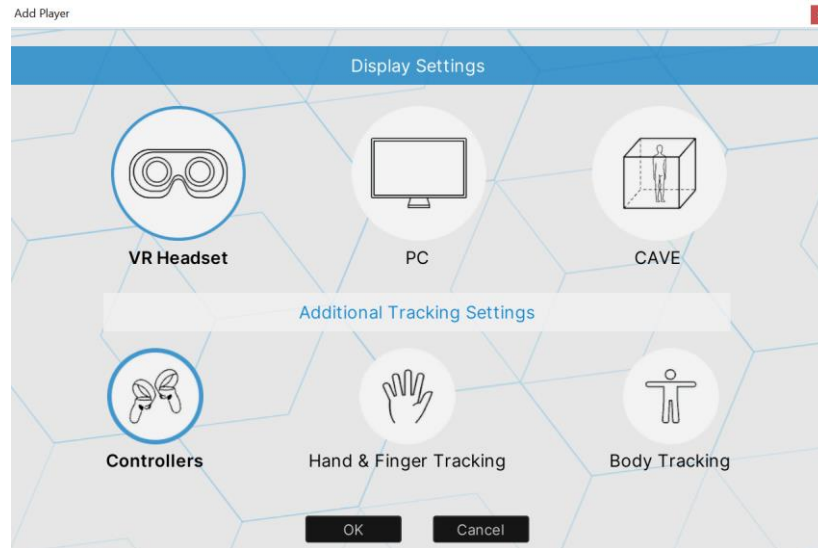
The initial steps when creating a new simulation are:

- Start from an empty Unity scene (File/New scene)
- In the INTERACT menu, click on "Create New Simulation"
- Choose between the two preconfigured environments (White Lab or Factory).



- Figure 3. INTERACT preconfigured environments

The next step is to choose the hardware devices (Display Device, Hand Tracking, Body Tracking) through which the user will interact in the XR environment.



- Figure 4. INTERACT hardware configuration and avatar creation

As a result, a default environment and a ready-to-use avatar have been created in the Unity scene and hierarchy. If a VR device is not available, the desktop player can be used in order to visualize the scene.

INTERACT allows users to switch from the working session to the VR simulation quickly. To that end:

- In the Unity toolbar, click on the Play button
- Put on a headset, a headset user should now visualize the environment at scale 1:1.



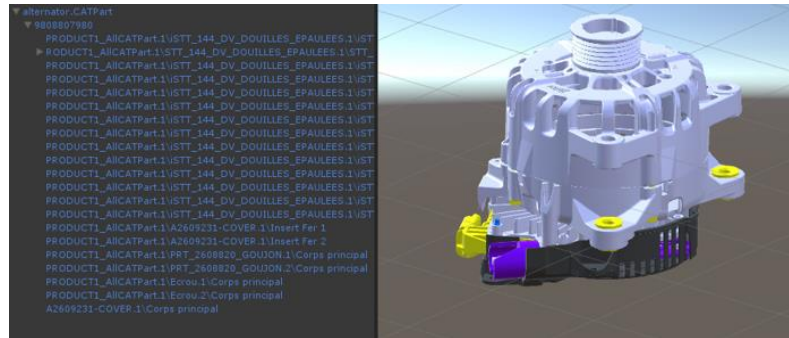
### Importing 3D objects and point clouds

The use of the Pixyz plugin is strongly recommended in order to import CAD models in Unity. PiXYZ plugin is delivered together with Unity Industrial licenses. Supported 3D formats can be found in the following link: [Supported formats | Pixyz Plugin | 2.0.3](#)

To import 3D CAD models into Unity3D:

- Click on Interact > Import > CAD model
- In the explorer, select the desired CAD file and click on Import. The CAD Import Settings window appears. From there, the default import settings can be adjusted if needed.





- Figure 5. Example of 3D object imported with PiXYZ and its hierarchy

INTERACT also supports natively point clouds, becoming increasingly popular for capturing and analyzing real-world data. Contrary to CAD data, a point cloud is a large set of 3D points that represent the surface of an object or environment. These points are usually obtained using a 3D laser scanner and can be highly dense, representing millions or even billions of points.

Importing Point Cloud is straightforward with INTERACT using the INTERACT/Import/Point Cloud menu. Supported point cloud formats are .ptx, .pts, .las, .e57.



- Figure 6. Example of point cloud imported into an INTERACT scene

## Configuring object interactions and physics

A physics engine is a software component that simulates physical phenomena, such as motion, forces, and collisions, in a virtual environment. In INTERACT, the physics engine is responsible for calculating the behavior of objects in the scene based on their physical properties and the interactions between them. Some examples of physics behavior provided by the INTERACT physics engine are:

- Collision between objects
- Part mobilities and constraints

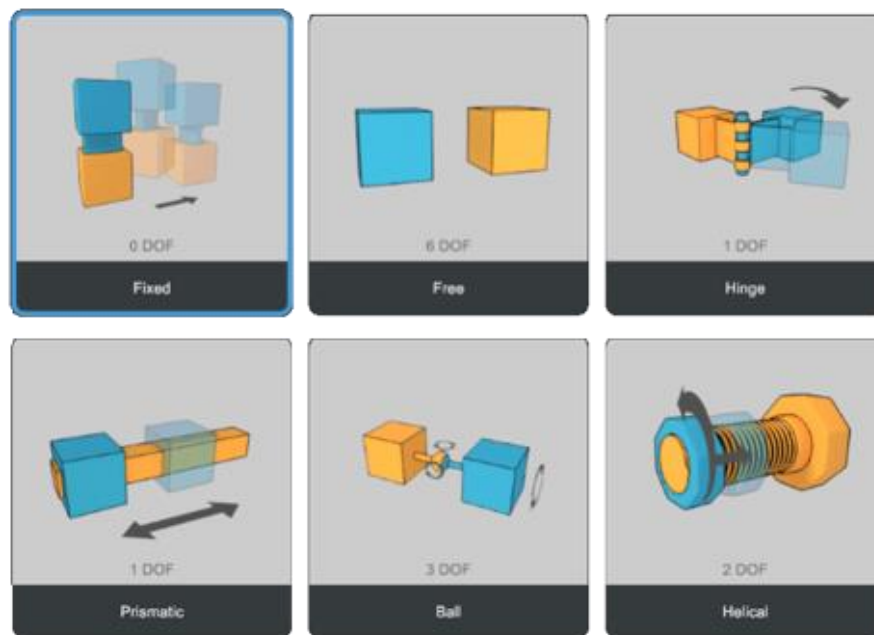


- Gravity
- Friction properties
- Part grabbing and manipulation
- Cables

*NB: INTERACT does not use Unity's default physics engine which is tailored for video games and makes assumptions and simplifications incompatible with industrial use cases. INTERACT embeds the XDE Physics engine, an interactive physics engine, featuring precise collision detection, and multi-body and beam dynamics.*

By default, when 3D objects are imported into INTERACT, they have no physical properties assigned to them. This means they cannot interact with or be affected by the laws of physics in the scene. To add physics behavior to an object, the following steps are needed:

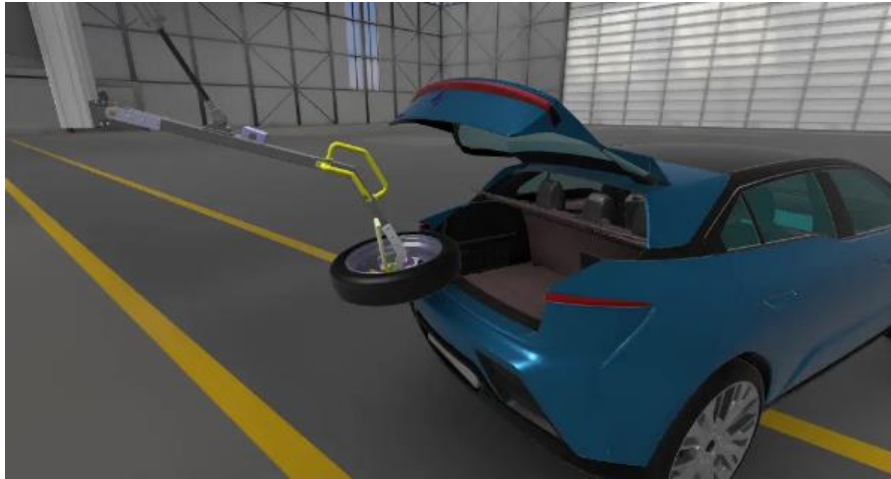
- Select the object.
- Click on the INTERACT/Physics/Physicalize object in the toolbar menu (shortcut: Shift+P).
- Choose the type of mobility or constraint to assign to the object (see joint types)



- Figure 7. Physicalizing an object and selecting kinematics properties

- Configure the physical properties of the object, such as mass, friction, motion constraints in the Inspector.

The object is then transformed into a so-called Rigid Body, recognized in the hierarchy by a gear icon. When launching the simulation, Rigid Bodies will behave according to the laws of physics and the chosen mobility type (joint). It will react to collisions as well as its children in the physical hierarchy.

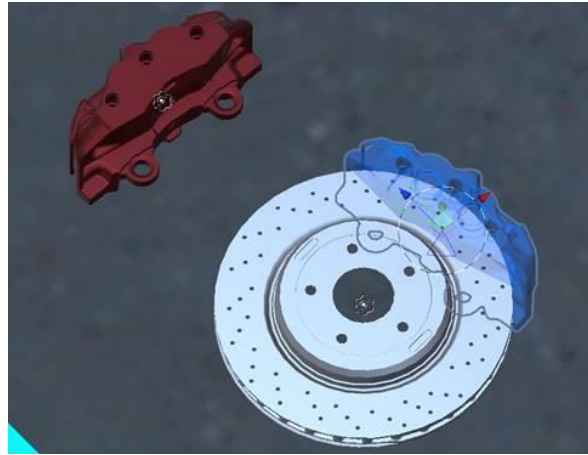


- Figure 8. Example of complex kinematic structure featuring INTERACT physics engine

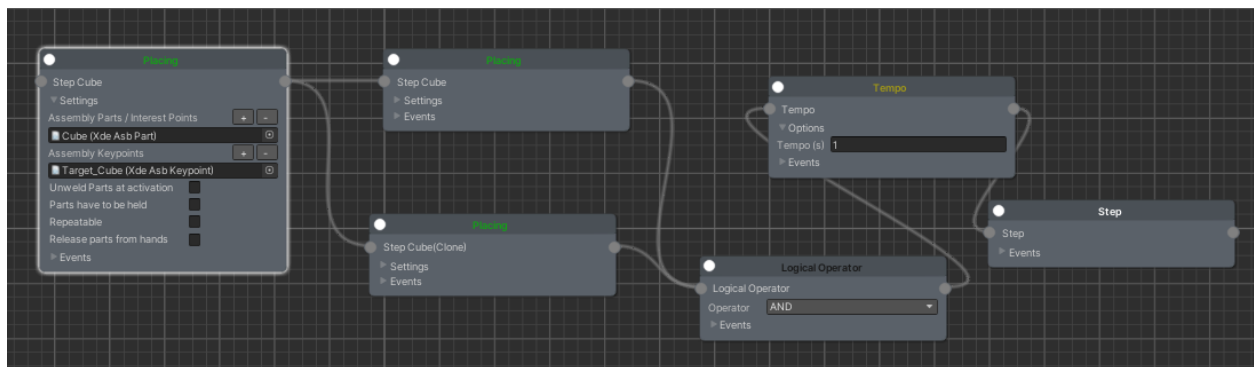
### Creating a step-by-step scenario

The next step is the scenarization, where an assembly sequence can be configured. Usually, the assembly sequences constitute the practical exercises of a training application. The user must define the order in which different parts are assembled to form a complete product. This typically involves a series of steps, in which each part is added to the product in a specific order. INTERACT helps to create such assembly sequences by visualizing the different parts and how they fit together.

In more detail, INTERACT provides the Scenario Graph to create a hierarchy of steps that create an assembly sequence. The user introduces 3D objects and indicates their connection through Placing Steps. The user can encode rules for the learning scenario to unlock the next steps. For example, the assembly of a wheel can only start if the brake disk is in place AND the bolts have been properly screwed. Several options are available to describe the assembly process in the Scenario window, including time constraints that are required before proceeding to a subsequent step, interaction with robots and actuators, among others. A scenario can also include Events, i.e., actions that are only triggered under specific conditions. For example, to unwind or activate another part when a keypoint is reached.



- Figure 9. 3D Visualization of the wheel object.



- Figure 10. Example of the scenario graph and the series of steps constituting the assembly process.

The steps to create a Scenario in INTERACT are described as follows:

- Create a scenario: INTERACT/Scenarize/Create scenario
- In the hierarchy, select a part to be used in the scenario.
- Click on this part and in the INTERACT menu: INTERACT/Scenarize/Make part grabbable.

This makes the part accessible from the scenario and grabbable in the VR.

Indicate the target of this object: click on INTERACT/Assembly/Create Part Target. The target is created and appears as a blue ghost. Place this target where the part should go. This creates a Placing Step in the scenario with the corresponding part to place and target.

The Scenario manager automatically handles the visual helpers in runtime (trajectories, ghost, instruction panel). In Simulation (when switching to the PLAY mode), the transition between steps occurs when the part-to-place reaches its target.

### 2.1.1.5. Repository and license

The INTERACT authoring tool can be found in the project's GitHub repository at:

- <https://github.com/XR2Learn/en-1-interact>

The public GitHub repository includes most of the core engine and behaviors of INTERACT, which are delivered as dll (compiled) files. Additionally, it contains many

other resources, such as texture samples and scripts, which have been made accessible, usable and editable to the users.

The repository also contains the current version of INTERACT's assets (runtime and Editor behaviors). A link to the official INTERACT delivery system is also included in the GitHub repository so that users can access an up-to-date version of INTERACT at any time.

## **LICENSE**

INTERACT is a proprietary software and distributed as closed source.

### **Available versions**

A description of the main changes in the project's versions can be found at:  
<https://light-and-shadows.com/documentation/interact/changelog/>

---

## **2.2. DISCLAIMER**

---

This document only provides a basic user manual to get started with INTERACT. To dive deeper into INTERACT's functionality, the user can read the complete documentation (<https://light-and-shadows.com/documentation/interact/>) or contact the XR2Learn consortium for one-to-one technical guidance.

---

### 3. EMOTION RECOGNITION TOOLS (ENABLERS 2-6)

---

The emotion recognition (ER) tools proposed in the XR2Learn ecosystem aim to personalize education scenarios in XR by enabling adaptive learning components that dynamically adjust to users based on their proficiency level, affective state (emotions), and challenge level of an educational scenario.

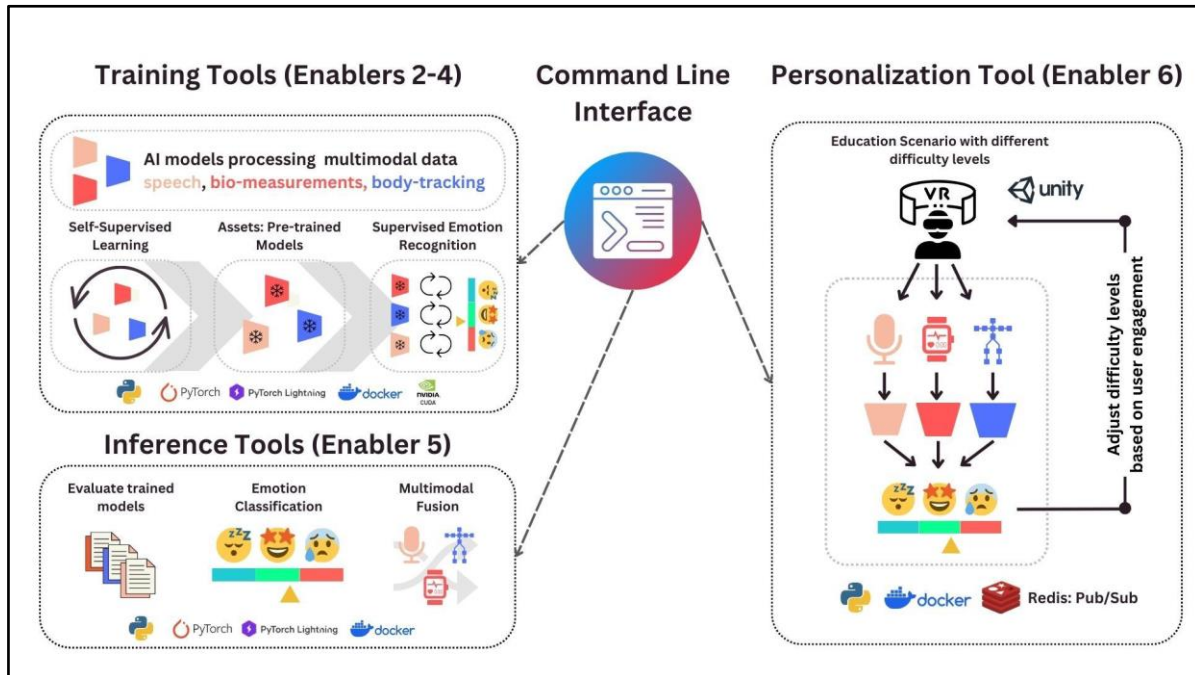
In this section, we describe the five ER enablers, first introduced in the proposal document, that were designed and implemented in the context of the XR2Learn project:

- Enabler 2: Emotion representation learning tool, including Self-Supervised Learning (SSL) pre-text task and handcrafted features extraction;
- Enabler 3: Tools for generating pre-trained emotion representations or handcrafted features, for one of the modalities exploited in the project consortium. Currently, Enablers 2-6 have been implemented and tested for speech signals available in open source data;
- Enabler 4: Tools for building emotion classifiers, in this version, for one of the modalities in the consortium;
- Enabler 5: Tools for fusing multiple modalities (decision-level fusion);
- Enabler 6: Personalization tool based on the Theory of Flow<sup>1</sup>.

Whereas each enabler clearly defines a separate functionality to be implemented, from the software engineering perspective, the enablers can be organized into four main domains based on their functionalities. A high-level diagram of the proposed components is illustrated in Figure 11. All the components within the different domains are cross-platform applications that can be hosted on a local or remote machine.

---

<sup>1</sup> Nakamura, Jeanne, and Mihaly Csikszentmihalyi. "The concept of flow." Handbook of positive psychology 89 (2002): 105.



- Figure 11. High-level overview of the five emotion recognition enablers (2-6).

A modularized software engineering architecture approach was utilized to deploy enablers to foster scalability, flexibility, and dynamic network topology. Different components can be deployed in separate machines, allowing for a heterogeneous and dynamic deployment of the components. This is essential to provide the necessary computation resources for each component. For instance, deep learning training components need heavy computational resources. With the proposed enabler's architecture, these heavy components can be deployed in a more robust computational machine. On the other hand, other components that do not require as many resources can be deployed in other machines and they can all communicate with each other.

In the proposed architecture, four domains have been implemented to cover the functionalities mentioned above the ER enablers:

#### 1. Training Tools:

The Training domain covers all enablers associated with Deep Learning model training. Specifically, under the context of the Training domain, Enabler 2 was implemented providing tools for pre-processing, handcrafted feature extraction, and learning emotion representations using Self-Supervised Learning techniques. The emotion representations are lower-dimensional descriptive features extracted from data in an automated manner by optimized Neural Networks. Moreover, Enabler 3 is delivered as a set of tools to be used to extract features (pre-trained or handcrafted) from raw data. Finally, pre-trained models can later be used to build emotion classifiers given annotated data with emotions as required by Enabler 4. All enablers are implemented using industry-standard Deep Learning frameworks (PyTorch<sup>2</sup>, PyTorch Lightning<sup>3</sup>) that support accelerated computing on GPUs. It is worth mentioning that all the enablers are delivered as standalone units for each modality via modern containerization tools (Docker<sup>4</sup>). Motivated by meeting **KPI 2.2** (Number of enablers developed,

<sup>2</sup> <https://pytorch.org/>

<sup>3</sup> <https://lightning.ai/docs/pytorch/stable/>

<sup>4</sup> <https://www.docker.com/>

contributed and used by third-parties), such a modular architecture has been proposed to facilitate and encourage consortium members, open-call participants, and open-source developers to propose and implement novel emotion recognition enablers that could be easily integrated into the proposed framework.

## 2. Inference Tools:

The Inference domain is a set of components a user will need to exploit and evaluate emotion representations and classifiers implemented within the Training domain. Most importantly, the Inference domain implements Enabler 5 by providing a fusion functionality to combine models for a flexible number of modalities previously trained in the Training domain. Moreover, additional model evaluation and emotion classification components are proposed to assess the ER models effortlessly.

## 3. Personalization Tool:

The Personalization Tool exploits the outputs of Training and Inference tools to provide personalization of XR scenarios to the users interacting with them. Utilizing the user's predicted emotions as the output of the Training and Inference domain, together with contextual information, e.g., a user and activity difficulty (challenge) levels, the personalization tool provides personalized suggestions on the recommended activity level for the user in educational XR applications.

The Personalization Tool exploits the Publisher/Subscriber messaging protocol implemented using Redis<sup>5</sup> to provide asynchronous, real-time communication between the Personalization Tool, Inference domain and an XR educational software implemented using Unity.

## 4. Command Line Interface:

Command Line Interface (CLI) is an automated interface to facilitate accessing the enablers' functionalities, in which a user can quickly and easily access enablers' use cases. CLI includes simplified installation and commands, pre-configured scripts for common use cases, and benchmarks to evaluate the end-to-end workings of the whole pipeline, working as an integration test for the system.

---

### **3.1. PRELIMINARY RESEARCH: EMOTION REPRESENTATION LEARNING AND EMOTION RECOGNITION**

---

A crucial step preceding the development of the ER tools is the preliminary research on emotion recognition that has been conducted to:

- Select Machine and Deep Learning architectures, feature extraction methods, and representation learning techniques, including Self-Supervised Learning methods, to be included in the XR2Learn emotion recognition enablers. In recent decades, emotion recognition methods have evolved from classical Machine Learning methods to sophisticated Deep Learning topologies. The state-of-the-art emotion recognition methods based on large models are more accurate and generalizable, allowing robust features extracted from raw data that can be transferred to various tasks. However, they generally require more computational resources and annotated data for training. Besides, the current

---

<sup>5</sup> <https://redis.io/docs/interact/pubsub/>



landscape of emotion recognition is heavily based on raw facial expression data, which might be challenging to obtain with commercial XR equipment. Thus, the conducted research is focused on modalities going beyond facial expressions, which (i) have been presented in the XR2Learn proposal document (namely, speech, bio-measurements, and bodily cues) and (ii) can be used in the XR context (eye tracking).

- Analyze the adequacy of existing open-source datasets for their use in the XR2Learn project for pre-training and fine-tuning Deep Learning Models. With the latest advancements in Deep Representation Learning, models trained on open-source datasets can be transferred to custom use cases. Nevertheless, various aspects should be considered, from dataset licenses to emotion elicitation protocols, before exploiting open-source data within the XR2Learn project context.
- Highlight challenges on emotion recognition in XR settings. On the one hand, the XR environment and previously unseen level of immersion provide lots of opportunities to elicit and, hence, utilize user affective states for adaptive learning. However, multiple challenges can be faced when Deep Learning-based emotion recognition should be integrated into educational VR scenarios. These challenges range from the feasibility of capturing and using certain modalities to obtaining data on the required range of affective states (e.g. Theory of Flow annotations).

Parts of the analysis conducted within this section have been presented as scientific peer-reviewed publications in international venues as follows:

- Mousavi, Seyed Muhammad Hossein, and Khaertdinov, Bulat, et al. "Emotion Recognition in Adaptive Virtual Reality Settings: Challenges and Opportunities." The 25th International Conference on Mobile Human-Computer Interaction, Workshop on Advances of Mobile and Wearable Biometrics, 2023.

---

### 3.1.1. Speech Emotion Recognition

---

Speech is a fundamental channel of expressing and interpreting emotions, encompassing semantics, paralinguistic information, and prosodic features. Achieving robust speech emotion recognition (SER) performance can be possible when suitable representations are extracted.

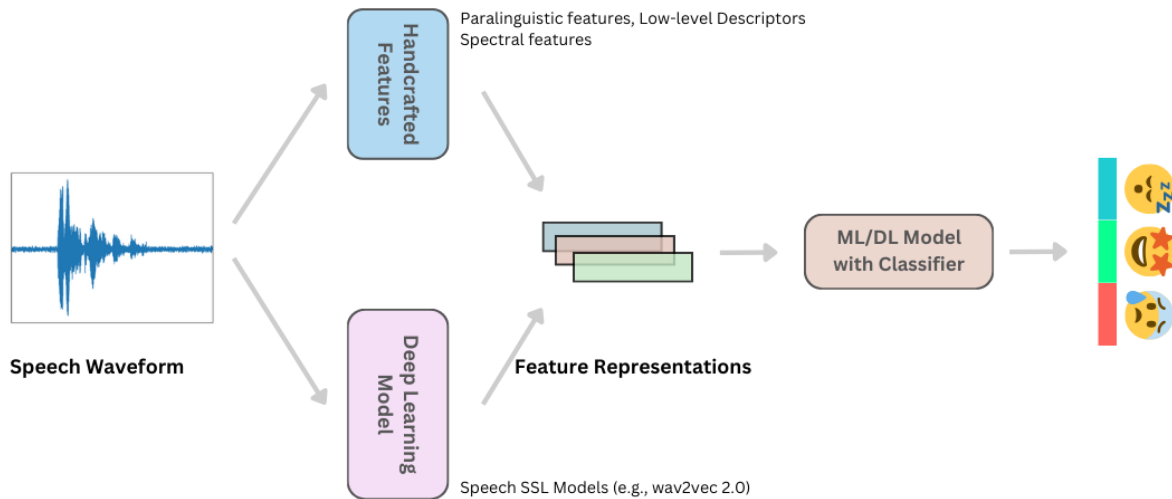
#### 3.1.1.1. Speech Representations

Approaches to SER can be categorized into two groups based on feature representation extraction, summarized in Figure 12. First, methods based on more conventional handcrafted feature extraction methods are still widely exploited in ER systems. For example, the extended Geneva Minimalistic Acoustic Parameter Set (eGeMAPS)<sup>6</sup> is a classical method of capturing acoustic and paralinguistic information without semantic richness. Besides, various spectral representations (Figure 13), such as mel-scale spectrograms or Mel-frequency Cepstral Coefficients (MFCCs), that can be obtained from speech signals present a broader outlook on speech characteristics. These handcrafted representations can be used to train relatively lightweight Neural Network models of different architectures, such as Multilayer Perceptron (MLP), and Convolutional Neural Networks (CNNs).

---

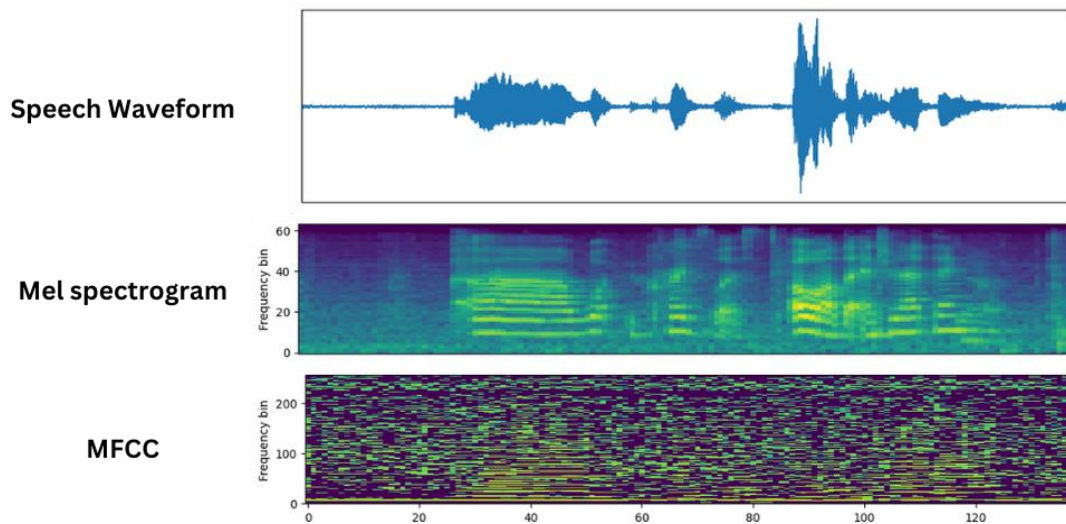
<sup>6</sup> Eyben, Florian, et al. "The Geneva minimalistic acoustic parameter set (GeMAPS) for voice research and affective computing." IEEE Transactions on Affective Computing 7.2 (2015): 190-202.





- Figure 12. Approaches to Speech Emotion Recognition.

Another family of approaches directly processing raw audio signals, also known as large speech models (e.g., wav2vec2.0<sup>7</sup> and HuBERT<sup>8</sup>), have shown superior SER performance in recent years. Nonetheless, they require much more computing power, sophisticated approaches to pre-training, based on SSL frameworks, and large amounts of speech data available for pre-training. Fortunately, open-source versions of these models are available (e.g. via the PyTorch audio framework<sup>9</sup>), that have been pre-trained on large general-purpose datasets with human speech. These representations can be re-used to build task-specific emotion recognition models with SSL and/or subsequently incorporate supervised classifiers<sup>10</sup>.



- Figure 13. Visualization of spectral representations.

<sup>7</sup> Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in neural information processing systems 33 (2020): 12449-12460.

<sup>8</sup> Hsu, Wei-Ning, et al. "Hubert: Self-supervised speech representation learning by masked prediction of hidden units." IEEE/ACM Transactions on Audio, Speech, and Language Processing 29 (2021): 3451-3460.

<sup>9</sup> <https://pytorch.org/audio/stable/pipelines.html#module-torchaudio.pipelines>

<sup>10</sup> Pepino, Leonardo, Pablo Riera, and Luciana Ferrer. "Emotion Recognition from Speech Using wav2vec 2.0 Embeddings." Proc. Interspeech 2021 (2021): 3400-3404.

### 3.1.1.2. Open-source Datasets and Experimental Setup

Before integrating models into the proposed tools described above (Section 3), we conducted a set of experiments validating the models from the literature on three widely used open-source datasets, namely IEMOCAP<sup>11</sup>, RAVDESS<sup>12</sup>, and K-EmoCon<sup>13</sup>. It is important to mention that all three datasets are available for non-commercial use and research purposes only. Besides, none of these datasets have been collected in XR environments.

The IEMOCAP dataset, recorded with 10 actors (5 males, 5 females) in 5 sessions, contains more than 7,000 audio segments with an average length of approximately 8 seconds. The dataset is typically used with 4 distinct emotions (*anger, happiness/excitement, sadness, neutral*). The distribution of classes is relatively balanced. Normally, the models are evaluated on the dataset in a leave-one-session-out cross-validation protocol, where each session is used for testing purposes once and performance metrics are averaged over all 5 folds. The RAVDESS dataset (24 actors: 12 males, 12 females) contains 8 emotions (*neutral, calm, happy, sad, angry, fearful, disgust, surprised*) with a dataset size of about 1400 samples. For this dataset, we employ random hold-out sets (80% train / 10% validation / 10% test) based on the actors. The emotions presented in the IEMOCAP and RAVDESS datasets were played by actors in scripted and improvised discussions. These datasets also contain facial expression modality in addition to speech.

The K-EmoCon dataset consists of untrimmed dialogues, including data from 32 participants (20 male, 12 female) that can be processed into approximately 1,000 samples of 10 seconds in length. The dataset contains different annotation types, including (i) a two-dimensional affective model with arousal and valence scores between 1 (very low) and 5 (very high); (ii) categorical emotional classes (*cheerful, happy, angry, nervous, sad*) each scored between 1 (very low) and 5 (very high); and (iii) Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP)<sup>14</sup> where a single class from engagement related categories (boredom, confusion, delight, engaged concentration, frustration, surprise, or none) is selected. The latter annotation system, BROMP, is closely related to education scenarios and derived from the Theory of Flow, the model intended to be used in the XR2Learn project. However, the obtained BROMP annotations are severely imbalanced, as shown in Figure 14, which brings an additional challenge for models to classify these affective states. Apart from audio signals, this dataset also contains visual data and bio-measurement signals. To elicit the emotions above, the subjects (non-native English speakers) were asked to participate in debates on sensitive political topics in English. In turn, the emotion annotations were obtained via self and external assessment. For this dataset, the typical procedure is leave-one-subject-out, where samples from each of the 32 subjects are subsequently used for model evaluation (test set).

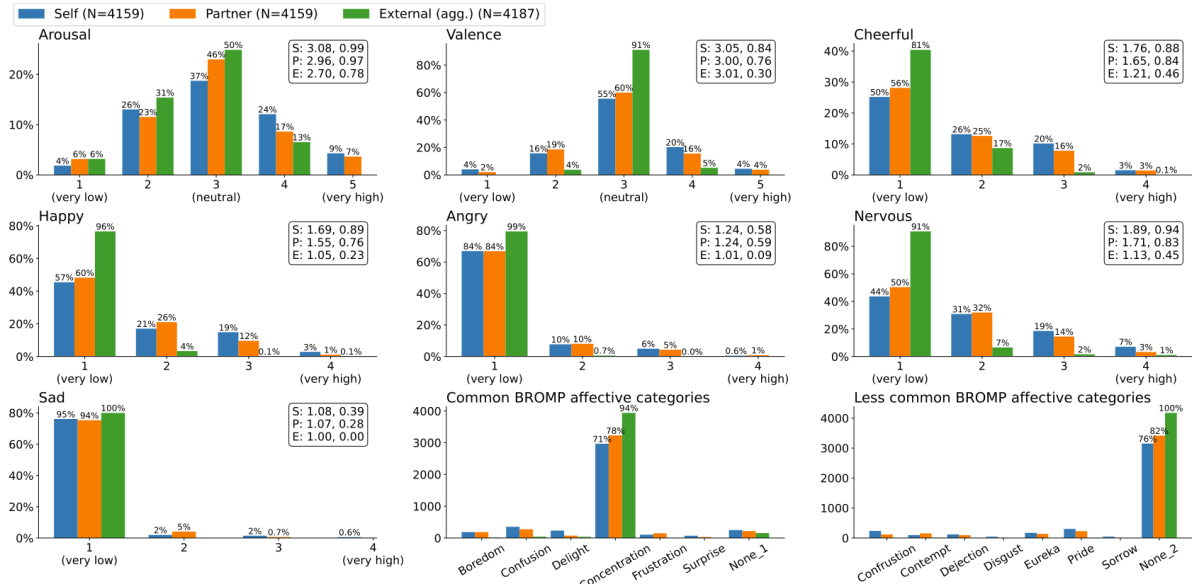
---

<sup>11</sup> Busso, Carlos, et al. "IEMOCAP: Interactive emotional dyadic motion capture database." *Language resources and evaluation* 42 (2008): 335-359.

<sup>12</sup> Livingstone, Steven R., and Frank A. Russo. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English." *PloS one* 13.5 (2018): e0196391.

<sup>13</sup> Park, Cheul Young, et al. "K-EmoCon, a multimodal sensor dataset for continuous emotion recognition in naturalistic conversations." *Scientific Data* 7.1 (2020): 293.

<sup>14</sup> Ocumpaugh, Jaclyn. "Baker Rodrigo Ocumpaugh monitoring protocol (BROMP) 2.0 technical and training manual." New York, NY and Manila, Philippines: Teachers College, Columbia University and Ateneo Laboratory for the Learning Sciences 60 (2015).



- Figure 14. Distribution of annotations (adapted from the K-EmoCon dataset paper).

In our preliminary research for Speech Emotion Recognition, we evaluated various Neural Network (Deep Learning) architectures on IEMOCAP, RAVDESS, and K-EmoCon using speech representations described in the previous section. Specifically, the input representations of speech and implemented models used in the experiments are described in Table 1. In particular, we implemented three methods based on handcrafted features, including MLP based on eGeMAPS low level descriptors and one-dimensional CNNs processing spectral representations of speech. Finally, we also investigated two SSL architectures processing raw speech data. These models are two versions, base and large, of the widely used wav2vec 2.0 neural network. In our experiments, we used models that have been already pre-trained on large speech dataset, namely LibriSpeech<sup>15</sup>, and available as open-source via torchaudio package.

<sup>15</sup> Panayotov, Vassil, et al. "Librispeech: an asr corpus based on public domain audio books." 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2015.

- Table 1. Speech Input Data and Deep Learning Models

Model No.	Input Format	Deep Learning Model / Approach to Learning	Description
1	eGeMAPs (Functional)	MLP	The eGeMAPs functional features correspond to 88 features related to acoustics computed for the whole input audio stream. These values are used as inputs for a fully connected Neural Network, or MLP, with 2 layers.
2	Spectral (mel-scale spectrograms)	CNN	One-dimensional Convolutional Neural Network with three layers and a Linear Classifier applied to mel-scale spectrograms extracted from raw speech.
3	Spectral (MFCC)	CNN	One-dimensional Convolutional Neural Network with three layers and a Linear Classifier applied to MFCCs extracted from raw speech.
4	Raw Audio	wav2vec2.0 base (SSL) + pointwise CNN + linear classifier	Apply the open-source base wav2vec 2.0 speech model pre-trained on LibriSpeech dataset using SSL to pre-processed speech. The obtained features are then passed to a pointwise CNN with a linear classifier.
5	Raw Audio	wav2vec2.0 large (SSL) + pointwise CNN + linear classifier	Apply the open-source large wav2vec 2.0 speech model pre-trained on the LibriSpeech dataset using SSL to pre-processed speech. The obtained features are then passed to a pointwise CNN a linear classifier.

### 3.1.1.3. Evaluations

We evaluated the proposed models on three datasets that we described earlier: IEMOCAP, RAVDESS and K-EmoCon. Table 2 presents the quantitative results of the conducted experiments. The metric used to assess the performance of models is the average (macro) F1-score (%).

- Table 2. F1-scores (%) for the speech emotion recognition task.

Model No.	IEMOCAP	RAVDESS	K-EmoCon (arousal-valence) <sup>16</sup>	K-EmoCon (BROMP)
1	48.1	29.5	38.6	11.7
2	52.09	30.7	<b>41.61</b>	11.6
3	50.56	40.9	31.65	11.7
4	60.65	70.4	34.2	11.7
5	<b>64.46</b>	<b>72.4</b>	35.87	11.7

As can be observed in Table 2, the largest model (model no.5, wav2vec 2.0 large) significantly outperforms all the other topologies on IEMOCAP and RAVDESS datasets, two datasets widely used in research on speech emotion recognition. Nevertheless, the

<sup>16</sup> The performance is averaged over two-class (high, low) arousal and valence classification scores.

performance on the K-EmoCon dataset (arousal-valence scales) is better for the model based on mel-scale spectrograms. Additionally, it is evident that all models perform remarkably low on BROMP annotations. Most probably, this issue arises due to the extremely unbalanced data.

While the emotion recognition rate is a key factor for selecting an architecture, in real-world inference settings, it is also important to consider other aspects related to model efficiency. While large speech models can significantly outperform other topologies, they are associated with higher costs and requirements for fine-tuning and higher latency during inference. In Table 3, the metrics related to computation costs and latency of the evaluated models are presented.

- Table 3. Computation efficiency of the Deep Learning models. Training and inference time are presented for the RAVDESS dataset. The training time is reported for the Nvidia Titan V GPU. The inference time is reported for the CPU (Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz).

Model No.	Number of parameters (millions)	Training time per epoch <sup>17</sup> (seconds)	Average inference time per instance (seconds)
1	0.03	0.35	0.0042
2	0.06	0.5	0.0051
3	0.34	0.4	0.0047
4	94.5 (without pre-trained part: 0.13)	14.8	0.3
5	315 (without pre-trained part: 0.17)	30.9	0.575

As anticipated, large speech models demand considerable computational resources. They present higher requirements to GPU resulting in training (or fine-tuning) times that are several orders of magnitude longer. They also exhibit slower inference compared to the models based on handcrafted features.

### 3.1.1.4. Challenges and Limitations

Based on the analysis of the literature and conducted experiments, the following challenges and limitations have been highlighted when operating with speech modality:

- While speech contains rich information about human emotions, to the best of our knowledge, there are no datasets with speech collected in natural education settings. Furthermore, most of the datasets containing speech present sets of emotions that are not related to user engagement. In the conducted experiments, one dataset with education-related BROMP annotations have been used. Nevertheless, it has been found that regardless of applied methods, the emotion recognition performance is very low. This can be explained by the quality of collected data and annotations, i.e. emotion elicitation, annotation

<sup>17</sup> RAVDESS training set contains approximately 1200 samples. Thus, an epoch is one iteration of training through 1200 instances.

procedures, and class-imbalance. Thus, this challenge can be addressed by a dedicated data collection pilot, or methodology to map standard models of emotion representation to education-related annotations (e.g. theory of flow).

- The large speech models typically show remarkable performance compared to models based on handcrafted features. Nevertheless, fine-tuning such models is significantly more computationally expensive compared to other approaches. Besides, their latency is considerably higher compared to the methods based on handcrafted features. If required, the latency of the models can be lowered by using contemporary model compression techniques, such as quantization and knowledge distillation. Also, the deployment architecture for training and inference components can be modularized to deploy different components in distinct machines according to how much computational resources the component requires.

---

### 3.1.2. Emotion Recognition through Bio-measurements

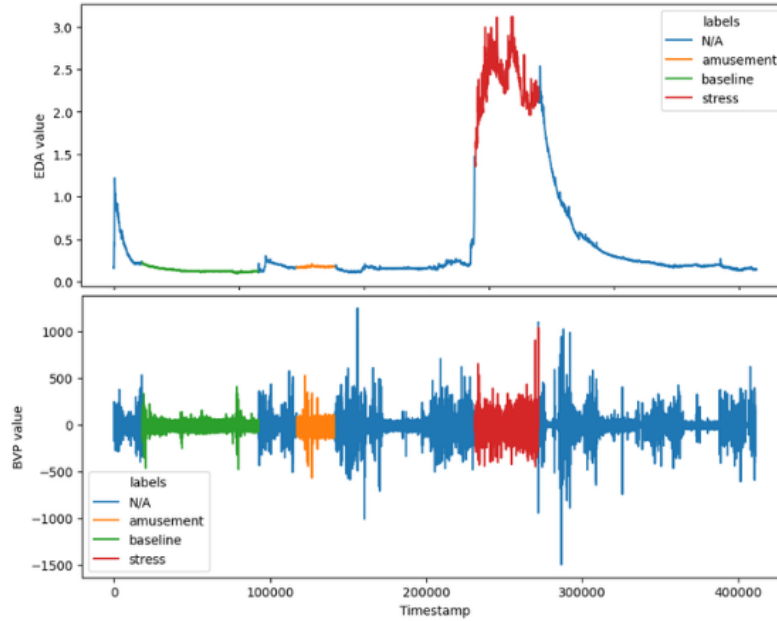
---

Perceived emotions trigger physiological responses in the body, such as changes in heart rate, skin conductance, and respiration rate. Moreover, such changes can be measured by wearable physiological sensors that have become more accessible in recent years. Nevertheless, collecting some physiological responses can require expensive intrusive equipment and a complex environment for accurate data collection, which is not always feasible and can be challenging to integrate with commercial VR educational scenarios.

Electrodermal activity (EDA) signals, also known as Galvanic Skin Response (GSR), are recorded by sensors that measure skin conductance changes, which can indicate emotional arousal levels. Heart rate variability is another source of information that can be used to reflect the changes in the affective state. In particular, an Electrocardiogram (ECG) is a robust heart rate monitor that requires connecting multiple electrodes to subjects' chests, which is hard to achieve beyond the laboratory environment. A less intrusive and more lightweight measurement of heart rate variability is Blood Volume Pulse (BVP).

Skin temperature (SKT) is also used for affect recognition, although several factors could influence it. It typically is a weaker signal compared to EDA and BVP. Wearable devices, including commercial-grade and research-grade options such as smart watches, bracelets, and rings, can record EDA, BVP, and SKT data through integrated electrodes. The EDA, BVP, and SKT data are commonly used to predict emotional arousal levels, signaling the intensity of the experienced affective states, and detecting stress and excitement. We visualize examples of data recordings corresponding to EDA and BVP signal obtained from the WESAD dataset in Figure 15.





- Figure 15. Signal recordings from EDA and BVP sensors collected from one participant in the WESAD dataset.

Unlike the other physiological sensors, Electroencephalography (EEG) signals can be used to evaluate two-dimensional affect recognition, i.e., both arousal and valence levels. However, EEG devices require complex installation and, ideally, laboratory settings to collect accurate data.

### 3.1.2.1. Learning Representations from Bio-measurements

In recent years, Self-Supervised Learning has been applied to emotion recognition to learn unimodal features from different modalities. Experiments conducted on various modalities, such as ECG data<sup>18</sup>, EDA, BVP, and SKT data<sup>19</sup>, show that unsupervised representation learning is a promising direction that allows Deep Learning models to learn robust unimodal representations from unlabeled data. Nevertheless, unlike in speech emotion recognition, there are no publicly available pre-trained models for these data modalities. Thus, the SSL frameworks should be tailored individually for bio-measurement data.

### 3.1.2.2. Open-source Datasets and Experimental Setup

In our preliminary experiments, we used WESAD<sup>20</sup> and K-EmoCon datasets (introduced earlier for speech) that include bio-measurement data obtained via various wearable sensors. In particular, we focus on less intrusive sensors presented in these datasets that were installed in wristbands. The WESAD dataset is collected from 15 subjects (12 males and 3 females), whereas the annotations are made based on the stimuli the subjects have received. Specifically, a three-class version of the dataset contains *neutral*, *amusement*, and *stress* classes. The elicitation materials were the Trier Social

<sup>18</sup> Sarkar, Pritam, and Ali Etemad. "Self-supervised ECG representation learning for emotion recognition." IEEE Transactions on Affective Computing 13.3 (2020): 1541-1554.

<sup>19</sup> Dissanayake, Vipula, et al. "Sigrep: Toward robust wearable emotion recognition with contrastive representation learning." IEEE Access 10 (2022): 18105-18120.

<sup>20</sup> Schmidt, Philip, et al. "Introducing wesad, a multimodal dataset for wearable stress and affect detection." Proceedings of the 20th ACM international conference on multimodal interaction. 2018.

Stress Test<sup>21</sup> for the stress state, and video clips for the amusement state. The K-EmoCon dataset has been previously discussed in the context of audio modality. Both datasets used Empatica E4<sup>22</sup> to collect EDA, SKT, and BVP data. As stated on the Empatica website, the E4 wristband is no longer available for purchase.

In our experimentation, we have implemented 3 models, described in Table 4, for bio-measurement data (EDA, BVP, SKT). All models are based on Convolutional Neural Networks, which are commonly used to process multivariate time-series data. Nevertheless, the difference between the proposed models resides in the methodology employed for their training. In particular, the first model is a CNN train from scratch in supervised settings, whereas the remaining models are first pre-trained using state-of-the-art Self-Supervised Learning frameworks (SimCLR and VICReg) that we adapted to the problem of emotion recognition.

- Table 4. Deep Learning Models for bio-measurement data.

Model No.	Input Format	Deep Learning Model / Approach to Learning	Description
1	Raw signals	Supervised CNN	Convolution Neural Network (CNN) with 3 layers and a fully connected classification layer. Applied to raw multi-channel bio-measurement signals. Each channel corresponds to a certain device (EDA, BVP, SKT). The signals are re-sampled to the same frequency, normalised to zero mean and unit variance per channel and segmented into 10-second time intervals with a 5-second overlap.
2	Raw signals	SimCLR pre-training for CNN	CNN architecture (same as in model 1), pre-trained with the SimCLR <sup>23</sup> contrastive SSL framework.
3	Raw signals	VICReg pre-training for CNN	CNN architecture (same as in model 1), pre-trained with the VICReg <sup>24</sup> SSL framework.

### 3.1.2.3. Evaluations

The described models have been tested on two datasets and two protocols per dataset. Specifically, we used 3-class (stress/neutral/amused) and 2-class (stress/no-stress) protocols from WESAD dataset to train out models and K-EmoCon dataset previously used for SER experiments. Table 5 presents the quantitative results of the conducted experiments. The metric used to assess the performance of models is the average (macro) F1-score.

<sup>21</sup> Kirschbaum, Clemens, Karl-Martin Pirke, and Dirk H. Hellhammer. "The 'Trier Social Stress Test'—a tool for investigating psychobiological stress responses in a laboratory setting." *Neuropsychobiology* 28.1-2 (1993): 76-81.

<sup>22</sup> <https://www.empatica.com/store/e4-wristband/>

<sup>23</sup> Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.

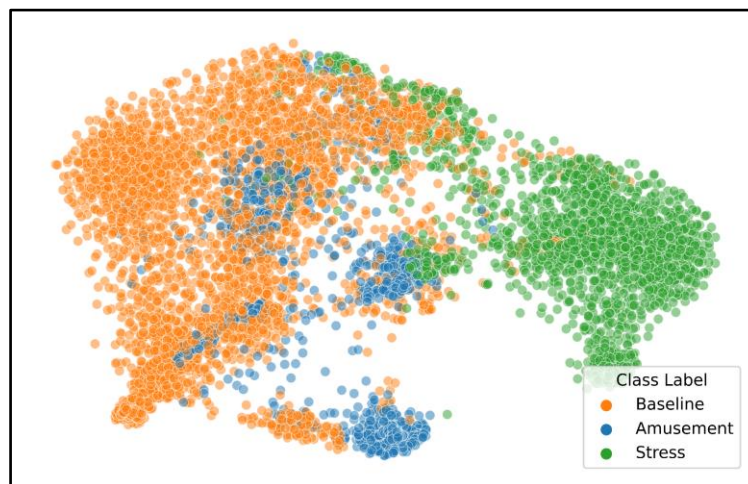
<sup>24</sup> Bardes, Adrien, Jean Ponce, and Yann Lecun. "VICReg: Variance-Invariance-Covariance Regularization For Self-Supervised Learning." *ICLR 2022-International Conference on Learning Representations*. 2022.



- Table 5. F1-scores (%) for Emotion Recognition using bio-measurement signals.

Model No	WESAD (3-class)	WESAD (stress / no-stress)	K-EmoCon (arousal-valence) <sup>25</sup>	K-EmoCon (BROMP)
1	<b>69.1</b>	89.4	41.6	11.9
2	67.7	<b>91.0</b>	<b>43.1</b>	11.8
3	67.4	90.1	39.98	11.7

As can be seen, all implemented models show an impressive performance for distinguishing stress episodes in subjects on the WESAD dataset. Nevertheless, the performance drops significantly when a third amusement class is introduced. This could have happened due to a number of reasons. First, the authors of the WESAD dataset used stimuli for emotion elicitation and annotation. For example, to provoke emotion, an amusing video has been shown to the participants and the authors assumed that the subjects felt amused while watching the video. Another reason might be related to the fact that bio-measurements reflect changes in arousal level, whereas they are not strong markers to estimate valence levels. To further illustrate this issue we visualize the representations that model 1 learnt for input data in two-dimensional space by using the t-SNE<sup>26</sup> dimensionality reduction technique. The t-SNE projections are shown in Figure 16. As can be seen, the models struggle to distinguish between amusement and neutral (baseline) classes.



- Figure 16. Representations of the instances in the WESAD dataset projected onto two-dimensional space with t-SNE. Each point corresponds to a separate bio-measurement recording.

For the K-EmoCon dataset, the proposed models show the performance comparable to performance of speech-based models based on spectrograms. For the education related BROMP annotations, the proposed models also show significantly lower performance, due to the issue with extreme data imbalance.

<sup>25</sup> The performance is averaged over two-class (high, low) arousal and valence classification scores.

<sup>26</sup> Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.11 (2008).

In Table 6, the metrics related to computation costs and latency of the evaluated models are presented. As can be seen, the pre-training does not affect the inference time. However, more resources are needed for the pre-training stage itself.

- Table 6. Computational efficiency of the Deep Learning models for emotion recognition using bio-measurements. Training and inference time are presented for the WESAD dataset. For SSL models (2, 3), *pt* and *ft* refer to pre-training and fine-tuning stages. Inference times during pre-training are not applicable (N/A), as the inference is performed with the fine-tuned model. The training time is reported for Nvidia Quadro RTX-5000 GPU. The inference time is measured on CPU.

Model No.	Number of parameters for training (millions)	Training time per epoch (seconds)	Inference time per example (seconds)
<b>1</b>	0.316	2	0.014
<b>2: pt</b>	1.8	10	N/A
<b>2: ft</b>	0.316	2	0.014
<b>3: pt</b>	1.8	8	N/A
<b>3: ft</b>	0.316	2	0.014

### 3.1.2.4. Challenges and Limitations

The following challenges and limitations were identified in the research phase with bio-measurement data:

- One of the key technical challenges in developing emotion recognition systems using bio-measurements is due to the wide variety of devices available on the market. Each device generates input signals with unique characteristics and formats, which makes it difficult to develop a universal system. Unlike audio waveforms, bio-measurement signals are recorded using various physiological sensors with different frequencies and sensitivities, which are dependent on the device used for data collection. Furthermore, there is no industry-standard wearable device, which means that the developed components should be flexible enough to allow for the quick and straightforward development of add-ons to process signals from different devices.
- Most of the datasets present sets of emotions that are not related to user engagement. Whereas these datasets can be exploited to pre-train Deep Learning models, annotated data needs to be collected for model fine-tuning using emotion model engagement, such as the Theory of Flow.
- Machine and Deep Learning models' performance using bio-measurement signals might not be as accurate compared to the performance of models based on facial expressions and speech. This is due to the fact that such physiological responses, as EDA and BVP, have been suggested as a valuable physiological indicator mainly for depicting emotional arousal through various methods of emotion elicitation<sup>27</sup> and might not reflect the valence dimension of emotions.

<sup>27</sup> Picard, Rosalind W., Szymon Fedor, and Yadid Ayzenberg. "Multiple arousal theory and daily-life electrodermal activity asymmetry." *Emotion review* 8.1 (2016): 62-75.

In other words, these signals are not considered as a strong marker for identifying positivity or negativity of certain emotions, but only reflecting their intensity.

- Emotion elicitation and data annotation are challenging aspects that should be carefully considered when building an emotion recognition system. As shown in the experiments with the WESAD dataset, this might have a significant impact on the quality of data and annotations. When developing a data collection tool, it is very important to carefully plan what emotions have to be elicited and how they can be annotated (e.g. self-annotation, external annotation) to reduce the number of incorrectly labeled instances.

---

### 3.1.3. Emotion Recognition using Body Tracking

---

Emotion recognition through body tracking and body motion, a pivotal area in affective computing, employs sophisticated algorithms and sensor technologies to decode non-verbal emotional cues. Algorithms such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), eXtreme Gradient Boosting (XGBoost), and Long Short-Term Memory (LSTM) play a crucial role in processing data captured by motion capture sensors like Kinect<sup>28</sup>, Vicon<sup>29</sup>, and Intel RealSense<sup>30</sup>. These sensors are adept at recording detailed body movements for computational analysis. For instance, the study by Kleinsmith et al.<sup>31</sup> exemplifies the use of body posture as a reliable indicator of emotional states. This technology's applications are vast, extending from virtual reality user experience enhancements to mental health care improvements, as explored in the work by Karg et al.<sup>32</sup>. Joint Angles, Joint Displacement, Joint Velocity and Acceleration, Spatial Position of Joints, Symmetry and Asymmetry in Movements, Sequential Patterns of Joint Movements, and Range of Motion are typically used for feature extraction purposes in emotion recognition from body motion<sup>33 34</sup>. The continuous evolution in this domain is set to significantly transform human-computer interaction, making it more intuitive and empathetic.

#### 3.1.3.1. Features and Classifiers

Fusing statistical features extracted from the spatial domain, Fourier transform, wavelet transform, and Hilbert transform could provide acceptable results. Features such as mean, median, standard deviation, variance, skewness, kurtosis, min value, max value, sum value, elements, peaks, sum of peaks, and percentiles are mentionable

---

<sup>28</sup> <https://learn.microsoft.com/en-us/azure/kinect-dk/windows-comparison>

<sup>29</sup> <https://www.vicon.com/>

<sup>30</sup> <https://www.intelrealsense.com/>

<sup>31</sup> Kleinsmith, Andrea, P. Ravindra De Silva, and Nadia Bianchi-Berthouze. "Recognizing emotion from postures: Cross-cultural differences in user modeling." User Modeling 2005: 10<sup>th</sup> International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005. Proceedings 10. Springer Berlin Heidelberg, 2005.

<sup>32</sup> Karg, Michelle, et al. "Body movements for affective expression: A survey of automatic recognition and generation." IEEE Transactions on Affective Computing 4.4 (2013): 341-359.

<sup>33</sup> Ahmed, Ferdous, ASM Hossain Bari, and Marina L. Gavrilova. "Emotion recognition from body movement." IEEE Access 8 (2019): 11761-11781.

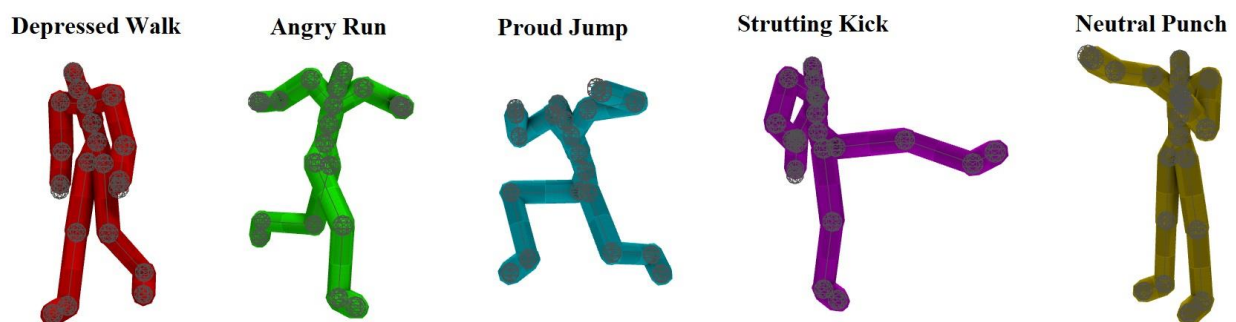
<sup>34</sup> Roether, Claire Louise. The Expression of Emotions through Full-body Movement: Features and Asymmetry. Diss. 2010.

as effective. For this type of feature extraction, LSTM<sup>35</sup>, 1-D CNN<sup>36</sup>, Gradient Boosting, XGBoost, and Decision Tree classifiers return satisfactory results. It has been reported that additionally applying feature selection by Principal Component Analysis (PCA), Lasso Regularization, Variance Threshold, F-test, Tree-based Feature Selection, Neighborhood Components Analysis (NCA), and feature importance by eXplainable Artificial Intelligence (XAI) algorithms such as SHapley Additive exPlanations (SHAP) could remove outliers and improve the accuracy. However, the XAI approaches are more complex and costly.

### 3.1.3.2. Open-source Datasets

It is important to mention that the datasets available in open-source contain a more extensive set of body keypoints, whereas the commercial XR environment allows to track a limited set of markers. Nevertheless, for the proof of concept, we will use the full set of keypoints in the subsequent experiments.

In our preliminary experiments, we used three motion capture datasets, starting with the Xia<sup>37</sup> dataset. The Xia dataset comprises approximately 11 minutes of motion data, equivalent to 79,829 frames (572 samples) in BioVision Hierarchy (BVH<sup>38</sup>) format and 38 joints. The motion data was captured using a Vicon optical motion capture system, employing eighteen 120 Hz cameras. The dataset includes a wide range of human actions such as walking, running, jumping, kicking, punching, and transitions between these behaviors. Each action is represented in eight distinctive styles: neutral, proud, angry, depressed, strutting, childlike, old, and sexy. All joint angles in the dataset, except for the root joint, are converted to Cartesian parameters using the exponential map parameterization. This ensures proper manipulation of the joint-angle quantities essential for style translation. Figure 17 depicts some samples generated by us from the Xia dataset in various actions and emotions using BVHView<sup>39</sup> software in Windows OS.



<sup>35</sup> da Silva, Rogério E., Jan Ondrej, and Aljosa Smolic. "Using LSTM for Automatic Classification of Human Motion Capture Data." VISIGRAPP (1: GRAPP). 2019.

<sup>36</sup> Li, Hai, Hwa Jen Yap, and Selina Khoo. "Motion classification and features recognition of a traditional Chinese sport (Baduanjin) using sampled-based methods." Applied Sciences 11.16 (2021): 7630.

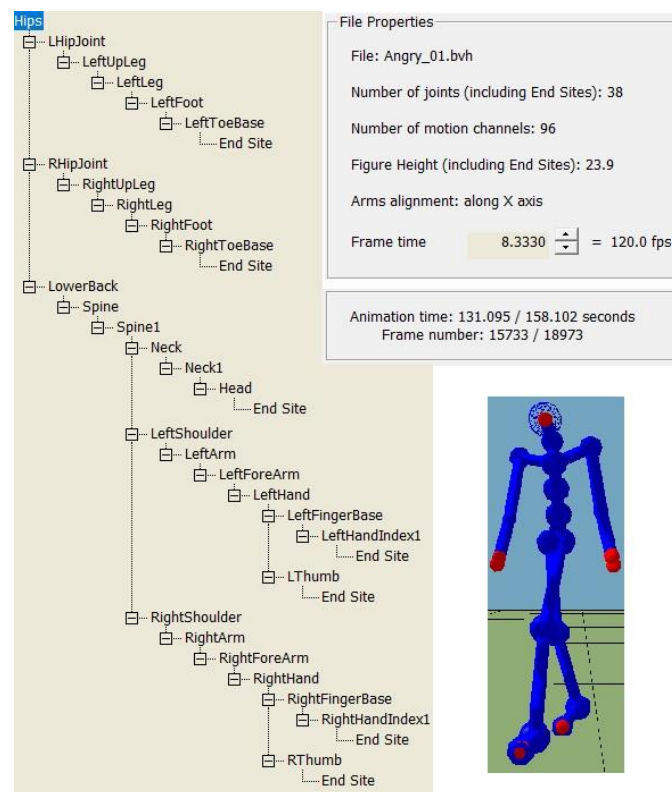
<sup>37</sup> Xia, Shihong, et al. "Realtime style transfer for unlabeled heterogeneous human motion." ACM Transactions on Graphics (TOG) 34.4 (2015): 1-10.

<sup>38</sup> <https://www.cs.cityu.edu.hk/~howard/Teaching/CS4185-5185-2007-SemA/Group12/BVH.html>

<sup>39</sup> <https://github.com/orangeduck/BVHView>

- Figure 17. Generated visual motion caption samples from Xia dataset in different actions and emotions

Edinburgh University published multiple motion capture datasets<sup>40</sup> in different actions and emotions namely, edinlocomotion, edinkinect, edinxsens, edinmisc, edinpunching, and edinterrain which here, we use edinlocomotion<sup>41</sup>. This is a database containing long clips of locomotion data, including running, walking, jogging, and various sidestepping motions. It contains around 20 minutes of raw data and is not segmented into individual strides. Each data point in the dataset is enhanced with additional scalar control signals, including the turning speed, forward velocity, and sideways velocity of the body. These signals provide more context to the motion data and allow for more detailed analysis and application. Every frame of a sequence in the dataset represents 21 joint positions in the local Cartesian space, with the origin at the hip of the character. However, for our experiment, we used the 38-joint version of it (47 samples) which was retargeted by Holden (2016)<sup>42</sup>. Figure 18 illustrates the body hierarchy of one of the walking samples from this dataset in the BVH Hacker tool<sup>43</sup>.



<sup>40</sup> de la Cruz, Vladimir. Human motion convolutional autoencoders using different rotation representations. Diss. Concordia University, 2019.

<sup>41</sup> Komura, Taku, et al. "A recurrent variational autoencoder for human motion synthesis." The 28th British Machine Vision Conference. 2017.

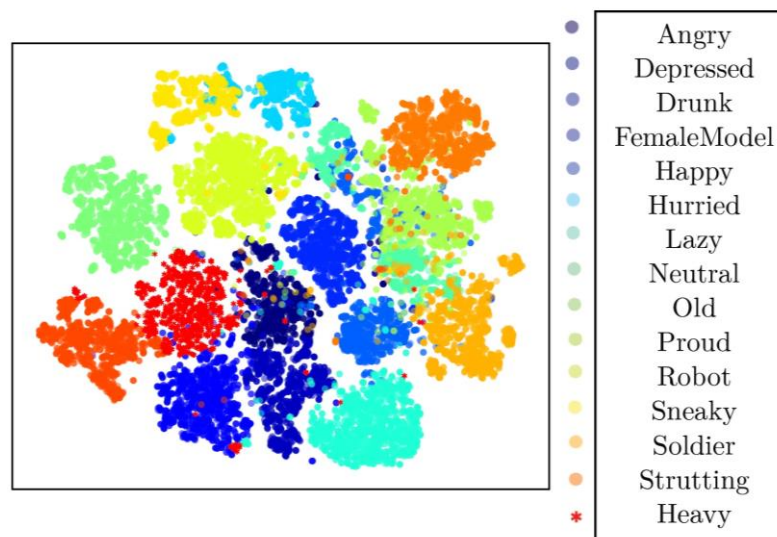
<sup>42</sup> Holden, Daniel, Jun Saito, and Taku Komura. "A deep learning framework for character motion synthesis and editing." ACM Transactions on Graphics (TOG) 35.4 (2016): 1-11.

<sup>43</sup> <https://www.bvhacker.com/>



- Figure 18. A walking body hierarchy sample from edinlocomotion dataset in
- BVHHacker Tool

The BFA dataset was introduced by Aberman (2020)<sup>44</sup> for two purposes; retargeting and motion style transfer. Retargeting means changing the number of positions of joints from one character to another and motion style transfer means transferring the style of different subjects' actions to each other. For instance, to transfer "angry walking" style from one subject to another person's body motion. In this work, the motion style transfer dataset was used. This dataset consisted of 33 long motion capture clips in 16 styles and emotions. Figure 19 represents the t-SNE plot of these styles and emotions (adopted from their paper). Table 7 represents our experimental results on the mentioned three datasets using different extract features. Five other body motion datasets are considered for the future experiments which are MPI<sup>45</sup>, CMU<sup>46</sup>, 100 Style Dataset<sup>47</sup>, KDAEE<sup>48</sup>, and Bandai-Namco Research Motion dataset<sup>49</sup>.



- Figure 19. t-SNE plot of BFA dataset

<sup>44</sup> Aberman, Kfir, et al. "Unpaired motion style transfer from video to animation." ACM Transactions on Graphics (TOG) 39.4 (2020): 64-1.

<sup>45</sup> Volkova, Ekaterina, et al. "The MPI emotional body expressions database for narrative scenarios." PloS one 9.12 (2014): e113647.

<sup>46</sup> De la Torre, Fernando, et al. "Guide to the carnegie mellon university multimodal activity (cmu-mmact) database." (2009).

<sup>47</sup> Mason, Ian, Sebastian Starke, and Taku Komura. "Real-time style modelling of human locomotion via feature-wise transformations and local motion phases." Proceedings of the ACM on Computer Graphics and Interactive Techniques 5.1 (2022): 1-18.

<sup>48</sup> Zhang, Mingming, et al. "Kinematic dataset of actors expressing emotions." Scientific data 7.1 (2020): 292.

<sup>49</sup> Kobayashi, Makito, et al. "Motion Capture Dataset for Practical Use of AI-based Motion Editing and Stylization." arXiv preprint arXiv:2306.08861 (2023).

- Table 7. Experiment accuracy results on three mentioned datasets (best results are reported)

Dataset	Data	Classifier	Train	Test	Features	Classes
<b>Xia</b>	Feature Extracted	Gradient Boosting	100 %	<b>66 %</b>	Statistical features from the Spatial and Wavelet domain	4 emotions of Angry, Depressed, Neutral, and Proud
	Raw	Gradient Boosting	100 %	<b>89 %</b>	Interpolated raw data (equal frames)	
<b>Edin Locomotion</b>		Decision Tree	100 %	<b>61 %</b>	Statistical features from the Spatial, Fourier, Short Time Fourier Transform, Wavelet, and Hilbert transform	7 actions of Jog, Jog side step, Run, Run side step, Transition, Walk, Walk side step
	Raw	Gradient Boosting	100 %	<b>77 %</b>	Interpolated raw data (equal frames)	
<b>BFA</b>	Feature Extracted	Gradient Boosting	100 %	<b>80 %</b>	Statistical features from the Spatial and Wavelet domain	5 emotions of Angry, Depressed, Neutral, Proud, and Happy
	Raw	Gradient Boosting	100 %	<b>80 %</b>	Interpolated raw data (equal frames)	

## - Limitations

## Challenges and

Based on our research, a few challenges and limitations<sup>50 51 52</sup> have been found. Some of these challenges are summarized below:

- **Complexity of Emotion Representation:** Human emotions are complex and multidimensional. They are not only expressed through body movements but also through facial expressions, tone of voice, and context. This complexity makes it difficult to accurately interpret emotions based solely on body motion data.
- **Individual Variability:** There is significant variation in how different individuals express emotions through their body movements. What might be a sign of happiness in one person could be a sign of discomfort in another. This individual variability requires personalized models, which can be challenging to develop and scale.
- **Cultural Differences:** Body language can vary significantly across cultures. For instance, gestures or postures that indicate a certain emotion in one culture might have a completely different meaning in another. This poses a challenge for creating universally applicable emotion recognition systems.
- **Real-Time Processing Constraints:** Processing body motion data in real-time for immediate emotion recognition can be computationally demanding, especially when dealing with high-resolution data or complex algorithms. This poses a challenge for implementation in real-world, resource-constrained environments.
- **Contextual Relevance:** Emotions are often context-dependent. Without understanding the context in which a body movement occurs, it can be difficult to accurately interpret the emotion it represents. This limitation can lead to misinterpretations in emotion recognition.
- **Integration with Other Modalities:** To improve accuracy, emotion recognition systems often need to integrate body motion data with other modalities like voice. However, effectively integrating and synchronizing these different data types can be technically challenging.

Also, some limitations are:

- **Data Quality and Availability:** High-quality, comprehensive datasets are crucial for training accurate models. However, collecting such datasets is challenging due to privacy concerns, the need for a diverse range of participants, and the complexity of accurately annotating emotional states.
- Additionally, emotion recognition through body tracking faces two main challenges: there's not enough data available, particularly in educational settings, and for emotions tied to the concept of flow. Collecting detailed and varied data in schools is difficult due to privacy concerns and practical issues, while the unique and personal nature of flow states complicates data collection.

---

<sup>50</sup> Sapiński, Tomasz, et al. "Emotion recognition from skeletal movements." *Entropy* 21.7 (2019): 646.

<sup>51</sup> Riemer, Hila, et al. "Emotion and motion: Toward emotion recognition based on standing and walking." *Plos one* 18.9 (2023): e0290564.

<sup>52</sup> Ahmed, Ferdous, ASM Hossain Bari, and Marina L. Gavrilova. "Emotion recognition from body movement." *IEEE Access* 8 (2019): 11761-11781.



These limitations prevent the technology's ability to accurately identify emotions in these specific contexts.

- **Sensor Limitations and Accuracy:** The accuracy of body tracking technologies, such as motion capture systems or wearable sensors, can vary. Factors like lighting conditions, sensor placement, and environmental interference can affect the quality of the data collected, leading to less accurate emotion recognition.
- **Ethical and Privacy Concerns:** The intrinsic privacy and ethical issues associated with monitoring and analyzing body movements, particularly in sensitive or private settings.
- **Lack of Standardized Benchmarks:** The absence of standardized benchmarks and evaluation metrics in the field, which hampers the comparison of different approaches and the measurement of progress.

---

## 3.2. TRAINING TOOLS

### 3.2.1. Technical Documentation

---

#### 3.2.1.1. Description

Training tools involve Enablers 2, 3, and 4 with related components, a total of five tools, for pre-training and fine-tuning models used in XR2Learn. Each tool is a modularized component with an isolated environment and dependencies that can be used separately, in combination, or as an end-to-end system (together with the Command-Line Interface – CLI).

The Training tools' architecture was designed to deploy each modality separately, e.g., audio and bio measurements (BM) modalities, to better manage and isolate the different dependencies per modality. Each component is deployed using Docker to ensure easy-to-use components, reproducible development and deployment environments, and consistent results. Thus, the Training tools support cross-platform use, i.e., Windows, Linux and macOS.

**Pre-processing:** Pre-process raw data into an organized time window of data and labels to be used by the other components.

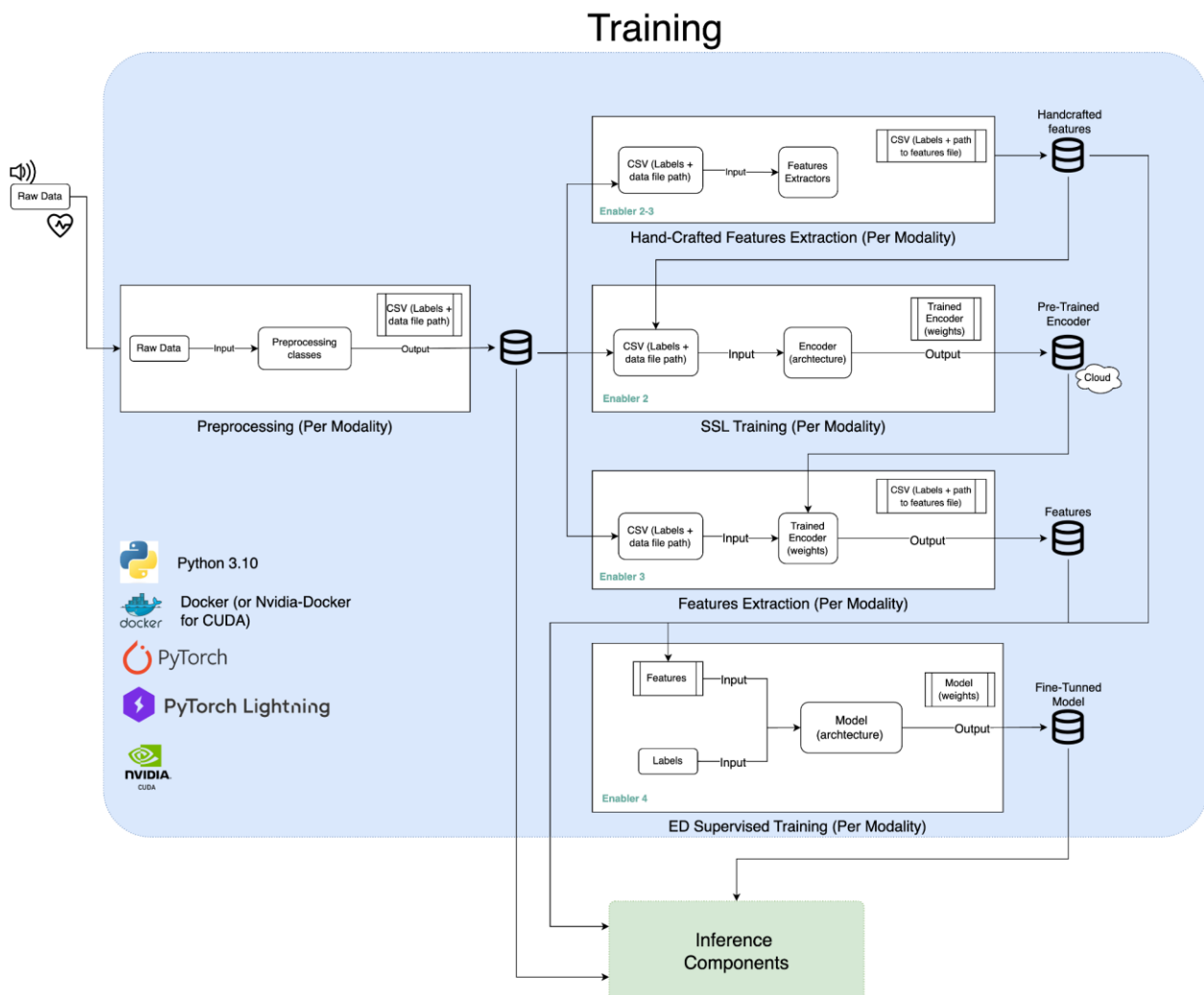
**Handcrafted Features Extraction:** Extracts features derived from the raw data type's properties instead of using Machine Learning for feature extraction. This component can be considered as part of Enabler 2 or 3 depending on the use-case. Specifically, generated handcrafted features can be used as inputs for both SSL (Enabler 2) and Supervised Training (Enabler 4). In other words, our architecture allows a user to decide whether they would like to pre-train the encoder with SSL or use it as a final representation for Supervised Learning.

**Self-Supervised Learning (SSL) Training (pre-train):** Pre-train an encoder (Enabler 2), with no use of labels.

**Features Extraction:** Uses an encoder to generate features (Enabler 3).

**Supervised Learning Training (fine-tuning):** Trains a classification model (Enabler 4) utilizing labels.

Figure 20 below depicts a diagram of the five components from the Training domain and their communication. Outputs produced by each component in the Training domain can be accessed by a user directly, and used as input by Inference components.



- Figure 20. Training domain architecture with its components connections.

### 3.2.1.2. Prerequisites

The Training Tools support the three main Operational Systems (OS): Linux, macOS, and Windows, as well as CPU and GPU use.

The two pre-requisites are:

- Docker<sup>53</sup> installed (or Docker-Nvidia<sup>54</sup> if GPU use is required)
- Python 3.10<sup>55</sup> installed

### 3.2.1.3. Installation

<sup>53</sup> <https://docs.docker.com/engine/install/>

<sup>54</sup> <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html>

<sup>55</sup> <https://www.python.org/downloads/>

1. Download the latest version of the code at:

<https://github.com/XR2Learn/Enablers-2-4-Training/tags>

2. Unzip the file
3. Navigate to the root directory of the downloaded project, and from the root repository, run the command to build the docker images
  - a. `docker compose build`
4. If using GPU, also build the docker images for GPU
  - a. `docker compose -f docker-compose.yml -f docker-compose-gpu.yml build`  
(for Windows systems)
  - b. `./compose-gpu.sh build` (for Unix based systems, i.e., Linux and MacOS)

#### 3.2.1.4. Basic User manual

The enablers were designed to be used with Enablers-CLI, a command-line interface that simplifies the use of enablers, so the easiest way to access the Enablers' functionalities is by using Enablers-CLI. Please refer to Section 3.5 for information on how to use CLI. However, if changing or expanding the enablers' functionalities is required, it is possible to access each component using docker commands, as exemplified below. Thus, the instructions described below are focused on running the enablers for a *development* environment.

A “*configuration.json*” file is required to provide the enablers with the necessary specifications for running. A default version of “*configuration.json*” is provided and can be changed by the user.

1. Run a docker image:
  - a. `docker compose run --rm <service-name>`

Note: Service names can be found in the “*docker-compose.yml*” file in the project's root folder. Each modality, i.e., audio, bio-measurements (bm), body movements, are deployed in separated docker containers and their service name follow the structure:

1. `pre-processing-<modality>`
2. `handcrafted-features-generation-<modality>`
3. `ssl-<modality>`
4. `ssl-features-generation-<modality>`
5. `ed-training-<modality>`

There is an additional script to run all the docker images from a given modality, which uses the available ‘*configuration.json*’ file:

1. For Unix-based OS, MacOS and Linux
  - a. `./run_all_dockers.sh`
2. For Windows:
  - a. `./run_all_dockers.ps1`

Additional Useful Commands:

1. Build a specific docker image
  - a. `docker compose build <service-name>`
2. Run a specific docker image
  - a. `docker compose run --rm <service-name>`
3. Run a specific docker image providing Environment Variables in the format KEY=VALUE:
  - a. `KEY=VALUE docker compose run --rm <service-name>`
4. Run a specific docker image with shell entry point
  - a. `docker compose run --rm <service-name> \bin\bash`

All the outputs produced by any component in the Training domain are saved and can be accessed in the folder `./outputs`.

### 3.2.1.5. Open-source Code

The Training tools can be found in the project's GitHub repository at:

- <https://github.com/XR2Learn/Enablers-2-4-Training>

#### LICENSE

The Training tools code is shared under a dual-licensing model. For non-commercial use, it is released under the MIT<sup>56</sup> open-source license. A commercial license is required for commercial use.

The handcrafted features extraction components for the audio modality is shared for non-commercial use only, to comply with OpenSMILE<sup>57</sup> license.

More details on the End User License Agreement (EULA) can be found at:

<https://github.com/XR2Learn/Enablers-2-4-Training/blob/master/LICENSE>

#### Available versions

- Table 8. Training tools enablers and components released versions and date

Version	Release Date
<b>v0.1.0</b>	<b>2023-10-19</b>
<b>v0.2.0</b>	<b>2023-11-14</b>
<b>v0.3.0</b>	<b>2023-12-06</b>
<b>v0.3.1</b>	<b>2024-01-19</b>
<b>v0.3.2</b>	<b>2024-02-15</b>

All the released versions can be accessed at: <https://github.com/XR2Learn/Enablers-2-4-Training/tags>

<sup>56</sup> <https://opensource.org/license/mit/>

<sup>57</sup> <https://github.com/audeering/opensmile-python/tree/main>

Table 8 lists the Training tools released versions with date. A description of the main changes in the project's versions can be found at:

<https://github.com/XR2Learn/Enablers-2-4-Training/blob/master/CHANGELOG.md>

---

### 3.2.2. Enabler 2: Emotion Representation Learning Tool

---

#### 3.2.2.1. Description

Enabler 2 includes the components for pre-processing, handcrafted feature extraction, and the tool for pre-training Deep Learning emotion recognition models. As stated before, emotion representations generated by pre-training models will be used for supervised training, and the different modalities for each component are deployed separately.

The pre-processing component is responsible for processing the raw input files into organized, structured data, generating CSV files in which each row represents an example of the given data with the corresponding label. This component is also responsible for splitting the input data into train, validation and test sets in a consistent approach to prevent data leakage across the different enablers' pipelines.

The handcrafted features component is responsible for extracting handcrafted features, i.e., numerical representation derived from the raw input data's properties. This component uses the input data already pre-processed and split into train, validation and test sets. For the audio modality, MFCCs and eGEMAPS handcrafted feature extraction are supported.

The SSL component aims to pre-train Deep Learning encoders for emotion recognition and learn representations that can be later used in Enablers 3 and 4. The current pipeline supports the SimCLR contrastive learning framework<sup>58</sup> utilizing augmentations of input data. This framework and augmentations can be applied on top of raw speech signals as well as handcrafted features (e.g., spectrograms for the audio modality) to pre-train architectures of deep learning models previously described in Section 3.1.1.2 (Table 1). This SSL pipeline allows users to pre-train Deep Learning architectures of their choice, taking into account their computational capabilities. Meanwhile, users can choose not to pre-train their own model and fine-tune the open-source pre-trained models (currently, base and large versions of wav2vec 2.0 are supported) within Enabler 4. In this case, as discussed in Section 3.1.1.3 (Table 3), more powerful equipment (GPUs) are needed. In the second sub-phase of Task 3.2, we plan to upgrade the current pre-training pipelines and extend them to other modalities (e.g., bio-measurements).

#### 3.2.2.2. Basic User manual

There are two approaches to run Enabler 2 directly: using a docker image or a local run.

1. For Docker running:
  - a. Running with CPU
    - i. `docker compose run --rm ssl-<modality>` (for running with CPU)
  - b. Running with GPU

---

<sup>58</sup> Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.

- i. `./compose-gpu.sh run --rm ssl-<modality>` (for Unix based OS, i.e. MacOS and Linux)
  - ii. `docker compose -f docker-compose.yml -f docker-compose-gpu.yml run --rm ssl-<modality>` (for Windows)
2. For local running:
  - a. Download the specific component from the Training tools repository (e.g., `SSL_Training/SSL_Audio_Modality`)
  - b. Prepare the virtual environment (Create and activate virtual environment with `venv`).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt` (for running with CPU)
    - ii. `pip install -r requirements-gpu.txt` (for running with CUDA)
  - d. Change the “`configuration.json`” file according to the desired use-case
  - e. Run the script
    - i. `python pre-train.py`

(Optional) Running Handcrafted Feature Extraction component can also be done locally or using a docker image.

1. For Docker running:

```
docker compose run --rm handcrafted-features-generation-<modality>
```

2. For local running:

- a. Download the specific component from the Training tools repository (e.g., `Handcrafted_Features_Extraction/Handcrafted_Features_Extraction_Audio_modality`)
- b. Prepare the virtual environment (Create and activate virtual environment with `venv`).
  - i. `python -m venv ./venv`
  - ii. `source ./venv/bin/activate`
- c. Install the requirements within the virtual environment
  - i. `pip install -r requirements.txt`
- d. Change the “`configuration.json`” file according to the desired use-case
- e. Run the script
  - i. `python generate_features.py`

---

### 3.2.3. Enabler 3: Tools for Using Emotion Representations

---

#### 3.2.3.1. Description

Enabler 3 involves the tool for using pre-trained models in order to generate representations (feature extraction). This component uses a pre-trained model, trained by the Enabler 2 component or a customized pre-trained model provided by the user, to compute features for a given modality.

The functionality of this component is a crucial part of deploying an Inference system (Section 3.3), i.e., utilizing the pre-trained models to generate feature representations of the input data. These representations can later be used to train and utilize an emotion classification model within the supervised training component (Enabler 4; Section 3.2.4) and the Inference tool (Section 3.3). Additionally, users that are only interested in obtaining the feature representations for their custom dataset can use this component as a standalone module.

As mentioned previously in Section 3.2.1.1, Handcrafted Feature Extraction component can also be considered part of Enabler 3 in the case where these representations are preferred to be used for Supervised Training (Enabler 4) skipping SSL. A description of this component and how to use it can be found in Section 3.2.2.

### 3.2.3.2. Basic User manual

There are two approaches to run Enabler 3 directly: using a docker image or a local run.

1. For Docker running
  - a. Running with CPU
    - i. `docker compose run --rm ssl-features-generation-<modality>`
  - b. Running with GPU
    - i. `./compose-gpu.sh run --rm ssl-features-generation-<modality>` (for Unix based OS, i.e. MacOS and Linux)
    - ii. `docker compose -f docker-compose.yml -f docker-compose-gpu.yml run --rm ssl-features-generation-<modality>` (for Windows)
2. For local running:
  - a. Download the specific component from the Training tools repository (e.g., `SSL_Features_Extraction/SSL_Features_Extraction_Audio_Modality`)
  - b. Prepare the virtual environment (Create and activate the virtual environment with `venv`).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt` (for running with CPU)
    - ii. `pip install -r requirements-gpu.txt` (for running with CUDA)
  - d. Change the “`configuration.json`” file according to use-case
  - e. Run the script
    - i. `python generate_features.py`

---

## 3.2.4. Enabler 4: Unimodal Emotion Classification Tools

### 3.2.4.1. Description

Enabler 4 is a tool for fine-tuning Deep Learning Emotion Recognition models through supervised training with labeled data. This component can receive the emotion representations generated by Enabler 3 as input. During supervised training of the emotion classification model, this component can utilize the Encoder, i.e., the pre-trained model generated by Enabler 2 (Section 3.2.2) with frozen weights or perform further fine-tuning of the pre-trained model, according to the user’s configuration. Alternatively, Enabler 4 can also receive pre-processed data (output of the Pre-processing component, Section 3.2.2) as input and implement the entire pipeline to train both the encoder and the classification model. This is helpful for users who want to use Enabler 4 as a standalone tool.

For the supervised training phase, this component employs a linear classification on top of the encoder or feature representations, in which the user can set hyperparameters and other options through a configuration file, such as the number of epochs, batch size, learning rate, and optimizer algorithm.

### 3.2.4.2. Basic User manual

There are two approaches to run Enabler 4 directly: using a docker image or a local run.



1. For Docker running
  - a. Running with CPU
    - i. `docker compose run --rm ed-training-<modality>`
  - b. Running with GPU
    - i. `./compose-gpu.sh run --rm ed-training-<modality>` (for Unix based OS, i.e. MacOS and Linux)
    - ii. `docker compose -f docker-compose.yml -f docker-compose-gpu.yml run --rm ed-training-<modality>` (for Windows)
2. For local running:
  - a. Download the specific component from the Training tools repository (e.g., Supervised\_Training/Supervised\_Audio\_Modality)
  - b. Prepare the virtual environment (Create and activate the virtual environment with venv).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt` (for running with CPU)
    - ii. `pip install -r requirements-gpu.txt` (for running with CUDA)
  - d. Change the “configuration.json” file according to use-case
  - e. Run the script
    - i. `python train.py`

---

### 3.3. INFERENCE TOOLS

#### 3.3.1. Technical Documentation

---

##### 3.3.1.1. Description

These tools are designed for the unimodal and multimodal emotion classification and evaluation in XR2Learn. This set of tools includes Emotion Classification (per modality), Multimodal Fusion and Evaluation components. Each tool is a modularized component with an isolated environment and dependencies that can be used separately, in combination, or as an end-to-end system (together with the Command-Line Interface – CLI).

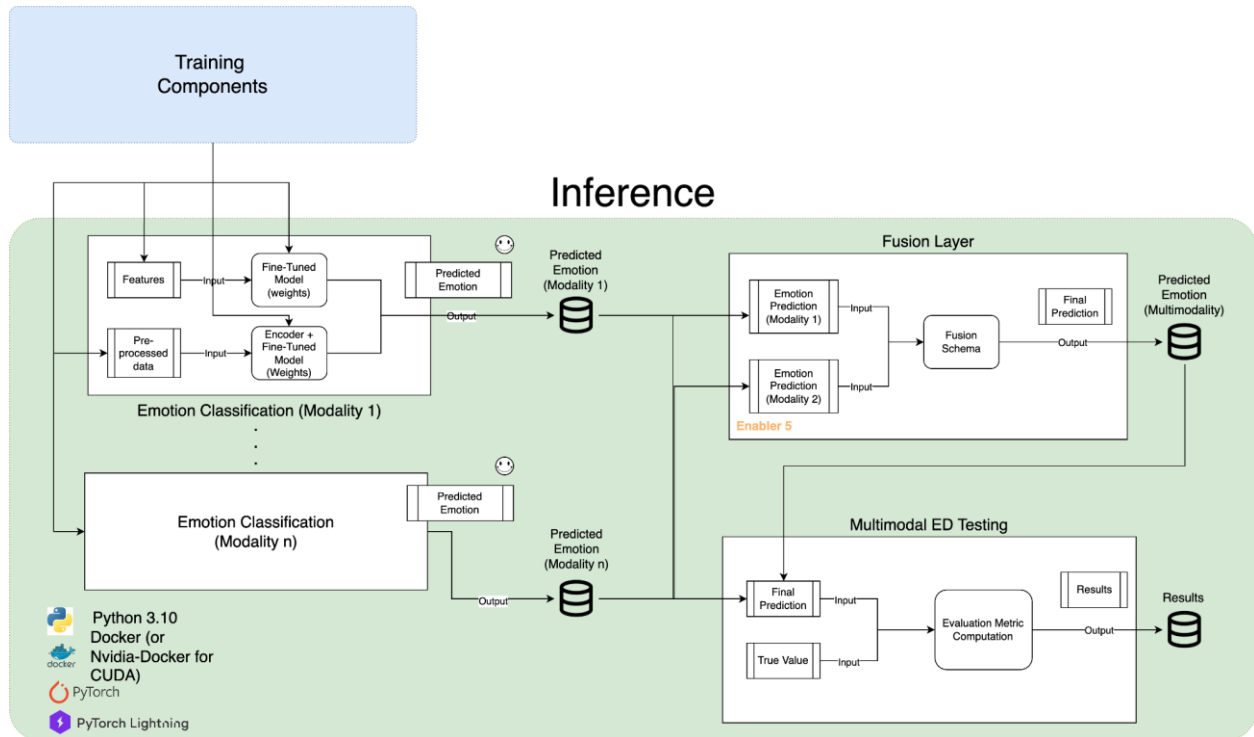
Each component is deployed using Docker to ensure easy-to-use components, reproducible development and deployment environments, and consistent results.

**Emotion Classification:** a component to recognize emotions. Each modality has a separated emotion classification component.

**Multimodal Fusion:** a component to execute a decision-level emotion detection multimodal fusion, i.e., to compute the combination of emotions from different modalities.

**Evaluation:** a component to evaluate a uni/multimodal emotion detection model according to different evaluation metrics.

Figure 21 below depicts the Inference components' overall architecture and communication. As can be observed, Inference components utilize outputs produced by Training components.



- Figure 21. An overview of the Inference domain architecture with its components connections.

### 3.3.1.2. Prerequisites

The Inference Tools support the three main Operational Systems (OS): Linux, MacOS, and Windows.

The two pre-requisites are:

- Docker<sup>59</sup> installed
- Python 3.10<sup>60</sup> installed

### 3.3.1.3. Installation

1. Download the latest version of the code at:
  - a. <https://github.com/XR2Learn/Enabler-5-Inference/tags>
2. Unzip the file
3. Navigate to the root directory of the downloaded project, and from the root repository, run the command to build the docker images
  - a. `docker compose build`

### 3.3.1.4. Basic User Manual

The enablers were designed to be used with Enablers-CLI, a command-line interface that simplifies the use of enablers, so the easiest way to access the enablers' functionalities is by using Enablers-CLI. Please refer to Section 3.5 for information on how to use CLI.

<sup>59</sup> <https://docs.docker.com/engine/install/>

<sup>60</sup> <https://www.python.org/downloads/>

However, if changing or expanding the enablers' functionalities is required, it is possible to access each component using docker commands, as exemplified below. Thus, the instructions described below are focused on running the enablers for a *development* environment.

A “*configuration.json*” file is required to provide the enablers with the necessary specifications for running. A default version of “*configuration.json*” is provided and can be changed by the user.

Run a docker image:

b. `docker compose run --rm <service-name>`

**Note 1:** Service names can be found in the “*docker-compose.yml*” file in the project's root folder. Each modality, i.e., audio, bio-measurements (bm), body movements, are deployed in separated docker containers and their service name follow the structure:

1. emotion-classification-<modality>
2. fusion-layer
3. ed-evaluation

**Note 2:** Some additional services can be found in the Inference domain “*docker-compose.yml*” file, namely:

- redis
- personalisation-tool
- demo-ui

These services are present in the Inference domain to facilitate *development* and are explained in detail in Section 3.4, Personalization Tool.

There is an additional script to run all the docker images from a given modality, which will use the available ‘*configuration.json*’ file:

For Unix-based OS, MacOS and Linux

b. `./run_all_dockers.sh`

For Windows:

c. `./run_all_dockers.ps1`

Additional Useful Commands:

1. Build a specific docker image
  - a. `docker compose build <service-name>`
2. Run a specific docker image
  - a. `docker compose run --rm <service-name>`
3. Run a specific docker image providing Environment Variables in the format KEY=VALUE:
  - a. `KEY=VALUE docker compose run --rm <service-name>`
4. Run a specific docker image with shell entry point
  - a. `docker compose run --rm <service-name> \bin\bash`

**All the outputs produced by any component in the Inference domain are saved and can be accessed in the folder `./outputs`.**

### 3.3.1.5. Open-source Code

The Inference tools can be found in the project's GitHub repository at:

- <https://github.com/XR2Learn/Enabler-5-Inference>

#### LICENSE

The Inference tools code is shared under a dual-licensing model. For non-commercial use, it is released under the MIT<sup>61</sup> open-source license. A commercial license is required for commercial use.

More details on the End User License Agreement (EULA) can be found at:

<https://github.com/XR2Learn/Enabler-5-Inference/blob/master/LICENSE>

#### Available versions

- Table 9. Inference tools enablers and components released versions and date.

Version	Release Date
v0.1.0	2023-10-27
v0.1.1	2023-12-06
v0.2.0	2024-01-09
v0.3.0	2024-01-19
v0.3.1	2024-02-15

All the released versions can be accessed at: <https://github.com/XR2Learn/Enabler-5-Inference/tags>

Table 9 lists the Inference tools released versions with date. A description of the main changes in the project's versions can be found at: <https://github.com/XR2Learn/Enabler-5-Inference/blob/master/CHANGELOG.md>

---

## 3.3.2. Enabler 5: Multimodal Fusion Tools

### 3.3.2.1. Description

This enabler combines identified emotions from multiple modalities into a final predicted emotion. The current version implements a decision-level multimodal fusion type using a majority voting fusion schema at the class probability level.

This enabler's inputs are the predicted emotions from each modality (as a .csv file with .npy file paths containing the predicted emotion label array).

For the generated output, Enabler 5 supports two different formats:

---

<sup>61</sup> <https://opensource.org/license/mit/>

1. A .csv file with the final fusion predicted emotions that can be found in /outputs/predictions.csv (currently default behavior)
2. Enabler 5 can also behave as a publisher component, publishing the fusion predicted emotion according to the Publisher/Subscriber messaging protocol.

To set up the multimodal fusion layer as a publisher, add "publisher: true" under the key "inference\_config". By setting the publisher configuration as true, this component can communicate with Personalization tool components using a publisher/subscriber messaging protocol as described in the following section, Personalization Tool.

### 3.3.2.2. Basic User Manual

There are two approaches to run Enabler 5 directly: using a docker image or a local run.

1. For Docker running:
  - a. `docker compose run --rm fusion-layer`
2. For local running:
  - a. Download the specific component from the Inference tools repository (e.g., Multimodal\_Fusion/Multimodal\_Fusion\_Layer)
  - b. Prepare the virtual environment (Create and activate virtual environment with venv).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt`
  - d. Change the "configuration.json" file according to the desired use-case
3. Run the script
  - a. `python predict.py`

---

## 3.3.3. Emotion Classification and Model Evaluation

---

### 3.3.3.1. Description

The emotion classification component utilizes the fine-tuning model's weights trained by the supervised training component (Enabler 4; Section 3.2.4) in the Training domain to identify the emotion from extracted features. It also can use the pre-trained encoder generated in the Training domain (Enabler 2; Section 3.2.2) combined with the fine-tuned model to predict emotions from pre-processed data.

Each modality of this component is deployed separately in different Docker container images to allow a separate environment for each modality and dependencies, thus facilitating the implementation and extension of additional modalities.

The Model Evaluation component utilizes the final predictions generated by the multimodal fusion component, or a given unimodality prediction, to evaluate the combined pipeline performance according to different evaluation metrics, for instance, accuracy, recall, precision, and confusion matrix.

### 3.3.3.2. Basic User Manual

There are two approaches to run an Emotion Classification component directly: using a docker image or a local run.

1. For Docker running:
  - a. `docker compose run --rm emotion-classification-<modality>`
2. For local running:

- a. Download the specific component from the Inference tools repository (e.g., Emotion\_Classification/Emotion\_Classification\_Audio\_Modality)
- b. Prepare the virtual environment (Create and activate virtual environment with venv).
  - i. `python -m venv ./venv`
  - ii. `source ./venv/bin/activate`
- c. Install the requirements within the virtual environment
  - i. `pip install -r requirements.txt`
- d. Change the “configuration.json” file according to the desired use-case
- e. Run the script
  - i. `python predict.py`

Similarly to other components, there are two approaches to run the Emotion Detection Evaluation component directly: using a docker image or a local run.

1. For Docker running:
  - a. `docker compose run --rm ed-evaluation`
2. For local running:
  - a. Download the specific component from the Inference tools repository (e.g., ED\_Evaluation/ED\_Evaluation)
  - b. Prepare the virtual environment (Create and activate virtual environment with venv).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt`
  - d. Change the “configuration.json” file according to the desired use-case
  - e. Run the script
    - i. `python evaluate.py`

---

## 3.4. PERSONALIZATION TOOL

### 3.4.1. Technical Documentation

---

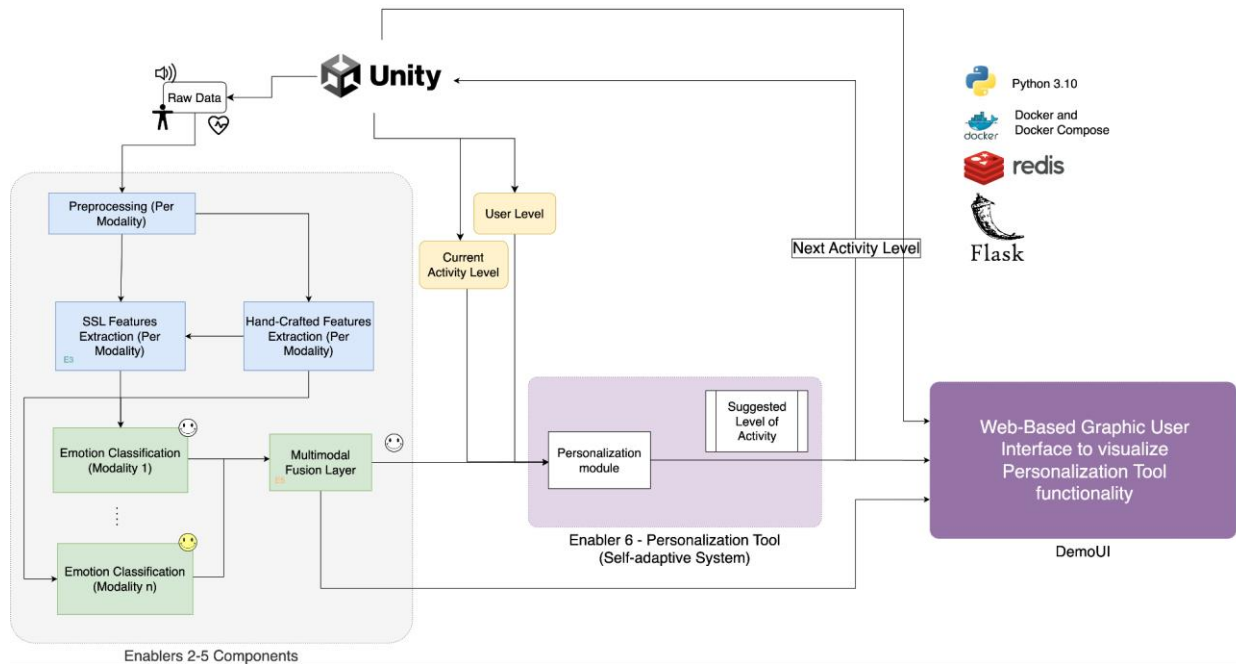
#### 3.4.1.1. Description

The Personalization Tool utilizes the user’s predicted emotions as the output of the Training and Inference domain, together with contextual information, e.g., a user and activity levels, to provide personalized suggestions on the recommended activity level for the user in educational XR applications.

The Personalization Tool exploits the Publisher/Subscriber messaging protocol implemented using Redis to provide asynchronous, real-time communication between the Personalization Tool, Inference domain and an XR educational software implemented using Unity.

A web-based DemoUI is also provided as a graphic interface for better visualizing the personalization tool functionality and how it communicates with the other domain’s components, i.e., multimodal fusion layer and Unity application.

Figure 22 below depicts the Personalization Tool architecture with its different components and communication with Training and Inference components, represented by blue (Training) and green (Inference) colors.



- Figure 22. Overview of the Personalization tool's architecture with its components communicating with the Training (blue) and the Inference (green) components.

### 3.4.1.2. Prerequisites

Personalization Tool supports the three main Operational Systems (OS): Linux, MacOS, and Windows.

The two prerequisites are:

1. Docker<sup>62</sup> installed
2. Python 3.10<sup>63</sup> installed

### 3.4.1.3. Installation

1. Download the latest version of the code at:
  - a. <https://github.com/XR2Learn/Enabler-6-Personalisation-Tool>
2. Unzip the file
3. Navigate to the root directory of the downloaded project, and from the root repository, run the command to build the docker images
  - a. `docker compose build`

For installing locally:

1. Personalization Tool
  - a. Navigate to the directory Personalisation\_Tool
  - b. Prepare the virtual environment (Create and activate virtual environment with venv).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`

<sup>62</sup> <https://docs.docker.com/engine/install/>

<sup>63</sup> <https://www.python.org/downloads/>



- c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt`
2. DemoUI
  - a. Navigate to the directory DemoUI
  - b. Prepare the virtual environment (Create and activate virtual environment with venv).
    - i. `python -m venv ./venv`
    - ii. `source ./venv/bin/activate`
  - c. Install the requirements within the virtual environment
    - i. `pip install -r requirements.txt`

#### 3.4.1.4. Basic User Manual

Personalization Tool can be used as a standalone application or with Enablers-CLI, a command-line interface to simplify the use of Enablers. The easiest way to access the Personalization Tool's functionalities is by using Enablers-CLI. Please refer to Section 3.5 for information on how to use CLI.

However, if changing or expanding the Personalization tool's functionalities is required, it is possible to access each component using docker commands, as exemplified below. Thus, the instructions described below are focused on a *development* environment.

A “*configuration.json*” file is required to provide the components with the necessary specifications for running. A default version of “*configuration.json*” is provided and can be changed by the user.

To run all the docker images and access the DemoUI:

1. Run the command:
  - a. `docker compose up -d`
2. Go to the URL to access the DemoUI:
  - a. <http://127.0.0.1:8000/>

#### 3.4.1.5. Open-source code

Personalization Tool can be found in the project's GitHub repository at:

- <https://github.com/XR2Learn/Enabler-6-Personalisation-Tool>

#### LICENSE

The Personalization tool code is shared under a dual-licensing model. For non-commercial use, it is released under the MIT<sup>64</sup> open-source license. A commercial license is required for commercial use.

Fine-tuned models created using the RAVDESS dataset are shared under the CC BY-NC-SA 4.0 license to comply with the RAVDESS license, as the models are derivative works from this dataset.

More details on the End User License Agreement (EULA) can be found at:

<https://github.com/XR2Learn/Enabler-6-Personalisation-Tool/blob/master/LICENSE>

---

<sup>64</sup> <https://opensource.org/license/mit/>

## Available versions

- Table 10. Personalization tool components released versions and date.

Version	Release Date
v0.1.0	2023-10-31
v0.1.1	2023-12-06
v0.2.0	2024-01-19
v0.2.1	2024-02-15

All the released versions can be accessed at: <https://github.com/XR2Learn/Enabler-6-Personalisation-Tool/tags>

Table 10 lists the Personalization tool released versions with date. A description of the main changes in the project's versions can be found at: <https://github.com/XR2Learn/Enabler-6-Personalisation-Tool/blob/master/CHANGELOG.md>

---

### 3.4.2. Enabler 6: Personalization tool

---

#### 3.4.2.1. Description

This tool is intended to recommend adjusting learning material difficulty based on the current learning application level, the user's skill level and current emotions. It uses components from Training Tools and Inference Tools to process the input data, identify the user's emotions and calculate the suggested next learning activity level.

As mentioned earlier, this component implements a publisher/subscriber messaging protocol using Redis. Thus, it subscribes to channels to get information from Inference components (multimodal fusion layer) and an XR Unity application. While it also publishes information (the suggested activity level) to a different channel so other components can access and process this information.

#### 3.4.2.2. Basic User Manual

There are two approaches to run the Personalization tool directly: using docker images or a local run. Because this component implements a publisher/subscriber using Redis, it needs to have an instance of Redis service running so the personalization tool can publish and subscribe to channels (or topics). A `simulate_input_output.py` script is provided so a user can visualize (on a shell terminal) the Personalization tool functionality.

1. For local running
  - a. Run an instance of Redis:
    - i. `docker compose up redis -d`
  - b. Run the Python script (from inside the virtual environment):
    - i. `python personalisation_tool/suggest_activity_level.py`
  - c. Run (in a different terminal window or tab):
    - i. `python personalisation_tool/simulate_input_output.py`
2. For Docker running

- a. Run the command below (this command will automatically run an instance of Redis, so personalization tool service can run as well):
  - i. `docker compose run --rm personalisation-tool`
- b. Run (in a different terminal window or tab):
  - i. `python personalisation_tool/simulate_input_output.py`

To stop Redis service:

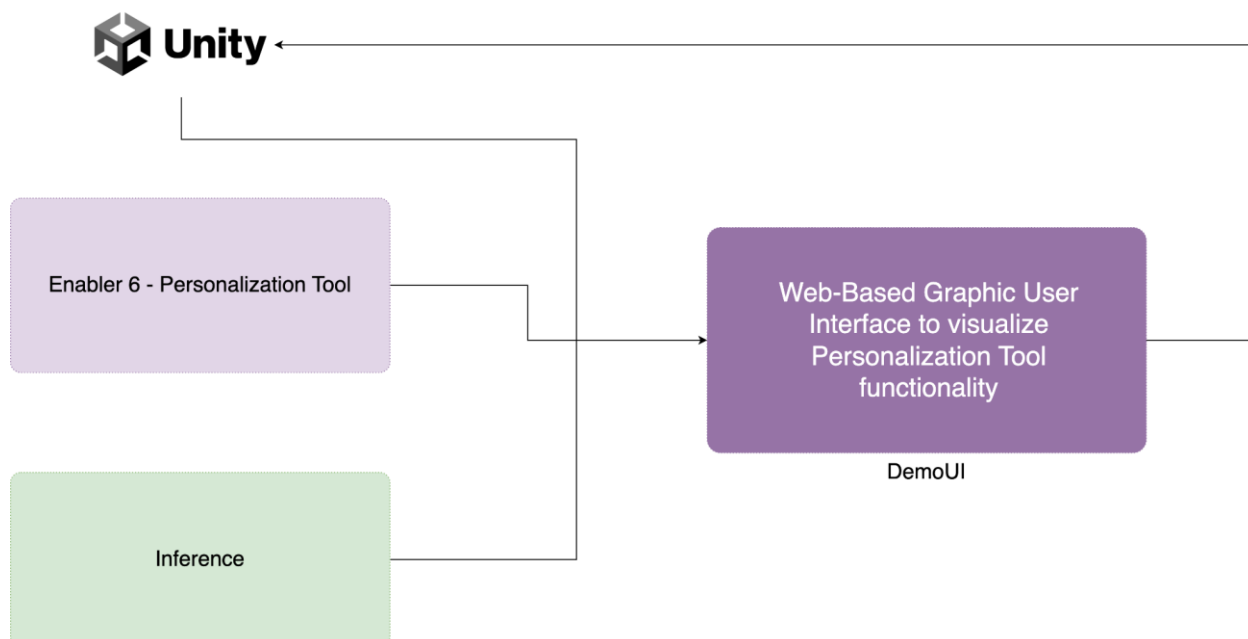
```
docker compose down
```

### 3.4.3. Demo UI

#### 3.4.3.1. Description

A web-based application (using Flask<sup>65</sup>) that uses websockets (socketio) to connect Javascript with the Flask server. The Flask server runs a background thread that publishes the websocket events into Redis channels and subscribes to channels as well, implementing a publisher/subscriber messaging protocol. The Flask server behaves as an API gateway that translates Redis protocol into websocket protocol. In simpler terms, DemoUI subscribes to messaging channels (or topics) to display the communication messages between Enabler 6 (Personalization tool; Section 3.4.2), Inference tools and XR Unity application, providing a graphic interface for better visualizing the Personalization Tool functionalities.

Figure 23 below depicts the architecture overview of the communication between the DemoUI with other components. DemoUI displays messages published by the Enabler 6 - Personalization Tool, Inference tool and the XR Unity application. While also displays the output of Enabler 6 - Personalization tool processing which will be eventually used by the XR Unity application.



<sup>65</sup> <https://flask.palletsprojects.com/en/3.0.x/>

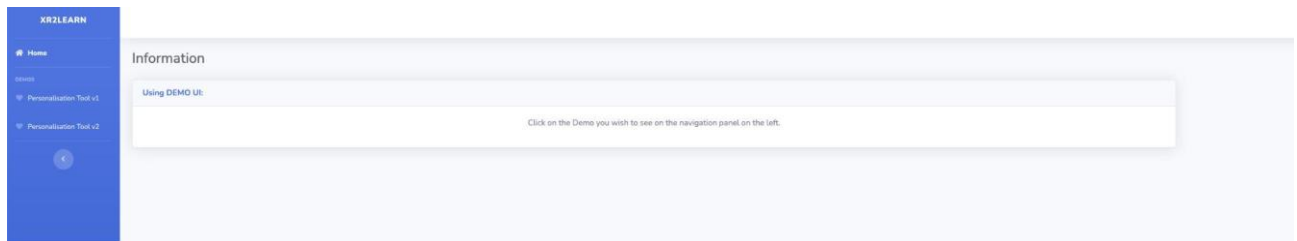
- Figure 23. Architecture overview of Demo UI communicating with other components: Enabler 6, Inference and XR Unity application.

### 3.4.3.2. Basic User Manual

DemoUI is a Flask web-based application and can be run locally or using a Docker image.

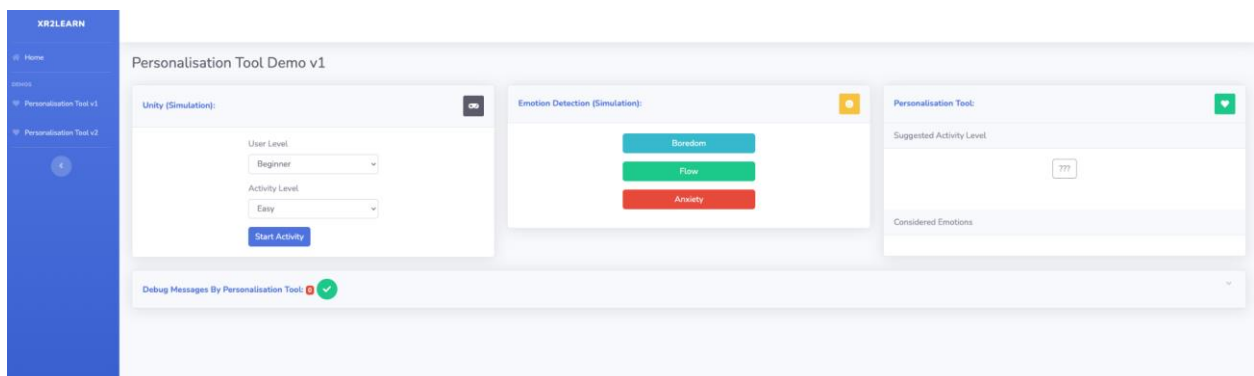
1. Running with Docker
  - a. `docker compose run --rm demo-ui`
  - b. Go to the URL <http://127.0.0.1:8000/> to access the DemoUI
2. Running locally
  - a. Start Redis server
    - i. `docker compose up redis -d`
  - b. From inside the virtual environment you prepared in Section 3.4.1.3, run the command:
    - i. `python web_app.py`

When you access the DemoUI, there are two different demos you can use, as shown by Figure 24 below.



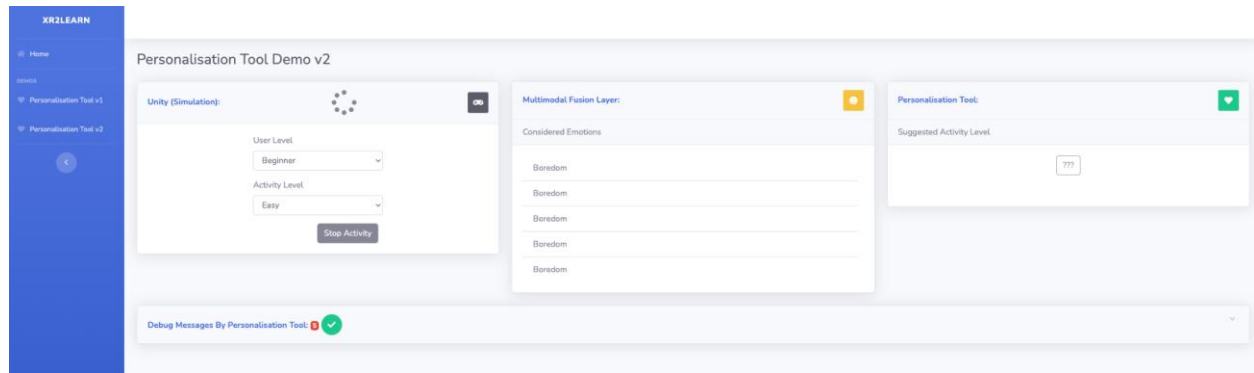
- Figure 24. DemoUI screenshot

1. Personalization Tool v1: DemoUI simulates the input from the Inference (emotion detection) and the XR Unity app. DemoUI displays the processed output from Enabler-6.



- Figure 25. DemoUI (version 1) screenshot, with Unity and Emotion Detection simulation

2. Personalization Tool v2: DemoUI simulates the input from the XR Unity app but receives the input from the Inference directly. DemoUI displays the processed output from Enabler-6.



- Figure 26. DemoUI (version 2) screenshot, with Unity simulation

The different versions of DemoUI showcase the Personalization Tool functionalities regardless of how many components are already deployed and connected with the whole system, demonstrating the robustness and flexibility of Enablers' modularized architecture.

## 3.5. COMMAND LINE INTERFACE (CLI)

### 3.5.1. Technical Documentation

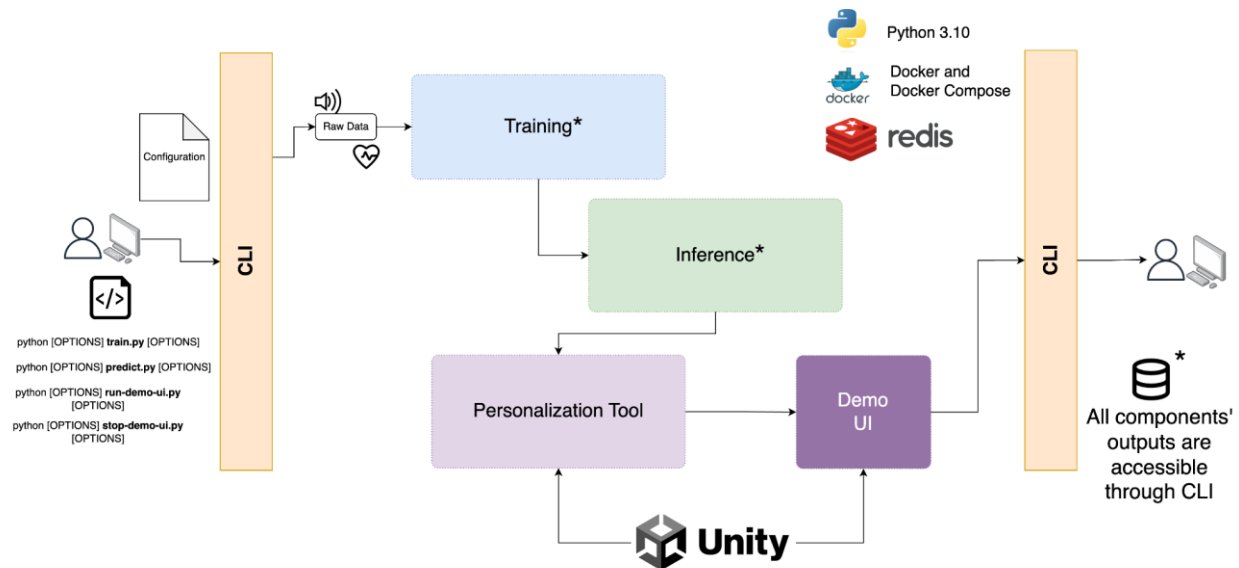
#### 3.5.1.1. Description

Enablers-CLI (Command Line Interface) was designed to facilitate the use of XR2Learn training, inference and personalization tools, i.e., Enablers 2-6 and their components. To access the Enablers' functionalities through CLI, you need two elements:

1. CLI commands and options
2. A *configuration.json* file (you can provide a JSON configuration file path as an option to the CLI command, if you do not provide a JSON configuration file path, the default file is *./configuration.json*).

A default *configuration.json* file is provided and it can be changed according to the use-case.

Figure 27 below depicts the Enablers and Enabler-CLI components and their communication. All components' outputs can be accessed through CLI at */output* folder.



- Figure 27. Overview of Enabler-CLI's architecture serving as a user entrypoint for accessing Training, Inference, Personalization tools and DemoUI functionalities.

### 3.5.1.2. Prerequisites

- Docker<sup>66</sup> installed (or Docker-Nvidia<sup>67</sup> if GPU use is required)
- Python 3.10<sup>68</sup> installed

### 3.5.1.3. Installation

1. Create virtual environment
  - a. `python -m venv ./venv`
  - b. `source ./venv/bin/activate`
2. Navigate to the XR2Learn-CLI directory
3. Install XR2Learn-CLI
  - a. `pip install -e .` (There is a full stop in the end of the command)

### 3.5.1.4. Basic User Manual

The general command format to use XR2Learn-CLI is:

```
python xr2learn_enablers_cli/xr2learn_enablers.py [OPTIONS] [COMMAND]
[OPTIONS]
```

For help with the options and commands, access a list of arguments and their description with:

```
python xr2learn_enablers_cli/xr2learn_enablers.py --help
```

Command examples:

1. Training:

<sup>66</sup> <https://docs.docker.com/engine/install/>

<sup>67</sup> <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html>

<sup>68</sup> <https://www.python.org/downloads/>

- a. `python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id model_001 train --dataset ravedss --features_type ssl --ssl_pre_train encoder_fe --ed_training true`
2. Inference (Predict):
  - a. `python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id model_001 predict --dataset ravedss`
3. Inference (Multimodal Fusion):
  - a. `python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id model_001 multimodal --dataset ravedss`
4. Inference (Evaluation):
  - a. `python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id model_001 evaluate --dataset ravedss`
5. Start Web-based DemoUI (for personalisation tool user interface):
  - a. `python xr2learn_enablers_cli/xr2learn_enablers.py run_personalisation`

Go to the URL <http://127.0.0.1:8080> to access the DemoUI

6. Stop Web-based DemoUI (for personalisation tool user interface):
  - a. `python xr2learn_enablers_cli/xr2learn_enablers.py stop-demo-ui`

## GPU

To use GPU, include an option with value true `--gpu true` before the command.

Example:

```
python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id model_001 --gpu true
train --dataset ravedss --features_type ssl --ssl_pre_train encoder_fe --ed_training
true
```

## Benchmarks

XR2Learn-CLI also includes pre-configured benchmarks, which represent use cases on the enablers functionalities and serve as integration tests for the end-to-end system formed by CLI and Enablers 2-5.

Some use cases included in the benchmarks are end-to-end systems for the audio modality using different representations of speech, for example spectrals, paralinguistic and transformer-based features.

1. Run benchmarks on Unix based OS:

```
`./run_benchmarks.sh`
```

2. Run benchmarks using GPU:

```
`GPU=true ./run_benchmarks.sh`
```

### 3.5.1.5. Open-source Code

The Training tool can be found in the project's GitHub repository at <https://github.com/XR2Learn/Enablers-CLI>



## LICENSE

The XR2Learn-CLI code is shared under a dual-licensing model. For non-commercial use, it is released under the MIT<sup>69</sup> open-source license. A commercial license is required for commercial use.

More details on the End User License Agreement (EULA) can be found at:

<https://github.com/XR2Learn/Enablers-CLI/blob/master/LICENSE>

## Available versions

- Table 11. Command line interface (CLI) released versions and date.

Version	Released Date
<b>v0.1.0</b>	<b>2023-11-14</b>
<b>v0.1.1</b>	<b>2023-11-28</b>
<b>v0.1.2</b>	<b>2023-12-07</b>
<b>v0.2.0</b>	<b>2024-01-09</b>
<b>v0.3.0</b>	<b>2024-01-11</b>
<b>v0.4.0</b>	<b>2024-01-19</b>
<b>v0.4.1</b>	<b>2024-02-15</b>

## Compatibility

CLI v0.1.x is compatible with:

- XR2Learn Training v.0.1.0, v0.2.0, v0.3.0
- XR2Learn Inference v.0.1.X

CLI v0.2.0 is compatible with:

- XR2Learn Training v.0.1.0, v0.2.0, v0.3.0
- XR2Learn Inference v.0.2.X

CLI v0.3.0 is compatible with:

- XR2Learn Training v.0.1.0, v0.2.0, v0.3.0
- XR2Learn Inference v.0.2.X
- XR2Learn Personalisation v.0.1.X

CLI v0.4.X is compatible with:

- XR2Learn Training v.0.1.0, v0.2.0, v0.3.0
- XR2Learn Inference v.0.2.X, v.0.3.0

<sup>69</sup> <https://opensource.org/license/mit/>

- XR2Learn Personalisation v.0.1.X, v.0.2.0

All the released versions can be accessed at: <https://github.com/XR2Learn/Enablers-CLI/tags>

Table 11 lists the Command line interface (CLI) released versions and date. A description of the main changes in the project's versions can be found at: <https://github.com/XR2Learn/Enablers-CLI/blob/master/CHANGELOG.md>

---

## 4. DATA ACQUISITION

---

The performance of Machine and Deep Learning models heavily depend on the quality of data used for training and evaluation. Furthermore, whereas Self-Supervised Learning can be effectively used to learn latent representations of different types of data, annotations are crucial for supervised learning and/or fine-tuning these representations and mapping them to outputs the models are intended to recognize.

In Section 3.1, we presented the preliminary research on emotion recognition conducted with open-source data containing speech, bio-measurements and body tracking. According to the analysis conducted, we identified a set of modality-specific and common challenges and limitations arising when developing emotion recognition models. In particular, the issues related to the open available datasets can be briefly summarized as follows:

- Emotion elicitation and data annotation are challenging aspects requiring personalized approaches based on the use case. First of all, there is a lack of available data that have been collected in educational settings. Furthermore, the existing datasets do not utilize emotional models related to education, such as the theory of flow, for annotations. Nevertheless, our enablers require annotated data for training classifiers using engagement-related metrics in order to allow personalization in educational settings. Besides, the experiments on open-source datasets have shown that not all emotion elicitation and annotation strategies are effective enough to provide accurately labeled data.
- As a result of the wide range of sensors and devices that can be used to record bio-measurements and the different methods of recording them, there is no standard format for storing bio-measurements. Therefore, creating a single data processing system that can handle data from multiple sources is challenging.
- Datasets based on body tracking data contain rich information regarding multiple key points in human bodies. Nevertheless, commercial VR equipment allows tracking fewer bodily cues, namely controllers and headset positions.

The identified problems make it feasible to only rely partially on open datasets for building models within Enablers 2-6. Thus, a dedicated data collection tool is needed to overcome these limitations and collect data suitable for training and utilizing models in the implemented enablers.

---

### 4.1. MAGIC XROOM: DATA COLLECTION TOOL

---

#### 4.1.1. Technical Documentation

---

##### 4.1.1.1. Description

The Magic XRoom is a Virtual Reality (VR) application developed in the framework of Task 3.2 to elicit specific emotions and gather data from external sensors through a set of scenarios. The application allows the user to experience four different scenarios composed of increasingly difficult tasks that require various skills to complete within a time limit.

The data collected with the Magic XRoom is organized, annotated and written to CSV files to facilitate its analysis. The sensors compatible with the Magic XRoom are the following:

- Virtual Reality headset and controllers (mandatory)
  - Collect position and rotation.
- Shimmer 3 GSR+ device (optional)
  - Collect position, rotation, acceleration, galvanic skin response (GSR, EDA), photoplethysmography (PPG, BVP), and heart rate (generated from the PPG data).
- Lip and Eye tracking devices (optional)
  - Collect eyes and lip features.

#### 4.1.1.2. Prerequisites

The Magic XRoom is compatible only with Microsoft Windows 10 (or higher) x64 systems.

The minimum requirements in terms of hardware sensors are a Virtual Reality headset and Virtual Reality controllers. For this the software required is:

- SteamVR (downloaded with Steam)
- VR headset linking/streaming software, i.e.
  - VIVE Focus 3 requires VIVE Business Streaming;
  - Oculus Quest requires the Oculus Desktop App.

#### 4.1.1.3. Installation

Depending on which external sensors will be used during the data collection, additional software must be installed and configured prior to launching the Magic XRoom application.

To use a *Shimmer 3 GSR+* device:

- No additional software is required, but the sensor must be configured with the *LogAndStream* firmware;
- Bluetooth 5.1+ availability required;
- The Shimmer sensor must be paired with the computer before starting the application.

*Please refer to the official documentation<sup>70</sup> for a detailed explanation of successfully setting up and pairing a Shimmer GSR+ device.*

To use face/eye tracking:

---

<sup>70</sup>

- SRanipal runtime (included in *VIVE Console for SteamVR* on Steam);
- The computer and the VR hardware must be connected to the same 5GHz WiFi network. The Windows *mobile hotspot* has shown promising results during testing when a WiFi network is unavailable or restricted.

#### 4.1.1.4. Basic User Manual

The actions mapped to the VIVE Focus 3 controllers are the following (mirrored on the left controller):

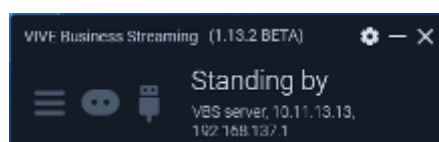


- Figure 28. Magic Xroom handheld controllers actions and buttons.

The following are the recommended steps to successfully launch and use the Magic XRoom with a VIVE Focus 3. Due to the nature of VR hardware/software and their environments, alternative methods might work but it is up to the user to choose which to follow and ensure the software works as intended.




*Note: software versions shown in images might differ*

1. Download the required software and setup the hardware
  - a. Link the VR headset and controllers, pair the Shimmer sensor, connect to a strong WiFi if eye/face tracking is enabled
2. Launch VIVE Streaming Business



- Figure 29. VIVE Business Streaming starting screen.

3. Launch VIVE SRanipal (Steam version) if eye/face tracking is enabled (it launches automatically at point 5. but this step ensures the correct version is used in case multiple versions are installed on the computer)
  - a. Default install directory is  
`<steam install directory>\steamapps\common\VIVEDriver\App\SRanipal`
  - b. When successfully connected with the eye tracking accessory the tray icon should change color (same for mouth tracking).

-  when eye tracking is either disabled or not supported on the attached headset.
-  when eye tracking is idle.
-  when eye tracking is in use.

- Figure 30. Representation of icon colors change indicating the use of eye tracking accessory.

*Note: In the new versions of the software, the icon turns blue when connected, instead of green.*

4. Configure the user settings if necessary (explained further down in this chapter)
5. Launch the Magic XRoom executable (SteamVR starts automatically)

At this point the user should be within the virtual world of the Magic XRoom.

In order to start the data collection one last step is required:



- Figure 31. Magic Xroom sensors panel and data collection start button

The red button shown in the above picture is used to start/stop the data collection. The panel above it shows the current state of the data collection [ **Stopped** / **Running** ] and the current state of the Shimmer device [ **Disconnected** / **Connecting** / **Connected** / **Streaming** / **Inactive** ]. If the Shimmer sensor is used, it's important to wait until it shows a **Connected** state while it's connecting to the PC before starting the data collection and wait for it to turn into the **Streaming** state, otherwise no data will be collected for this sensor.

It is possible to work on multiple sessions without restarting the application by starting/stopping the data collection. Each time a new data collection is started a unique set of files is generated.

---

The Magic XRoom application can be configured with an external file to enable/disable some features or to tweak parameters related to the data collection.

The configuration file can be found in a subfolder of the directory of the executable:  
<Magic Xroom directory>\xr2learn-magicxroom\_Data\StreamingAssets\UserSettings.xml

For all sensors that require **samplingRate** or **samplingBufferSize**, to calculate the delay  $\Delta t$  in [s] between each write operation for a specific sensor use the following formula:

$$\Delta t = \frac{\text{samplingBufferSize}}{\text{samplingRate}}$$

Higher values will impact less on performance but will result in more data being lost in case of abrupt interrupts in the data collection.

Example: with a **samplingBufferSize** of 50 and a **samplingRate** of 10Hz the resulting delay will be 5s, meaning that at most 5s of data might be lost in case of unexpected errors or problems with the application/hardware/connection.

The UserSettings.xml file contents are the following:

- **keyboard**

- **enableShortcuts** [true/false] => enables/disables the following keyboard shortcuts (the application window must be focused to receive keyboard inputs):
  - **F** => manually shows the Feedback Board
  - **D** => toggles (start/stop) the data collection

- **dataCollection**

- **autoStart** [true/false] => enables/disables the automatic start of the data collection when the application starts.
- **outputPath** [Windows compatible directory] => specifies the location where the data collection files will be generated. If left empty the default location is the same as the executable file. Ensure read/write operations are allowed for the current user

- **vr**

- **config**
- **samplingRate** [integer] => the frequency in [Hz] at which the application will sample data from the VR headset and controllers
- **samplingBufferSize** [integer] => the number of samples buffered before writing to file

- **shimmer**

- **enabled** [true/false] => enable/disable the Shimmer sensor data collection
- **deviceName** [any] => the Shimmer device internal name
- **config**
  - **heartbeatsToAverage** [integer] => the number of heartbeat inputs used to calculate an average. Higher values tend to generate better results but sometimes break due to the volatility of the Bluetooth communication, i.e. if the value is set to 10 and one of the 10 inputs is corrupted or invalid, the overall average will be invalid for the next 10 calculations.



- **trainingPeriodPPG** [integer] => the delay in [s] before the PPG signal is able to calculate a heartbeat
- **samplingRate** [integer] => the frequency in [Hz] at which the application will sample data from the Shimmer sensor
- **samplingBufferSize**<sup>3</sup> [integer] => the number of samples buffered before writing to file.
- **sensors**
  - **enableAccelerator** [true/false] => enables/disables the Shimmer internal accelerator sensor
  - **enableGSR** [true/false] => enables/disables the Shimmer internal GSR sensor
  - **enablePPG** [true/false] => enables/disables the Shimmer internal PPG sensor
- **eyeTracking**
  - **enabled** [true/false] => enables/disables eye tracking
  - **config**
    - **samplingRate** [integer] => the frequency in [Hz] at which the application will sample data from the eye tracking sensor
    - **samplingBufferSize** [integer] => the number of samples buffered before writing to file.
- **faceTracking**
  - **enabled** [true/false] => enables/disables face tracking
  - **config**
    - **samplingRate** [integer] => the frequency in [Hz] at which the application will sample data from the face tracking sensor
    - **samplingBufferSize** [integer] => the number of samples buffered before writing to file.
- **feedback**
  - **enabled** [true/false] => enables/disables the feedback feature
  - **afterScenario** [true/false] => show the feedback panel at the end of a scenario
  - **afterLevel** [true/false] => show the feedback panel at the end of each level

Note that in case the `UserSettings.xml` file is not found in the specified directory or in case of any erroneous or missing data, the application will use any or all the default values shown in the below image and in the original file provided with the executable.

It is advised to modify the `UserSettings.xml` file and disable any sensors not in use to avoid unnecessary burden on the application.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<settings>
  <keyboard enableShortcuts="true" />
  <dataCollection autoStart="false" outputPath="" />
  <vr>
    <config samplingRate="10" samplingBufferSize="50" />
  </vr>
  <shimmer enabled="true" deviceName="Shimmer3">
    <config heartbeatsToAverage="1" trainingPeriodPPG="10" samplingRate="10" samplingBufferSize="50" />
    <sensors enableAccelerator="true" enableGSR="true" enablePPG="true" />
  </shimmer>
  <eyeTracking enabled="true">
    <config samplingRate="10" samplingBufferSize="50" />
  </eyeTracking>
  <faceTracking enabled="true">
    <config samplingRate="10" samplingBufferSize="50" />
  </faceTracking>
  <feedback enabled="true" afterScenario="true" afterLevel="false" />
</settings>
```

- Figure 32. UserSettings.xml file with the supported values.

The output of the Magic XRoom is a set of files containing the data collected by the sensors enabled prior to launching the application.

Each of the enabled sensors will generate a comma-separated values (CSV) file sharing a unique identifier for the session, using the following naming convention:

```
data_collection_<session ID>_<sensor type>.csv
```

The session ID is generated as a number referring to the number of ticks since midnight 01.01.0001. Each tick represents 100 nanoseconds and to retrieve the corresponding date, input the number in an epoch converter (compatible with C# DateTime).

## VR

```
data_collection_<session ID>_VR.csv
```

Contains the data collected from the Virtual Reality Headset and Controllers. The columns represent the position and rotation of the headset and controllers:

- timestamp => application timestamp
- head\_pos\_x => headset absolute position x axis
- head\_pos\_y => headset absolute position y axis
- head\_pos\_z => headset absolute position z axis
- head\_rot\_x => headset absolute rotation x (quaternion)
- head\_rot\_y => headset absolute rotation y (quaternion)
- head\_rot\_z => headset absolute rotation z (quaternion)
- head\_rot\_w => headset absolute rotation w (quaternion)
- lcontroller\_pos\_x => left controller absolute position x axis
- lcontroller\_pos\_y => left controller absolute position y axis
- lcontroller\_pos\_z => left controller absolute position z axis
- lcontroller\_rot\_x => left controller absolute rotation x (quaternion)
- lcontroller\_rot\_y => left controller absolute rotation y (quaternion)
- lcontroller\_rot\_z => left controller absolute rotation z (quaternion)
- lcontroller\_rot\_w => left controller absolute rotation w (quaternion)
- rcontroller\_pos\_x => right controller absolute position x axis
- rcontroller\_pos\_y => right controller absolute position y axis
- rcontroller\_pos\_z => right controller absolute position z axis
- rcontroller\_rot\_x => right controller absolute rotation x (quaternion)
- rcontroller\_rot\_y => right controller absolute rotation y (quaternion)
- rcontroller\_rot\_z => right controller absolute rotation z (quaternion)
- rcontroller\_rot\_w => right controller absolute rotation w (quaternion)

## Shimmer

```
data_collection_<session ID>_SHIMMER.csv
```

Contains the data collected from the Shimmer device. The columns represent the values captured by the Shimmer sensors

- timestamp => application timestamp
- int\_timestamp => Shimmer internal timestamp
- accel\_x => Shimmer accelerator x axis
- accel\_y => Shimmer accelerator y axis

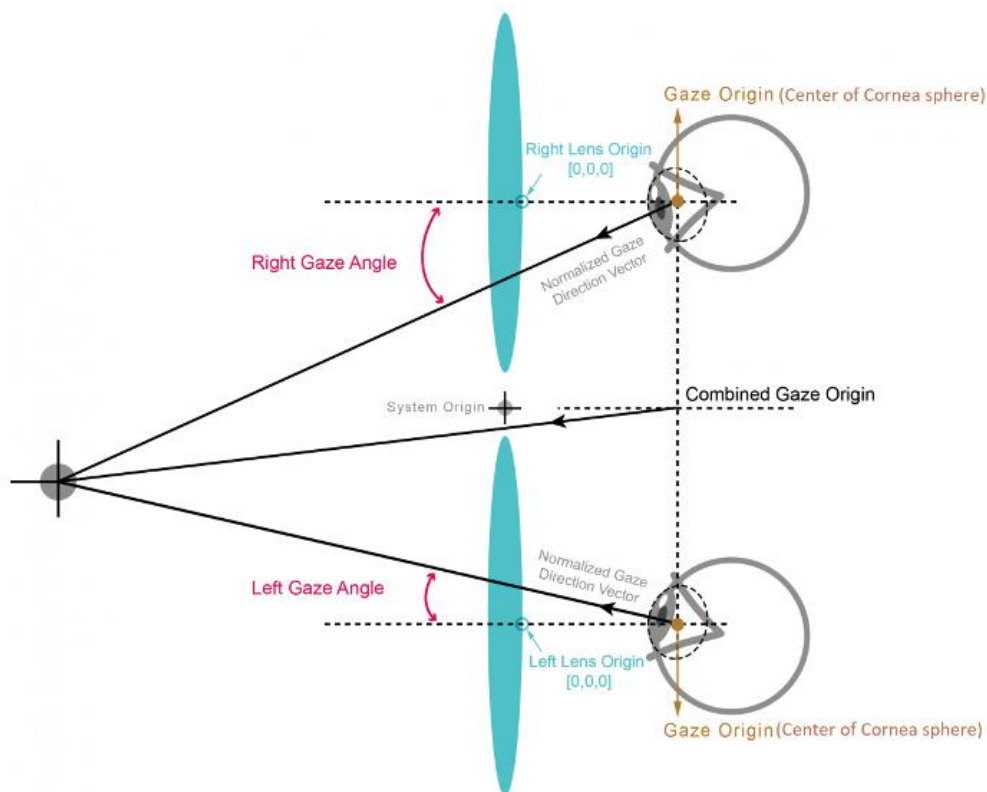
- `accel_z` => Shimmer accelerator z axis
- `gsr` => Shimmer Galvanic Skin Response (GSR) sensor
- `ppg` => Photoplethysmograph (PPG) sensor
- `hr` => heart rate computed from the PPG data

### Eye Tracking

`data_collection_<session ID>_EYE.csv`

Contains the data collected from the Eye Tracking device (vectors are right-handed). The columns represent the gaze, pupil and position of each eye

- `timestamp` => application timestamp
- `int_timestamp` => Eye Tracking device internal timestamp
- `left_gaze_origin_x` => left eye x cornea center relative to each lens center [mm]
- `left_gaze_origin_y` => left eye y cornea center relative to each lens center [mm]
- `left_gaze_origin_z` => left eye z cornea center relative to each lens center [mm]
- `left_gaze_dir_norm_x` => left eye gaze x direction normalized [0,1]
- `left_gaze_dir_norm_y` => left eye gaze y direction normalized [0,1]
- `left_gaze_dir_norm_z` => left eye gaze z direction normalized [0,1]
- `left_pupil_diameter` => left eye pupil diameter in [mm]
- `left_eye_openness` => left eye openness (0 closed, 1 open)
- `left_pos_norm_x` => normalized left eye pupil x pos relative to lenses (0.5,0.5 is center)
- `left_pos_norm_y` => normalized left eye pupil y pos relative to lenses (0.5,0.5 is center)
- `right_gaze_origin_x` => right eye x cornea center relative to each lens center [mm]
- `right_gaze_origin_y` => right eye y cornea center relative to each lens center [mm]
- `right_gaze_origin_z` => right eye z cornea center relative to each lens center [mm]
- `right_gaze_dir_norm_x` => right eye gaze x direction normalized [0,1]
- `right_gaze_dir_norm_y` => right eye gaze y direction normalized [0,1]
- `right_gaze_dir_norm_z` => right eye gaze z direction normalized [0,1]
- `right_pupil_diameter` => right eye pupil diameter in [mm]
- `right_eye_openness` => right openness (0 closed, 1 open)
- `right_pos_norm_x` => normalized right eye pupil x pos relative to lenses (0.5,0.5 is center)
- `right_pos_norm_y` => normalized right eye pupil y pos relative to lenses (0.5,0.5 is center)



- Figure 33. Representation of normalized gaze direction vectors.

## Face Tracking

data\_collection\_<session ID>\_FACE.csv

Contains the data collected from the Face Tracking device (vectors are right-handed). The columns represent 27 facial points/features and how much these points are influencing the resulting facial expression (some are self-explanatory)

- timestamp => application timestamp
- int\_timestamp => Face Tracking device internal timestamp
- none => no difference compared to the default shape
- jaw\_forward => jaw position on the forward axis
- jaw\_right => jaw position on the right side of the horizontal axis
- jaw\_left => jaw position on the left side of the horizontal axis
- jaw\_open => jaw openness
- mouth\_ape\_shape => mouth aperture shape
- mouth\_o\_shape => mouth O shape (i.e. while making an “O” sound)
- mouth\_pout => mouth pouting shape
- mouth\_lower\_right => mouth lower right shift
- mouth\_lower\_left => mouth lower left shift
- mouth\_smile\_right => mouth smile shape right side
- mouth\_smile\_left => mouth smile shape left side
- mouth\_sad\_right => mouth sad shape right side
- mouth\_sad\_left => mouth sad shape left side
- cheek\_puff\_right => cheek puff shape right side
- cheek\_puff\_left => cheek puff shape left side
- mouth\_lower\_inside => mouth inside lower shape

- `mouth_upper_inside` => mouth inside upper shape
- `mouth_lower_overlay` => mouth inside lower overlay
- `mouth_upper_overlay` => mouth inside upper overlay
- `check_suck` => cheek “suck” expression
- `mouth_lower_right_down` => mouth lower right down shift
- `mouth_lower_left_down` => mouth lower left down shift
- `mouth_upper_right_up` => mouth upper right up shift
- `mouth_upper_left_up` => mouth upper left up shift
- `mouth_philtrum_right` => mouth philtrum right shape
- `mouth_philtrum_left` => mouth philtrum left shape

### Progression Events

`data_collection_<session ID>_PROGRESS_EVENT.csv`

Contains the events representing the user interaction with the environment and the scenarios. The columns represent the event type and the information relative to that event

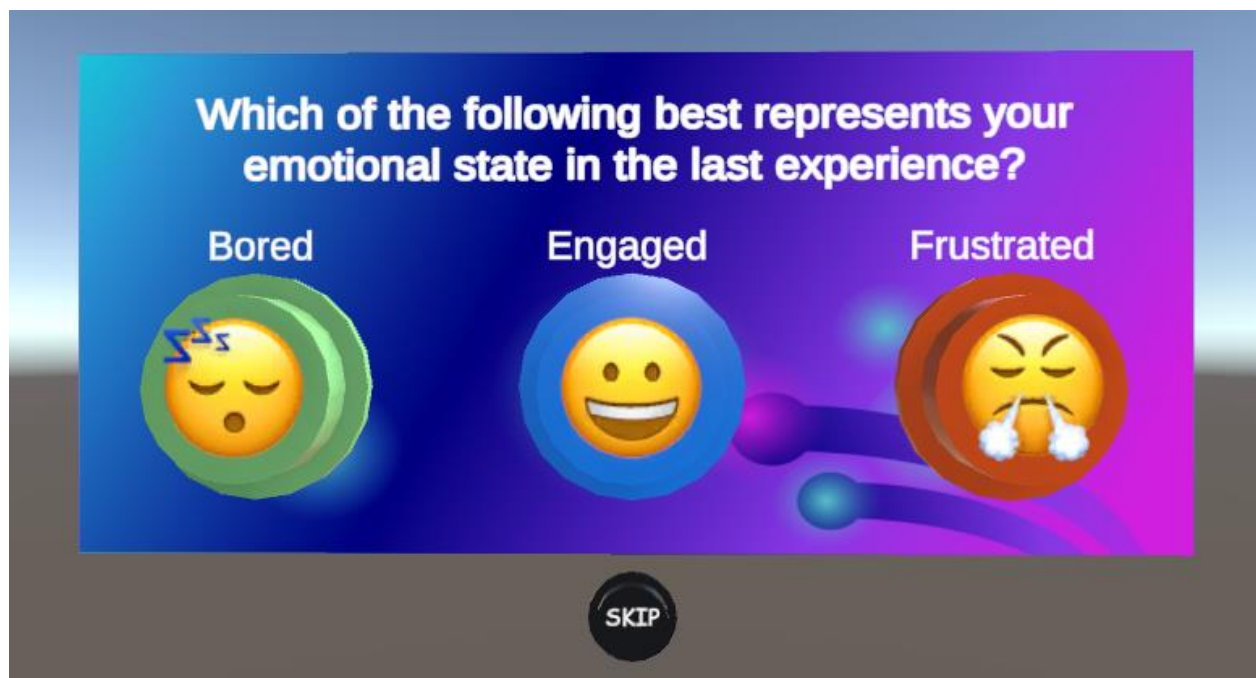
- `timestamp` => application timestamp
- `event_type` => event type
- `info` => information relative to the event

The events generated fall in the following categories:

- **Scenario** => a scenario can either be started manually by the user, or is ended by completing all the levels or teleporting outside the scenario area. The info column represents the name of the scenario.
  - `SCENARIO_STARTED` => a scenario is manually started
  - `SCENARIO_ENDED` => a scenario is ended, either by competition or by leaving the scenario area
- **Level** => a scenario is composed of one or more levels. A level can be started, completed or failed. The info column represents the level difficulty as a number depending on the specific scenario.
  - `LEVEL_STARTED` => a level is started
  - `LEVEL_FAILED` => a level is failed
  - `LEVEL_COMPLETED` => a level is completed successfully
- **Teleport** => the user uses a teleport feature to enter or exit a scenario area. The info column represents the name of the scenario.
  - `TELEPORT_IN` => the user was outside a scenario area and has teleported inside a scenario area
  - `TELEPORT_OUT` => the user was inside a scenario area and has teleported outside a scenario area
- **Feedback** => the user can provide feedback on his/her emotional state in the latest scenario or level. The info column represents the time in [ms] that the user waited before making a decision. The SKIP option is used for exceptional situations, i.e. the user entered a scenario area by mistake and exited it right away without starting a level.
  - `BORED` => the user was bored
  - `ENGAGED` => the user was engaged
  - `FRUSTRATED` => the user was frustrated
  - `SKIP` => the user skipped the selection
- **Shimmer** => in order to facilitate the synchronization between the Shimmer device and the other sensors, the Shimmer internal timestamp is added when a scenario is started/stopped

Note that the value of `timestamp` written in each of the files just mentioned is synchronized and taken from the same context. Although the application itself runs mainly on one thread, some of the sensors might have a delay in reading or providing data. When available, refer to both the `timestamp` and `int_timestamp` values to understand if the delay between the polling of data from a sensor and the writing to file operation should be taken into consideration or if it's negligible.

The Magic XRoom provides a system to request the user for their emotional state (self-annotations) at specific moments throughout the experience.



- Figure 34. User feedback screen in the Magic XRoom, including the three emotions according to the Theory of Flow

The Feedback Panel contains four interactable buttons with the following results:

- **Bored:** the experience is not providing engagement comparable to the user skill level
- **Engaged:** the experience is engaging for the user, the difficulty is correctly balanced for the user skill level
- **Frustrated:** the user is experiencing a frustration or anxiety because of a gap between their skill level and the experience difficulty
- **Skip:** the user has the option to not answer

The panel is shown after each level/scenario depending on the User Settings, but can also be toggled manually by pressing the `F` key on the keyboard.

As mentioned before, the Magic XRoom contains four scenarios meant to elicit specific emotions and gather data from external sensors. Each experience is composed of increasingly difficult tasks that require various skill levels to complete before a given time limit.

Some of the experiences are positioned on desks that can be adjusted in height with a handle.





- Figure 35. Desk handle for level adjustment

### Stacking Cubes

Stacking Cubes is an experience in which the user is tasked with positioning 4 cubes on top of each other within a time limit. The cubes properties vary between levels:

1. The cubes have no particular property and are the same size
2. The cubes are slippery
3. The cubes are bouncy
4. The cubes are of different size and must be stacked from small to big
5. The cubes are the same size but an external force (wind) is making it difficult to stack them straight

Failed levels must be repeated until successfully completed.



- Figure 36. Initial screen of stacking game scenario

### Color Words

Color Words is a fast-paced experience in which the user is tasked to select (touch) one of the cubes on the desk depending on its color. The correct color is shown on the screen as a word describing the color but coloured with a different one. For example,



if the word *yellow* appears in *red* color, the user must select a yellow cube, not a red one. Multiple cubes of the same color might appear.

As the game progresses the number of cubes available increases and the time available to make a decision decreases, until the user chooses the wrong color or there is not enough time to select a cube and the experience ends.



- Figure 37. Color words scenario screen when the time is out

## Canvas Painting

Canvas Painting is a painting minigame where the user is tasked with drawing a specific shape without exiting a given area. A limited number of mistakes are allowed.

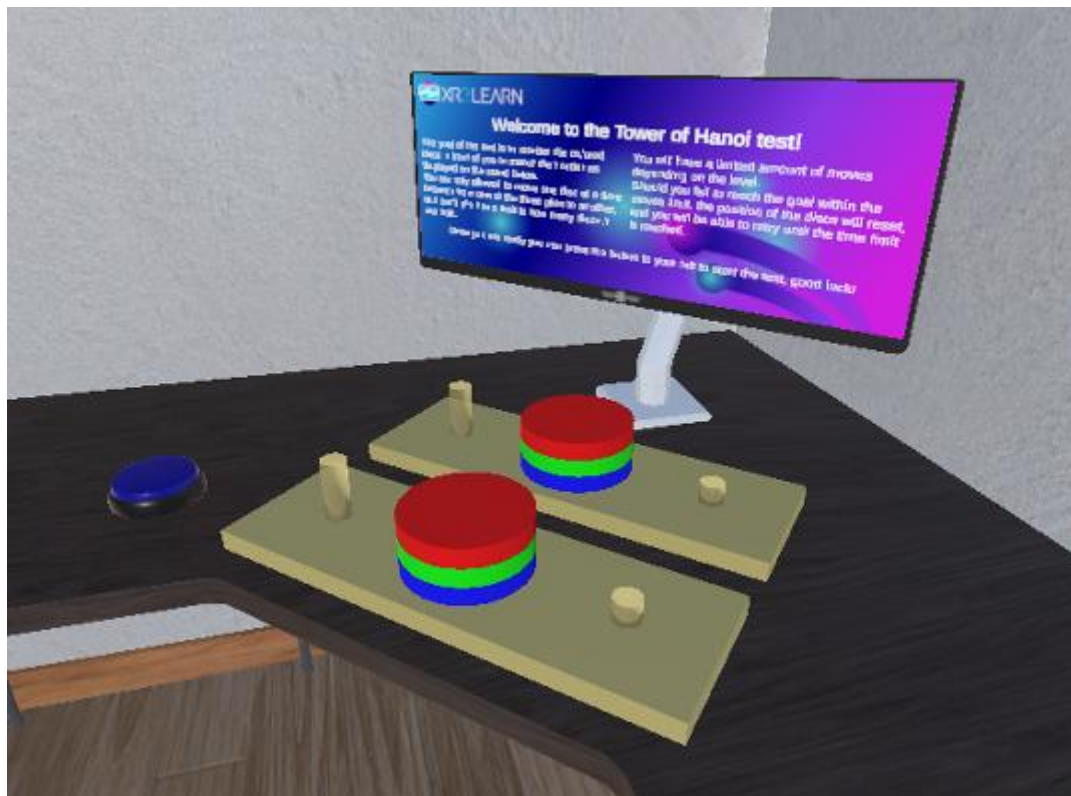


- Figure 38. Canvas painting scenario instructions screen

## Tower of Hanoi

Tower of Hanoi is a relatively famous game. The version used in the Magic XRoom is composed of 3 rods and 3 discs. The rods are of different sizes: the left-most can hold 3 disks at the same time, the middle one 2, and the right-most only 1.

The user is presented with two sets of rods and disks. The set closer to the user is the interactive one (interaction set), while the other one represents the target configuration (configuration set). At the beginning of each level the configuration set shows the target configuration and the user must try and match it with the disks in the interaction set. The moves available are limited (depending on the level and difficulty) and a time limit is displayed on the screen.



- Figure 39. Tower of Hanoi scenario instructions screen

### 4.1.1.5. Open-source Code

The latest version of the Magic Xroom and its source code can be found at:

<https://github.com/XR2Learn/magic-xroom>

- Table 12. Magic XRoom versions and their release dates.

Version	Release date
---------	--------------

<b>v0.1.0</b>	<b>2023-11-07</b>
<b>v0.2.0</b>	<b>2023-11-13</b>
<b>v0.2.1</b>	<b>2023-11-16</b>
<b>v0.2.2</b>	<b>2023-11-20</b>
<b>v0.3.0</b>	<b>2023-11-30</b>
<b>v0.4.0</b>	<b>2024-01-18</b>
<b>v1.0.0</b>	<b>2024-02-12</b>
<b>v1.1.0</b>	<b>Planned for 2024-02-29</b>

**v1.0.0** is the result of continuous improvements and feedback from the January pilot. **v1.1.0** is planned for the submission of this document as a public release. The project contents will be cleaned and formatted following sector standards in order to facilitate its use and understanding.

#### 4.1.1.6. Known Issues

The following is a comprehensive list of known issues and potential bugs related to the Magic Xroom as of the publication of this document.

##### **SRanipal**

SRanipal serves as the runtime environment, enabling interaction with the Vive Facial Tracker and related Vive eye/face tracking hardware on Windows PCs. It is currently the only option for the Vive Focus face and eye accessories. A review of the existing online resources, including documentation, support, and code, indicates that SRanipal is not yet in a finalized release state.

To provide a better understanding of the current state of the SRanipal framework, the following are a few examples of the many issues found during development which one would usually not expect from this type of software:

- Several recent versions were tested, and they presented unnecessarily use and allocation of huge portions of the computer resources;
- In the latest versions, if left unchecked, the framework quickly generates gigabytes of data in log files. This feature cannot be turned off;
- Launching the SRanipal runtime triggers the launch of additional processes which seem unnecessary and use a significant amount of system resources;
- If the installation process fails, which has been shown to occur frequently, the uninstaller fails to entirely cleanse the system, leaving it in an intermediate state. Attempting to reinstall the software consequently triggers a well-known error, requiring manually deleting specific system registry keys with administrative privileges to fix the issue.

The Magic Xroom is significantly dependent on the data generated by this framework, which frequently appears inconsistent or missing. Throughout the initial data collection trial, several specific behaviors were noted, leading to the compilation of a list of guidelines aimed at preventing or reducing issues associated with data obtained through this framework:

- Before initiating a data collection session, verify the functionality of all sensors through a preliminary simulation.

- Adjust the headset to fit the user's facial structure accurately:
  - Calibration of the eye tracker should start only after the headset has been correctly positioned;
  - It is essential to position the headset in a manner that does not obstruct any portion of the mouth area, as visibility of this region is crucial for the face tracker's effectiveness.

It is important to note that the version of SRanipal installed on the computer can have a significant impact on the outputs of Magic Xroom. Currently, Magic Xroom only supports the Steam SRanipal version, which should be the only version installed on the machine. Using multiple SRanipal versions on a single computer can cause interference issues between them and affect the functionality of Magic Xroom.

We will closely monitor future releases of this framework and evaluate potential upgrades of the version used for the Magic Xroom.

### **VR tracking area**

For consistent performance with the Vive Focus headset, the user must stay within the predefined tracking zone. Should the controllers or headset move beyond the boundaries of this area, updates to their positional and rotational data might cease, and this interruption will persist even upon re-entering the designated tracking zone.

The best solution currently is to monitor the user's movements during data collection strictly.

### **Unity physics engine**

During the operation of the Magic Xroom application, it has been documented that removing the headset activates a 'low performance' mode, resulting in a reduced application refresh rate. This adjustment adversely impacts the Unity physics engine, as it may cause frames to be skipped or not processed within the anticipated timeframe. Consequently, objects within the virtual reality environment can unexpectedly accelerate to excessive velocities, potentially colliding with other objects and disrupting specific scenarios.

The recommended corrective action, in situations where the physics within the virtual environment break, is to restart the application and resume from the point before the disruption.

Further testing of the Unity layers and colliders systems is necessary to understand what triggers these situations and to develop ways to prevent them.

### **Vive Focus eye tracker**

The Vive Focus eye tracker (Figure 40) is positioned between the headset lenses and the padding. Despite its slim profile, it introduces a gap that causes visual blurriness. Adjustments to the headset position or interpupillary distance offer minimal improvement. This issue has been widely reported by users as the cause of diminished virtual reality immersion, complicated text readability, and impaired depth perception. Furthermore, the accuracy of data collected from users who wear thick prescription glasses or glasses with wide frames is notably compromised, leading to data loss or inconsistent outcomes from the eye tracker.



- Figure 40. Vive Focus eye tracker.

## 4.2. DATA COLLECTION

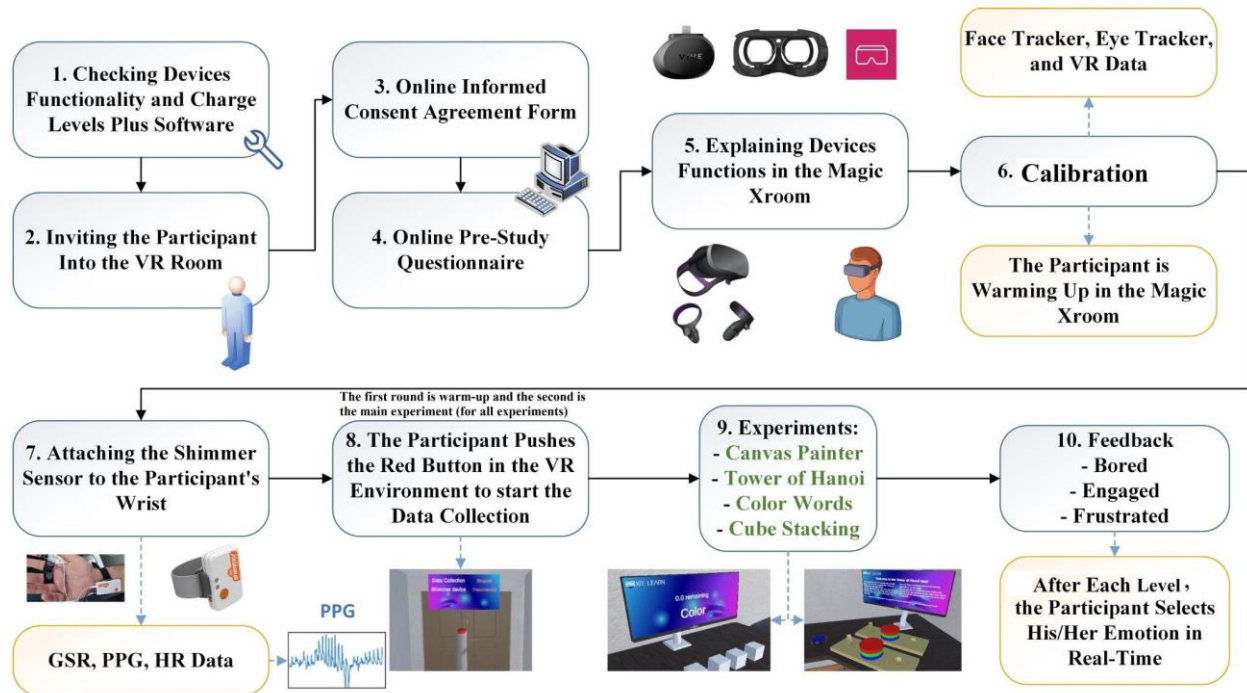
### 4.2.1. Data Collection Protocol for Magic XRoom

The main objective of creating a data collection protocol<sup>71 72</sup> is to have a unified approach that can be exploited by a party that collects data using Magic XRoom. This would ensure homogeneity across participants, make data processing and analysis more convenient, improve data readability, and reduce possible outliers. The proposed data collection protocol uses the power of Virtual Reality (VR) technology to create a framework for studying human emotions. By combining a VR headset with hand-held controllers, participants are immersed in a stimulating Magic XRoom environment where their emotional responses to puzzle-solving tasks are precisely recorded. Incorporating the Shimmer sensor to capture Galvanic Skin Response (GSR) and PhotoPlethysmoGram (PPG) data is very valuable, as it directly measures physiological changes associated with emotional states. Moreover, adding eye and face tracking sensors mounted on the VR headset enriches the data by capturing subtle eye and facial movements of various emotions. For the technical documentation concerning Magic XRoom, please refer to Section 4.1.1.

<sup>71</sup> Costa, Jean, et al. "Boostmeup: Improving cognitive performance in the moment by unobtrusively regulating emotions with a smartwatch." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 3.2 (2019): 1-23.

<sup>72</sup> Gasparini, Francesca, et al. "Personalized PPG Normalization Based on Subject Heartbeat in Resting State Condition." Signals 3.2 (2022): 249-265.





- Figure 41. Data collection protocol diagram.

We illustrate the suggested data collection protocol as a diagram in Figure 41. Furthermore, the step-by-step protocol for a data collection session can be summarized as follows:

1. Ensure that the VR headset and both hand controllers are charged, running, and connected to the system. Additionally, all required softwares must be run before starting the Magic XRoom. All these steps help avoiding data loss.
2. Invite the participant to the data collection space.
3. Provide participants with the **informed consent agreement** form and require them to read it carefully. This form is expected to be filled online and on paper for archiving purposes.
  - a. This form explains the purpose of this research study, study procedures, time required for the whole experiment, risks and benefits, data confidentiality, data usage, possibility of data withdrawal by the participant, and participants' contact information. More specifically, The informed consent document explains data modalities that will be collected during sessions which shortly are physiological (GSR, PPG), VR, eye tracking, and face tracking emotional data via wearable sensors during a VR puzzle-solving task. The session lasts about 30 minutes with minimal associated risks. The document emphasizes the confidentiality of the data collected and the anonymity of participants. It also highlights compliance with data protection laws (GDPR). Participation is voluntary, and participants can withdraw at any time without consequences, including the option to have their data removed. The document also provides contact information for the study organizers and includes a section for participants to acknowledge their consent.
  - b. The consent forms can be created and hosted online using dedicated platforms compliant with GDPR and offering secure data storage and

encryption of data. The examples of such platforms are “jotform”<sup>73</sup> and “Qualtrics”<sup>74</sup>.

4. Ask participants to fill the second form, namely the **pre-study questionnaire**, that collects demographic information and metadata. Specifically, it collects basic demographic information such as age, gender, education, occupation, and/or ethnicity. Participants are also asked about their proficiency with computers and electronic devices, previous experience with VR, and familiarity with VR technology. The form inquires about the last time they experienced VR and if they have any conditions like motion sickness or anxiety that might affect their VR experience. Additionally, participants are asked about any recent medications and their main motivation for participating in the study. Finally, there's a section for participants to confirm their understanding and agreement to proceed with the VR experiment. This form can also be hosted online.
5. The participant is demonstrated how handheld controllers and VR headset work in the Magic XRoom. This step is done by the person responsible for collecting the data.
  - a. A tutorial is conducted by the dataset collector to avoid confusion during the execution of the experiments. The tutorial includes a brief description of the devices functionality, the navigation in the VR environment and the functioning of the puzzles.
  - b. The dataset collector wears all VR equipment and explains each and every part of the virtual environment including puzzles (one by one), how to teleport in the virtual environment, feedback system, and interaction buttons on the handheld controllers. This process continues until the dataset collector is sure that the participant is aware of the expected previous knowledge from the participant. However, during the experiment, participants will be guided by the dataset collector during the whole process if needed.
6. Adjust the VR headset (including face and eye tracker) on the participant's head and give them handheld controllers, in which they enter the Magic XRoom. Before starting the data collection, the user can take some time to familiarize them with the system.
7. The next step is to attach the Shimmer sensor to the subject's wrist of the non-dominant hand in which GSR (recording EDA) sensors will be attached to the index finger and the finger next to it. Also, the PPG sensor (recording BVP) will be attached to the ring finger. The sensor must be tightened on the fingers and wrist in order to collect more accurate data. The participant could continue familiarizing with the system if needed.
8. Participants start data collection. Magic XRoom offers 4 interactive games, also referred to as scenarios, that should be completed in the order specified in the next step. Before/after each scenario, a participant should press a red button next to the “Tower of Hanoi” scenario to start/finish data collection. Detailed instructions:
  - Participants can teleport to the experiments/scenarios and start interacting with them.

---

<sup>73</sup> <https://eu.jotform.com/>

<sup>74</sup> <https://www.qualtrics.com/>



- Each experiment has a table (except the “Canvas Painter”) which has a blue button in its left corner. By pushing the blue button, the experiment starts.
  - Each scenario is suggested to be completed in two stages:
    - a. **Warming up:** makes participants familiar with the scenario.
    - b. **Main trial:** collect data when the participant is familiar with the experiment environment.
9. The suggested order of the games/scenarios is presented below. Overall, 4 (without warming up) or 8 (with warming up) recordings are collected within a single session with one participant.
- a. **Canvas Painter**
  - b. **Tower of Hanoi**
  - c. **Color Words**
  - d. **Cube Stacking**
10. After finishing each level in each scenario, the participant will be asked to provide self-annotation on the emotion experienced. As previously mentioned, the options that can be selected are “Bored”, “Engaged” and, “Frustrated” (based on flow theory) and “Skip”.

---

#### 4.2.2. Data Collection Pilots

---

In January 2024, the first data collection pilots using Magic XRoom were conducted at SUPSI and UM premises. These pilots involved 31 subjects and were conducted after the consent forms and questionnaires were approved by SUPSI's and UM's ethics committees. The main objective of these pilots was to test Magic XRoom's performance with users and identify any issues with the tool, which has been used to improve Magic XRoom functionalities and performance. The data collected during the pilots has been used to establish the format of bio-measurement data and develop pre-processing pipelines within the emotion recognition enablers. The brief statistics of these pilots are summarized in Table 13.

- Table 13. Data collection pilots statistics.

<b>Dates</b>	<b>Location</b>	<b>Number of participants</b>	<b>Purpose</b>
January-February 2024	SUPSI	20	Testing Magic XRoom
January 2024	UM	13	Testing Magic XRoom, obtain data input format to implement enabler components for bio-measurement modality

---

## 5. CONCLUSION

---

This document describes the progress achieved in Task 3.2, "XR2Learn Enablers", during the first 14 months of the XR2Learn project, which is part of XR2Learn Phase B, as described in Section 1.2.2 of the proposal document. Task 3.2 is focused on designing, implementing, and delivering novel enablers for XR applications, and during the first sub-phase, the enablers' specification and development have been carried out. The report also contains information on the additional components and functionalities that were developed beyond the project proposal enablers. These include Magic XRoom, a data collection tool, a command line interface to make the enablers more user-friendly, and a graphical user interface demo to demonstrate the communication between Personalization tools (Enabler 6), Inference tools, and XR Unity applications. All tools described were developed following established software engineering practices to foster principles of open science, software sustainability and quality. They will also be available as open-source repositories on GitHub.

According to the project workplan, the second sub-phase will focus on improving and integrating enablers with applications. As per the plan, we will undertake actions to improve and expand the functionality of enablers and integrate them with beacon application(s), demonstrating, this way, how to integrate enablers with other applications, including the open-call projects. The improvement effort will focus on expanding the supported modalities (e.g., bio-measurements, body-tracking) and including additional functionalities and trained models. Meanwhile, the integration effort will focus on building an end-to-end system, including data input capture and pre-processing components, Inference tools, Personalization tools, and a Unity application. These improvements and integration progress for the enablers will be presented in the second version of the deliverable in Month 26 of the project.