Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Skeleton-based online action recognition using inertial sensors in industrial scenarios

Master Candidate

Riccardo Vendramin

Student ID 2087923

Supervisor

Prof. Stefano Ghidoni

University of Padova

Co-supervisor

Prof. Matteo Terreran

University of Padova

Academic Year
2023/2024

Abstract

Online action recognition is crucial in industrial settings to boost productivity, enhance safety, and streamline automation. This research leverages inertial sensors attached to the body to generate precise skeleton representations, facilitating real-time recognition of complex actions performed in industrial environments. The primary goal is to accurately distinguish between actions that exhibit high interclass similarity across different tasks and significant intraclass variety across different subjects. We employ the InfoGCN++ neural network, known for its effectiveness in online skeleton-based action recognition. Our experiments were conducted on both the AnDy dataset, which includes industrial actions like pick, place, and carry, and a custom dataset which simulate the assembly of a chair, with actions such as screwing and place components. The model demonstrated the ability to learn actions well and generalize effectively to new subjects. However, it struggled to generalize to new assembly process scenarios. Fine-tuning the model improved its performance, enabling it to better differentiate between similar actions in these new environments. These findings demonstrate the potential of combining inertial sensors with advanced neural networks for precise, real-time action recognition in industrial settings, offering practical applications for human action monitoring, ergonomics, and process optimization.

Contents

List of Figures xi

List of Tables xiii

List of Algorithms xvii

List of Code Snippets xvii

List of Acronyms xix

1 Introduction 1

2 Literature review 7

2.1 Offline skeleton-based action recognition 8

2.2 Online skeleton-based action recognition 10

2.3 Ergonomics assessment 11

3 Skeleton-based action recognition methodology 15

3.1 Skeleton model 16

3.2 Model 19

3.2.1 INFOGCN++ 19

3.2.2 Model customizations 29

3.3 Ergonomics assessment 32

4 Experiments and results on AnDy dataset 35

4.1 Dataset 36

4.1.1 AnDy Dataset 36

4.1.2 Data rearrangement 37

4.2 Train and Evaluation 40

4.2.1 Results and discussion 42

4.3 Subject cross-validation 44
4.3.1 Results and discussion 44

5 **Experiments and results on custom dataset** **47**
5.1 Custom data collection 49
5.2 Inference on custom dataset using AnDy weights 53
5.2.1 Results and discussion 54
5.3 Inference on custom dataset using AnDy weights after fine tuning 56
5.3.1 Results and discussion 57
5.4 Data augmentation and dataset balancing 60
5.4.1 Results and discussion 61

6 **Conclusions and Future Works** **63**
6.1 Summary 63
6.2 Limitations 64
6.3 Future developments 65
6.4 Final conclusions 66

References **67**

List of Figures

1.1 Skeletal data estimated from rgb camera 2
1.2 Model based on skeletal data measured with inertial sensors . . . 2

2.1 Number of research publications on human action recognition over the decade 2012-2022 [14] 7
2.2 RULA assessment worksheet 13

3.1 A general overview of our work 16
3.2 On the left: musculoskeletal model. It highlights the correspondence between sensors, joints and links. On the right: raw model of skeleton based on joint positions and links 17
3.3 An overview of the InfoGCN++ framework workflow 20
3.4 Example of graph and its adjacency matrix: rows and columns indexes represent nodes. If cell (i,j) is a 1, it means there is an edge between node i and node j 29

4.1 The high-level workflow of the data rearrangement from AnDy to the right format for our model 38
4.2 Confusion Matrix of inference on AnDy dataset 43
4.3 AnDy dataset train and evaluation Area Under the Curve (AUC) 44
4.4 Train and evaluation AUC of cross-validation with subjects 1 and 2 of AnDy 46

5.1 The chair used for our sequence of actions, at an intermediate stage of the assembly process 49
5.2 Experimental setup: in particular this is the initial situation, with one side part on the work table, other components on the left and screws is the red container on the right 49
5.3 Example of side part component of the chair 50

5.4 Example of crossbars component of the chair 50
5.5 Laboratory setup schema for the acquisition of the custom dataset 51
5.6 Examples of variations of reaching action: reaching a screw with one hand, reaching with screwdriver, reaching chair component with two hands. 52
5.7 Examples of variations of placing action: place object with left hand far away from the body, place object with two hands, place object with right hand close to the body. 53
5.8 Examples of variations of screwing action: two hands with horizontal key, two hands with vertical key, one hand 53
5.9 Screwing action in AnDy dataset 54
5.10 Screwing action in our routine 54
5.11 Confusion Matrix of inference on subjects 1 and 2 of custom dataset, evaluated using weights computed after 70 epochs of training on 70% of AnDy 55
5.12 The evolution of overall accuracy over different fine-tuned models. In the y-axis it is shown the accuracy value, in the x-axis the number of subjects used during fine-tuning 58
5.13 The evolution of AUC over different fine-tuned models. In the y-axis it is shown the AUC value, in the x-axis the number of subjects used during fine-tuning 58
5.14 The evolution of accuracy of each label over different fine-tuned models. In the y-axis it is shown the accuracy value, in the x-axis the number of subjects used during fine-tuning 59
5.15 Confusion Matrix of inference on subjects 1 and 2 of custom dataset, evaluated using model fine-tuned on augmented custom dataset 61

List of Tables

3.1 Body parts tracked with a sensor 18

3.2 Segments of the skeleton model 18

3.3 Joints of the skeleton model 18

4.1 Accuracy of each label for AnDy dataset evaluation 43

4.2 Accuracy of cross-subjects validation experiments between the 12 participant of AnDy. 45

5.1 Accuracy of each label for custom dataset evaluation 56

5.2 Initial distribution of samples per label 60

5.3 Final distribution of samples per label after data augmentation . . 60

5.4 Accuracy per label of model fine-tuned on augmented dataset . . 61

List of Algorithms

1 Feeder Data Loading 23

2 Preprocessing 25

3 Training Procedure for Skeleton-based Action Recognition 26

4 Real-Time Action Recognition 32

5 Dataset creation 38

List of Code Snippets

3.1 Freezing layers for fine tuning 31
4.1 Snippet of example of annotation file 39
4.2 Command to train model on AnDy dataset 40

List of Acronyms

HAR Human Action Recognition

IMU Inertial Measurement Unit

RNN Recurrent Neural Network

CNN Convolutional Neural Network

GCN Graph Convolutional Network

STGCN Spatio-Temporal Graph Convolutional Network

AGCN Adaptive Graph Convolutional Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

SA-GC Self-Attention Graph Convolution

WMSDs Work-related musculoskeletal disorders

AUC Area Under the Curve

ROC Receiver Operating Characteristic

EAWS European Assembly Worksheet

RULA Rapid Upper Limb Assessment

OWAS Ovako Work Posture Analysis System

JSA Job Safety Analysis

PERA Postural Ergonomic Risk Assessment

OSHA Occupational Safety and Health Administration

HMM Hidden Markov Model

ROS Robot Operating System

GRU Gated Recurrent Unit

SGN Semantic-Guided Network

HCN Hierarchical Co-occurrence Network

AUC Area Under the Curve

1

Introduction

Action recognition is the task of automatically analyzing, recognizing and classifying human movements from various data inputs. This research field of action recognition has gained significant traction in recent years due to its wide range of applications. Nowadays Human Action Recognition (HAR) is exploited in surveillance to detect dangerous movements [1], healthcare [2] and rehabilitation [3], human-robot interaction and collaboration [4] and smart environments [5]. Early methods developed to address this task relied heavily on visual data [6], utilizing advancements in computer vision to detect and classify actions from video footage. Traditional techniques focused on handcrafted features [7] [8], experimenting with different features and approaches to represent actions in order to enhance performances [9]. The advent of deep learning has revolutionized the field, enabling more robust and accurate recognition methods through convolutional and recurrent neural networks [10]. In parallel with the exploitation of new deep learning architectures and the use of machines with increasingly greater computing power, researchers explored different data modalities, such as skeleton, depth cameras, infrared cameras, point cloud, event stream, audio, accelerations of body parts measured through sensors, and radar [11]. Generally, visual modalities are very effective for HAR. Among them, RGB video data is the most common data type, which has been widely used in surveillance and monitoring systems, but it is limited from suffering from occlusions, poor lighting conditions, and privacy concerns. Skeleton data, which encodes the positions, and so trajectories, of human body joints, is feasible and efficient for HAR when the action performing does not involve objects or scene context. Another advantage of skeletons is that they could be estimated from video but also computed from other sensors, in particular inertial sensors. They offer data richness, providing robust measurements useful to compute accurate skeleton representation, which are unaffected by environmental factors - like lighting condition - and ensure user privacy. In contracts, using cameras is faster, cheaper and require shorter setup time, but skeleton data based on vision suffers from view occlusions, lighting conditions, and the 3D data is an estimate, which leads to less precise measurements. However, inertial sensors come with their own set of challenges too, including sensor drift, noise, and the need for precise placement.

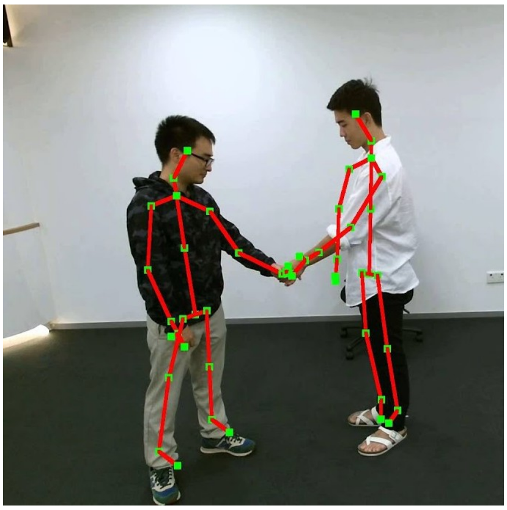


Figure 1.1: Skeletal data estimated from rgb camera

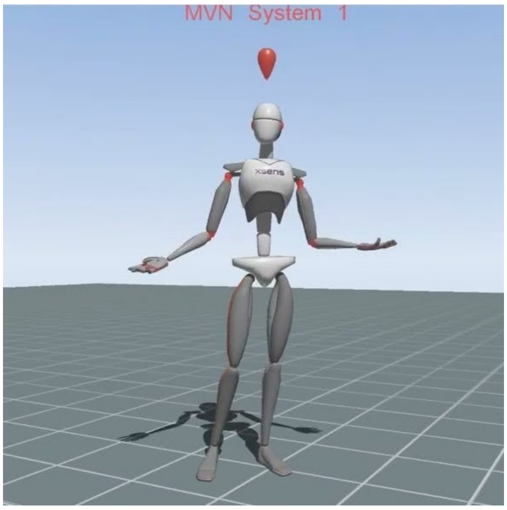


Figure 1.2: Model based on skeletal data measured with inertial sensors

Since each data modality has its strengths and limitations, methods which combine different modalities have shown to achieve a better accuracy [12]. State-of-the-art methods nowadays leverage multimodal data and sophisticated algorithms to achieve high performance in complex environments. We will briefly see some approaches on chapter 2

Despite the progress, action recognition still faces several challenges. One of the primary challenges is the high intra-class variability, where the same action can be performed differently by different individuals or even by the same individual under varying conditions. This variability can arise from differences in speed, style, body morphology, and execution dynamics, making it difficult to have a model that accurately recognizes actions across different subjects.

Another challenge is the opposite problem: inter-class similarity, where distinct actions share similar movements or joint configurations, leading to potential mistakes in classification. Then there are a list of technical limitations, for example environmental factors, such as occlusions, lighting conditions, and background clutter, in addition to sensor limitations, like noise, drift, and loss of data. Finally, another challenge is real-time inference where the main difficult arise from the need to process and analyze large volumes of high-dimensional data in real-time, requiring efficient algorithms that can find a well performing trade-off between accuracy and computational efficiency. Online skeleton-based action recognition involves the real-time analysis and classification of human actions without requiring the entire sequence to be observed beforehand. The key advantage of online recognition is its ability to provide immediate feedback, making it particularly useful in dynamic environments, where quick response times are crucial. Moreover, it allows for continuous monitoring and can adapt to ongoing activities, offering a more flexible and responsive solution compared to traditional offline methods, and open to a world of real-world applications, from health-care to industry scenarios.

One scenario where all the challenges mentioned above of action recognition are present is an industrial assembly line. Workers frequently perform similar actions, such as reaching components and placing them or assembling parts and use a manual tool, making it difficult to distinguish between subtle variations. For example, reaching an object and placing are two actions with a very similar movement of the arm. On the other side different workers can perform action in a different way, for example picking parts with one or two hands, or in other cases using right or left hand. In this environment, real-time action recognition is a crucial task, since it is needed for interactive systems, and it allows to monitor productivity, identify errors and improve human and robot collaboration, ensuring an increase of both productivity and safety. In addition, in industrial context, the comprehension of human behaviour gains more value due to the fact that there is an intersection with ergonomics. In this environment, ergonomics is about the monitoring, assessing and even correcting human posture during work shifts, and the processing and recognition of human movements has a key role for improving the ergonomics of workstations, allowing for a reduction of Work-related musculoskeletal disorders (WMSDs) and injury risk. In particular, real-time action recognition provide enough information to develop collaborative robots, which could be employed to help the worker to avoid repetitive dangerous movements and maintain a healthy posture.

The motivation for this work stems from the need for accurate and real-time action recognition systems that can operate effectively in industrial contexts, where actions are often repetitive and similar. In this work, we focused on action recognition in industrial settings, particularly considering various assembly tasks. The main objective was to investigate the ability of one state-of-the-art network to distinguish between very similar actions in real-time, and make some minor adjustments to the model to achieve our goal. To exploit the most accurate representation of these movements, we explored the use of inertial sensors and skeleton-based networks. The choice to employ inertial sensors has also other advantages: skeletal data has a lower-demanding computational cost with respect to video processing and they perform well also in narrow and occluded spaces, making them perfect for workstations in industry lines, which often are confined spaces. The set of action chosen to achieve our goal is from the AnDy Dataset [13], that we present in detail in section 4.1.1. It is a dataset recorded with inertial sensors which contains actions that are part of an assembly process. We firstly investigate how the model perform on this data with a set of experiments, for example cross-validation between subjects. Then we used the model trained on AnDy on a smaller dataset collected in our lab, which contains actions belonging to the assembly of a chair and which are similar to the ones in AnDy. The purpose was to explore how a trained model has learned the actions and can generalize its knowledge to a different scenario. To improve accuracy we also explore the technique of fine-tuning, which led to a better final result. This research aims to contribute to the field of action recognition, addressing the limitations of current methods and improving the robustness and efficiency of online action recognition systems.

This document is organized as follows:

- Chapter 2 will take an overview over literature about human offline and online action recognition and ergonomics assessment, reviewing existing methodologies and highlighting their strengths and weaknesses.
- Chapter 3 will explain the approach for the proposed solution and the methodology in details.
- Chapter 4 will describe the experiments conducted on AnDy dataset, alongside their motivation, results and a critical analysis.
- Chapter 5 will outline the process of data collection, and then illustrate experiments on custom dataset, followed by a discussion on results.
- Chapter 6 will provide a summary of the work, an insight on its limitations, followed by a look to future developments and then a final conclusion.

2

Literature review

Human action recognition has emerged as a critical area of research in computer vision and machine learning, because of its wide range of applications in several fields as surveillance, healthcare, human-computer interaction, and more. Because of this, researchers are showing increasing interest in human activity recognition, as shown by the growing number of research publications in the field over the last ten years as we can see in 2.1. In this chapter we will review the state-of-the-art methods to address this task, which is, learn to recognize a human movement and classify it as a particular action from a set of labels. The HAR problem is in this sense a classification problem. Recent works in action

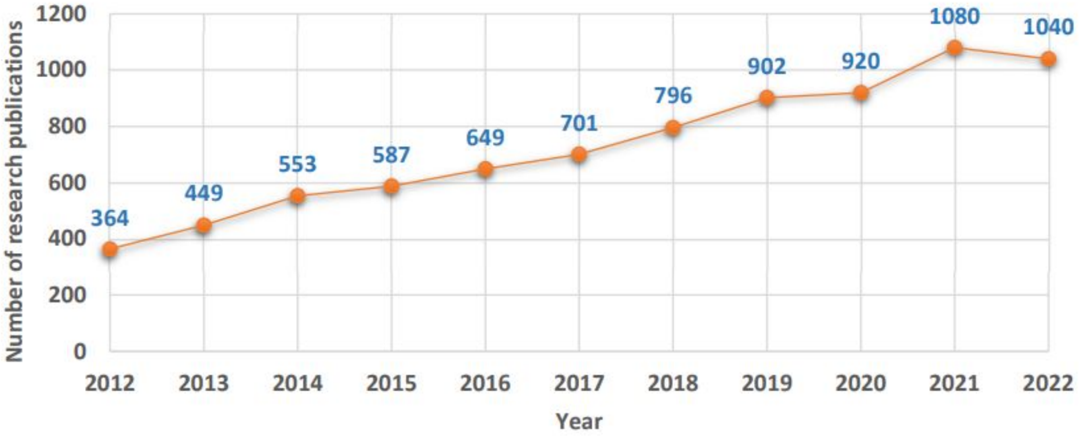


Figure 2.1: Number of research publications on human action recognition over the decade 2012-2022 [14]

recognition research have focused on improving the accuracy and efficiency of recognition systems by leveraging advanced deep learning techniques. The field has evolved from simple gesture recognition to more complex and nuanced action understanding, incorporating various data modalities such as video, depth maps, skeletal data, body sensors and combination of them.

For the purposes of this research, we focused specifically on skeleton-based action recognition. This approach utilizes a simplify model of the human skeletal structure to identify and classify human actions. The skeleton is modeled as a graph, where each node represent a joint and each edge represent a bone. Joints 3D position are estimated from cameras or from body sensors like Inertial Measurement Unit (IMU). Skeleton-based action recognition has emerged as a prominent field of research due to the increasing availability and accuracy of 3D skeleton data. This modality, which represents human actions through the trajectories of skeletal joints, offers a robust and semantically rich way of understanding human movements, making them suitable for real-world applications.

The findings and discussions of literature review related to skeleton-based action recognition are presented in three subsections: offline recognition, online recognition, and ergonomic assessment.

2.1 Offline skeleton-based action recognition

First approach to this task is offline, which means performing the recognition task starting from a complete set of data, usually a dataset widely used in literature to allow for performance comparison. Offline skeleton-based action recognition counts a lot of works, as presented in recent surveys on this topic: in the GitHub repository Awesome Skeleton-based Action Recognition[1] a collection of previous works is available. With the remarkable development and outstanding performance of deep learning methods in various computer vision areas, many types of deep learning architectures are employed in this field [15]: Recurrent Neural Network (RNN) based methods leverage skeleton sequences as natural time series data, treating joint coordinates as sequential vectors, aligning well with the RNNs capacity for processing time series information [16]. To enhance the learning of temporal context within skeleton

[1] https://github.com/firework8/Awesome-Skeleton-based-Action-Recognition

sequences, many variants like Long Short-Term Memory (LSTM) [17] [18] and Gated Recurrent Unit (GRU) [19] have been employed and several variants have been developed. While RNN-based methods excel in processing temporal evolution, their main downside is the lack to capture spatial information of the input data: Convolutional Neural Network (CNN) complement RNN-based techniques, as they are effective in capturing spatial patterns and also joints relationships of the input data [20]. Additionally, a relatively recent approach, the Graph Convolutional Network (GCN) has gained attention for its ability to model skeleton data in a natural topological graph structure, with joints and bones as vertices and edges, respectively. From the general idea of the GCN, other architectures have been developed as Spatio-Temporal Graph Convolutional Network (STGCN) to consider the spatial and temporal evolution of the graph [21]. Advancements in skeleton-based action recognition have also been made through hierarchical modeling techniques [22]: instead of processing the entire skeleton as a single entity, researchers have proposed dividing the skeleton into multiple parts according to human physical structure. These parts are then individually processed by separate subnetworks, which are hierarchically fused at higher network layers. An end-to-end hierarchical RNN model has been developed that effectively captures the long-term contextual information of temporal sequences. This approach has demonstrated state-of-the-art performance with high computational efficiency, outperforming other deep RNN architectures and traditional methods on several publicly available datasets, for example the two NTU RGB-d datasets[23] [24], which are the most widely use in this field. More recently, transformer-based methods capture the spatial-temporal relation of the input 3D skeleton data mainly based on its core multihead self-attention (MSA) mechanism. Offline action recognition had gained enough importance that exist open-source frameworks for this task, as MMAction[2]. This framework allows to train and test different models, customize them and use them on existing or custom datasets. There is a specific section for the skeleton-based models, containing four different architectures: PoseC3D, 2s-AGCN, STGCN [25] and STGCN++. The advantage of using this type of framework is to have full access to state-of-the-art methods used, debugged and improved by a big community of researchers.

[2] https://mmaction2.readthedocs.io/en/latest/

2.2 Online skeleton-based action recognition

As offline action recognition, multi-view cameras are widely used also for this task, but when needed a fast processing of the video even small changes of the skeleton perspective could lead to massive errors in the recognition of the movement. One of the main challenges in skeleton-based action recognition is dealing with large variations in viewpoints, which can significantly impact the accuracy of action recognition models. A novel approach to address this issue involves the use of view-adaptive recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) architecture [26]. This method allows the network to automatically regulate observation viewpoints during the occurrence of an action, transforming the skeletons from various views into more consistent viewpoints. Unlike traditional methods that re-position skeletons based on predefined criteria, this adaptive approach ensures that the continuity of the action is maintained while significantly improving performance on benchmark datasets. Despite the success of RNNs, they suffer from limitations such as non-parallelism and the gradient vanishing problem, which can hinder optimization. To overcome these challenges, a new approach based on Transformer architecture has been proposed. The OadTR framework, an encoder-decoder model, captures the relationships and global interactions between historical observations and predicts future context simultaneously. This model not only achieves higher training and inference speeds but also significantly outperforms state-of-the-art RNN-based methods in terms of accuracy, as demonstrated in [27]

In recent years, skeleton-based action recognition has evolved significantly, particularly in online settings where real-time recognition is essential. One of the core challenges in this area is managing variations in viewpoints that can severely impact the performance of action recognition models. Traditional approaches, like those using Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM), focus on learning temporal dependencies in sequential data. A novel advancement in this space is the view-adaptive RNNs, which can dynamically adjust the observation viewpoints during an action, ensuring more consistent recognition by transforming skeletons from varying angles into unified viewpoints. This approach, introduced by [26], significantly enhances performance on benchmark datasets compared to methods that rely on predefined criteria for repositioning skeletons. However, RNN-based approaches still suffer from limitations like non-parallelism and gradient vanishing, which affect optimization efficiency.

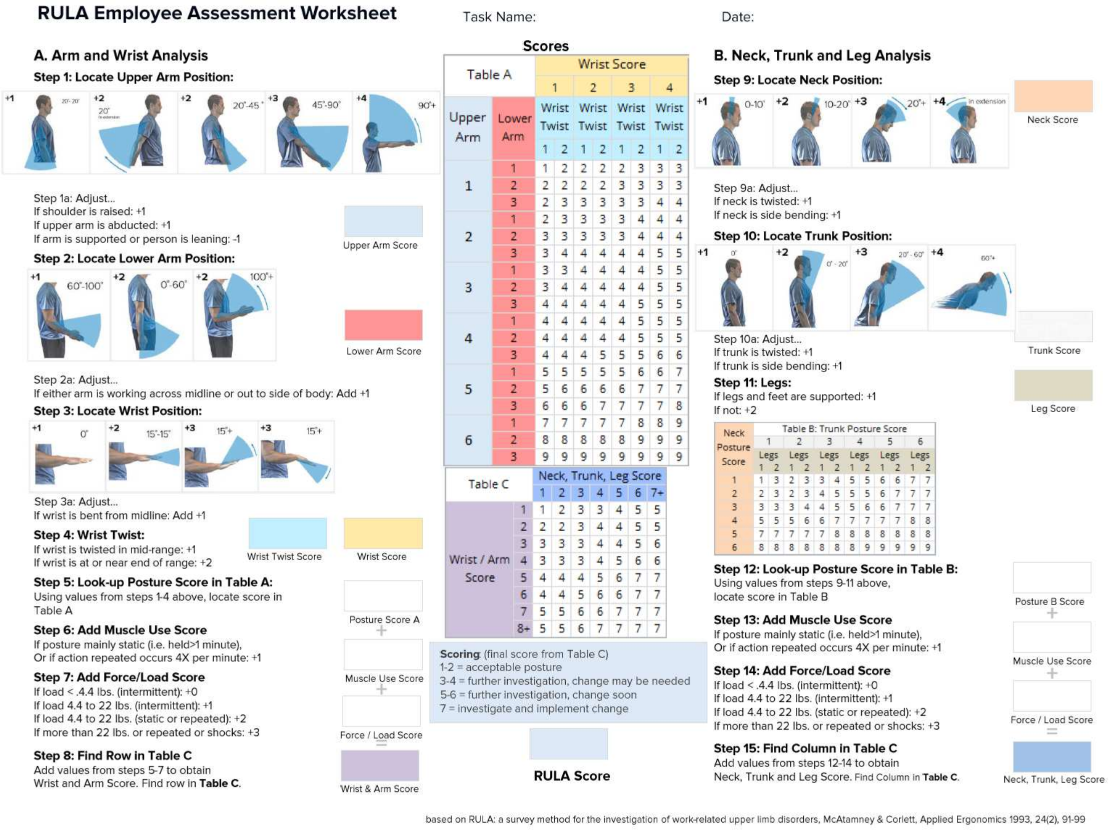
To address these limitations, Transformer architectures have gained traction. A notable model, the OadTR framework [27], introduces an encoder-decoder structure to tackle online action recognition. The encoder focuses on extracting relationships and global interactions from historical data, while the decoder anticipates future clip representations to predict upcoming actions. This approach not only enhances recognition accuracy but also achieves faster training and inference speeds compared to traditional RNNs, but also significantly outperforms state-of-the-art RNN-based methods in terms of accuracy, as demonstrated by its authors.

In addition to camera-based skeleton tracking, wearable sensor-based human action recognition has emerged as a powerful alternative for real-time applications. A recent model, the Bidirectional-Gated Recurrent Unit-Inception (Bi-GRU-I) [28], was developed to improve the accuracy of HAR using inertial sensors. This model integrates Bi-GRU layers and Inception layers, capturing both temporal and spatial features from inertial signals while maintaining a lower parameter count. The Bi-GRU-I model has demonstrated superior performance on multiple datasets highlighting its robustness in recognizing human actions with minimal sensor configurations.

In online skeleton-based action recognition, integrating spatial and temporal features plays a crucial role in improving recognition accuracy. A recent approach leverages a distributed camera network to enhance human pose estimation in occlusion scenarios by fusing multiview skeleton data [29]. This method introduces a group sampling mechanism, which selectively fuses past and current action frames, addressing the redundancy in neighboring frames and incorporating long-term contextual information. The combination of spatial skeleton features (geometrical data) and temporal motion features ensures that both dimensions are adequately represented in the recognition process. Evaluations on datasets show impressive accuracy levels, illustrating the efficacy of this approach for real-time action recognition with distributed camera networks.

2.3 Ergonomics assessment

Human action recognition together with motion capture technologies like inertial sensors, has evolved as a key technologies in monitoring and assessing workers' movements, enabling real-time feedback on task execution and posture. This, in turn, helps prevent injuries caused by repetitive strain or poor ergonomics and supports the optimization of workflows for efficiency. Over the past decades, numerous approaches have emerged to assess ergonomics and worker safety, as presented by surveys as [30] [31] [32]. These methods primarily aim to evaluate risk factors that contribute to dangerous works, focusing on posture and movements and excluding environmental and chemical exposures. Traditionally, ergonomics assessment is divided into three broad categories: subjective judgments, systematic observations, and direct measurements. Subjective judgments involve self-reported questionnaires or expert interviews, while systematic observations are expert-led procedures based on visual assessments of the workplace, often supported by video recordings. Technological advancements, particularly in sensor technology, have automated many of these observational methods. Direct measurements, on the other hand, utilize sensors to capture precise, real-time data on physical strain, which can be further processed using ad-hoc models. In the realm of ergonomics assessment, tools such as the European Assembly Worksheet (EAWS) [33] and the Rapid Upper Limb Assessment (RULA) [34] have become essential for evaluating physical workload and identifying risks associated with WMSDs. Alongside these, many other well-established tools are used to assess ergonomic risks in various industrial tasks. These include the O (OWAS), the Occupational Safety and Health Administration (OSHA) guidelines, the Job Safety Analysis (JSA), the Postural Ergonomic Risk Assessment (PERA), and the Moore-Garg Strain Index. Each of these tools addresses specific aspects of ergonomics, such as posture, force, or repetition, providing comprehensive frameworks for evaluating and mitigating the risk factors associated with WMSDs in different work environments. EAWS is widely used in industrial settings to assess overall physical strain, offering a systematic approach to evaluate postures, forces, and repetitive movements. It assigns a risk score based on task-related physical demands, categorizing the results into a traffic light system (green, yellow, red) to indicate low, moderate, or high levels of ergonomic risk. EAWS is particularly effective for comprehensive assessments in manufacturing and assembly environments, as it ensures compliance with international standards while allowing for easy integration into workplace ergonomics programs. Similarly, RULA provides a quick and simple way to evaluate the ergonomic risk of upper-limb tasks, particularly those involving repetitive or static postures, focusing more on a posture-based evaluation. By assigning scores to various body segments, including the neck, trunk, and arms, RULA produces a cumulative risk score that guides corrective actions, as it is possible to see in its worksheet reported in figure 2.2. It focuses on the risk factors associated with neck, shoulder, and arm movements, making it an effective tool for pinpointing high-risk tasks where immediate ergonomic interventions are necessary. A novel approach to ergonomics assessment involves the use of multiple ergonomic indexes to provide a more comprehensive evaluation of physical workload and risks. By combining different indexes, this methodology accounts for various contributors to work-related musculoskeletal disorders, ensuring a more robust and personalized analysis of the factors affecting workers. For example, framework presented in [35] introduces an online system that continuously monitors workers kinematic and dynamic quantities, providing real-time estimates of physical load during typical manufacturing tasks. Through statistical and surface electromyography analyses, the most relevant ergonomic indexes for each task are identified, ensuring that the evaluation aligns with both task-specific and worker-specific requirements. Multi-index



RULA Employee Assessment Worksheet

Task Name:

Date:

A. Arm and Wrist Analysis

Step 1: Locate Upper Arm Position:

+1
20°-20°
+2
20°
in extension
+2
20°-45°
+3
45°-90°
+4
90°+

Step 1a: Adjust...
If shoulder is raised: +1
If upper arm is abducted: +1
If arm is supported or person is leaning: -1

Upper Arm Score

Step 2: Locate Lower Arm Position:

+1
60°-100°
+2
0°-60°
+2
100°+

Lower Arm Score

Step 2a: Adjust...
If either arm is working across midline or out to side of body: Add +1

Step 3: Locate Wrist Position:

Step 3a: Adjust...
If wrist is bent from midline: Add +1

Step 4: Wrist Twist:
If wrist is twisted in mid-range: +1
If wrist is at or near end of range: +2

Wrist Twist Score

Wrist Score

Step 5: Look-up Posture Score in Table A:
Using values from steps 1-4 above, locate score in Table A

Posture Score A
+

Step 6: Add Muscle Use Score
If posture mainly static (i.e. held>1 minute),
Or if action repeated occurs 4X per minute: +1

Muscle Use Score
+

Step 7: Add Force/Load Score
If load < .4.4 lbs. (intermittent): +0
If load 4.4 to 22 lbs. (intermittent): +1
If load 4.4 to 22 lbs. (static or repeated): +2
If more than 22 lbs. or repeated or shocks: +3

Force / Load Score
=

Step 8: Find Row in Table C
Add values from steps 5-7 to obtain
Wrist and Arm Score. Find row in **Table C**.

Wrist & Arm Score

Scores

Table A		Wrist Score							
		1		2		3		4	
Upper Arm	Lower Arm	Wrist Twist		Wrist Twist		Wrist Twist		Wrist Twist	
		1	2	1	2	1	2	1	2
1	1	1	2	2	2	2	3	3	3
	2	2	2	2	2	3	3	3	3
	3	2	3	3	3	3	3	4	4
2	1	2	3	3	3	3	4	4	4
	2	3	3	3	3	3	4	4	4
	3	3	4	4	4	4	4	5	5
3	1	3	3	4	4	4	4	5	5
	2	3	4	4	4	4	4	5	5
	3	4	4	4	4	4	5	5	5
4	1	4	4	4	4	4	5	5	5
	2	4	4	4	4	4	5	5	5
	3	4	4	4	5	5	5	6	6
5	1	5	5	5	5	5	6	6	7
	2	5	6	6	6	6	7	7	7
	3	6	6	6	7	7	7	7	8
6	1	7	7	7	7	7	8	8	9
	2	8	8	8	8	8	9	9	9
	3	9	9	9	9	9	9	9	9

Table C		Neck, Trunk, Leg Score						
		1	2	3	4	5	6	7+
Wrist / Arm Score	1	1	2	3	3	4	5	5
	2	2	2	3	4	4	5	5
	3	3	3	3	4	4	5	6
	4	3	3	3	4	5	6	6
	5	4	4	4	5	6	7	7
	6	4	4	5	6	6	7	7
	7	5	5	6	6	7	7	7
	8+	5	5	6	7	7	7	7

Scoring: (final score from Table C)
1-2 = acceptable posture
3-4 = further investigation, change may be needed
5-6 = further investigation, change soon
7 = investigate and implement change

RULA Score

B. Neck, Trunk and Leg Analysis

Step 9: Locate Neck Position:

Neck Score

Step 9a: Adjust...
If neck is twisted: +1
If neck is side bending: +1

Step 10: Locate Trunk Position:

Trunk Score

Step 10a: Adjust...
If trunk is twisted: +1
If trunk is side bending: +1

Step 11: Legs:
If legs and feet are supported: +1
If not: +2

Leg Score

Neck Posture Score	Table B: Trunk Posture Score											
	1		2		3		4		5		6	
	Legs		Legs		Legs		Legs		Legs		Legs	
	1	2	1	2	1	2	1	2	1	2	1	2
1	1	3	2	3	3	4	5	5	6	6	7	7
2	2	3	2	3	4	5	5	5	6	7	7	7
3	3	3	3	4	4	5	5	6	6	7	7	7
4	5	5	5	6	6	7	7	7	7	7	8	8
5	7	7	7	7	7	8	8	8	8	8	8	8
6	8	8	8	8	8	8	8	9	9	9	9	9

Step 12: Look-up Posture Score in Table B:
Using values from steps 9-11 above,
locate score in Table B

Posture B Score
+

Step 13: Add Muscle Use Score
If posture mainly static (i.e. held>1 minute),
Or if action repeated occurs 4X per minute: +1

Muscle Use Score
+

Step 14: Add Force/Load Score
If load < .4.4 lbs. (intermittent): +0
If load 4.4 to 22 lbs. (intermittent): +1
If load 4.4 to 22 lbs. (static or repeated): +2
If more than 22 lbs. or repeated or shocks: +3

Force / Load Score
=

Step 15: Find Column in Table C
Add values from steps 12-14 to obtain
Neck, Trunk and Leg Score. Find Column in **Table C**.

Neck, Trunk, Leg Score

based on RULA: a survey method for the investigation of work-related upper limb disorders, McAtamney & Corlett, Applied Ergonomics 1993, 24(2), 91-99

Figure 2.2: RULA assessment worksheet

volves the use of multiple ergonomic indexes to provide a more comprehensive evaluation of physical workload and risks. By combining different indexes, this methodology accounts for various contributors to work-related musculoskeletal disorders, ensuring a more robust and personalized analysis of the factors affecting workers. For example, framework presented in [35] introduces an online system that continuously monitors workers kinematic and dynamic quantities, providing real-time estimates of physical load during typical manufacturing tasks. Through statistical and surface electromyography analyses, the most relevant ergonomic indexes for each task are identified, ensuring that the evaluation aligns with both task-specific and worker-specific requirements. Multi-index approaches not only enhances the precision of ergonomic assessments but also supports the adoption of preventive measures by identifying key physical risk factors earlier and more effectively than traditional single-index tools.

Machine learning and deep learning techniques play a pivotal role in modern ergonomics assessment, automating the detection and classification of postures and movements to evaluate physical strain. Among these, Hidden Markov Model (HMM) have been successfully applied in the field. For instance, one study [36] proposes an automatic ergonomic assessment system based on posture recognition using HMMs, trained with data from inertial sensors. In this approach, a taxonomy of activities, compatible with standard ergonomic worksheets like EAWS, is defined to categorize actions and postures. The system is trained and tested using data from participants mimicking industrial tasks, and dedicated feature selection methods are employed to enhance recognition performance. This methodology shows promising results in accurately identifying ergonomically relevant postures and actions, providing a robust framework for automatic risk assessment in industrial settings. Other machine learning models like k-Nearest Neighbors (kNN), Decision Trees (DT), and Generalized Linear Models (GLMs) further contribute to the classification and risk evaluation, providing a comprehensive set of methodologies for detecting and mitigating ergonomic hazards. Deep learning methods further enhance ergonomic assessment by capturing more complex patterns in data. Artificial Neural Networks (ANNs) are widely used for this purpose, and include models such as Multilayer Perceptron (MLP), Convolutional Long Short-Term Memory (CLSTM), Static Neural Networks (SNN), Convolutional Neural Networks (CNN), and Learning Vector Quantization (LVQ). These architectures allow for the automatic extraction of features from data, enabling more accurate and real-time assessments of ergonomic risks.

3

Skeleton-based action recognition methodology

In this chapter, we will describe the methodology employed in this thesis. This work leverage a state-of-the-art skeleton-based action recognition model as a starting point for a custom development. The chapter aims to explain the operational blocks of our work, which are depicted in figure 3.1, and it is organized as follows: we begin by introducing the skeleton model, which serves as the core representation of human poses and movements in our system. This model captures key joint positions and skeletal structure, providing robust features for action recognition. Following this, we dive into the architectures and algorithms utilized in this work. We outline the base model selected for customization, discussing its original design and the reasoning behind its selection. Subsequently, we detail the modifications and improvements made to the model, aimed at optimizing it for the specific context we are dealing with. Next, we describe the dataset used in this study, including its characteristics, structure, and relevance to the task of action recognition. The dataset provides the necessary data for training, validating, and testing the model, and its selection is critical to ensuring the model's effectiveness in real-world scenarios.

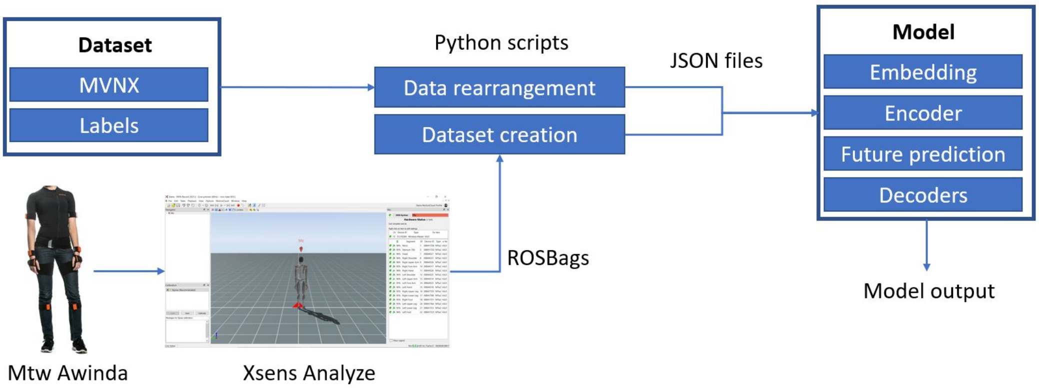


Figure 3.1: A general overview of our work

3.1 Skeleton model

Skeleton data are acquired using the Xsens MTw Awinda system [37], a state-of-the-art wireless motion capture solution designed for accurate and efficient measurement of human body movements. The Xsens Awinda system is composed of a set of 17 wireless sensors, each embedded with 3D accelerometers, gyroscopes, and magnetometers. These sensors are strategically placed on the subject's body at specific anatomical locations listed in table 3.1 to capture the 3D orientation, angular velocity, and acceleration of each segment, from which other measurements such as joint positions or relative angles can be calculated by the algorithm that fits data into the musculoskeletal model. The sensors communicate wirelessly with a central station that collects the data and streams it to a computer for real-time processing or storage. This system is particularly advantageous due to its high degree of mobility and ease of setup, making it suitable for various applications, including biomechanics, sports science, and animation.

The skeletal model used in this study is based on the Xsens MVN Animate software, which provides a detailed and anatomically accurate human skeleton model 3.2. According to the MVN user manual [1], the skeletal model consists of

[1] https://www.movella.com/hubfs/MVN_User_Manual.pdf?__hstc=233546881.1fa5198786ade7bb8cace6bc2dd887f0.1663745187069.1670940876901.1670945468596.94&__hssc=233546881.14.1670945468596&__hsfp=700330257

23 segments (bones) and 22 joints, listed respectively in tables 3.2 and 3.3. The tracked joints include key anatomical locations such as pelvis, some vertebrae, head, shoulders, elbows, wrists, hips, knees, and ankles. These joints are critical for capturing the full range of human motion and are essential for tasks such as action recognition. The Xsens system tracks the position and orientation of these joints in real-time, allowing for the precise reconstruction of the skeletal structure and its movements. The accuracy of joint tracking is facilitated by the placement of sensors on both proximal and distal ends of each bone segment, ensuring that even complex motions, such as twisting or bending, are accurately captured. The accuracy of the data is sensitive to the correct placement and calibration of the sensors, in addition to inserting body measurement of the subject. The resulted detailed skeletal representation is crucial for the effective recognition and analysis of human actions, as it allows for the extraction of meaningful motion features that are directly related to the dynamics of the human body.

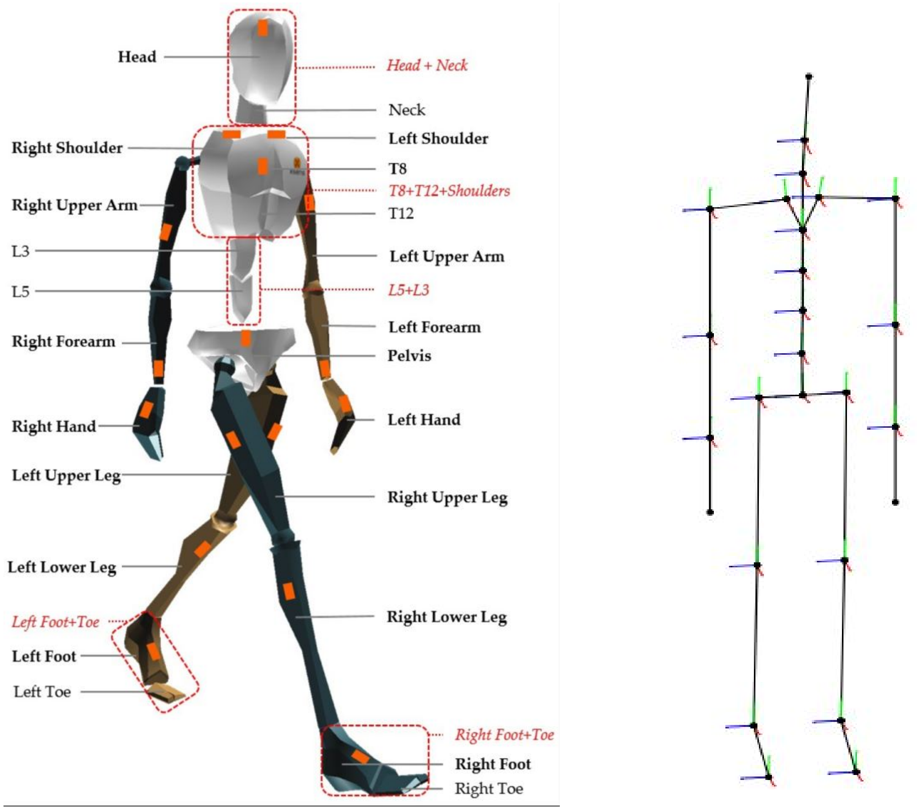


Figure 3.2: On the left: musculoskeletal model. It highlights the correspondence between sensors, joints and links.
On the right: raw model of skeleton based on joint positions and links

Index	Tracker
1	Pelvis
2	T8
3	Head
4	Right shoulder
5	Right upper arm
6	Right forearm
7	Right hand
8	Left shoulder
9	Left upper arm
10	Left forearm
11	Left hand
12	Right upper leg
13	Right lower leg
14	Right foot
15	Left upper leg
16	Left lower leg
17	Left foot

Table 3.1: Body parts tracked with a sensor

Index	Segment
1	Pelvis
2	L5
3	L3
4	T12
5	T8
6	Neck
7	Head
8	Right shoulder
9	Right upper arm
10	Right forearm
11	Right hand
12	Left shoulder
13	Left upper arm
14	Left forearm
15	Left hand
16	Right upper leg
17	Right lower leg
18	Right foot
19	Right toe
20	Left upper leg
21	Left lower leg
22	Left foot
23	Left toe

Table 3.2: Segments of the skeleton model

Index	Joint
1	L5-S1
2	L4-L3
3	L1-T12
4	T9-T8
5	T1-C7
6	C1-Head
7	C7-Right shoulder
8	Right shoulder
9	Right elbow
10	Right wrist
11	C7-Left shoulder
12	Left shoulder
13	Left elbow
14	Left wrist
15	Right hip
16	Right knee
17	Right ankle
18	Right ball foot
19	Left hip
20	Left knee
21	Left ankle
22	Left ball foot

Table 3.3: Joints of the skeleton model

3.2 Model

Traditional methods typically require the complete observation of an action before making a classification, which is impractical in dynamic environments and even in real-time streaming of data, as in our case. To address this, we needed a model being able to base the prediction only on previous frames, without knowing, during the inference phase, the whole sequence of skeletons. After a wide research we choose to exploit the *InfoGCN++* framework [38], specifically designed for online skeleton-based action recognition.

3.2.1 INFOGCN++

InfoGCN++ [38] is a state-of-the-art model developed for online skeleton-based action recognition. Building upon the original *InfoGCN* model, which achieved notable accuracy, *InfoGCN++* addresses a critical limitation of previous models, which is the necessity for complete observation of action sequences before classification. *InfoGCN++* enables recognition based on partial observations, making it suitable for situations that require immediate responses. The code implementing this model is available in a public GitHub repository [2]

[2] https://github.com/stnoah1/infogcn2/tree/main

Architecture

The overall architecture of *InfoGCN++* is shown in figure 3.3 and consists of four essential components: an embedding layer, an encoder, a future motion predictor, and task-specific decoders. The embedding layer, encoder, and action classification decoder are inherited from the offline version of the model *InfoGCN* [39], while the future motion predictor and future motion prediction decoder are newly introduced to augment the model's ability to anticipate and recognize actions in real time. The architecture is designed to handle continuous skeleton data, ensuring that the spatial-temporal relationships between joints are efficiently captured and processed. The operational workflow of *InfoGCN++* can be summarized as follows:

1. **Data Embedding:** Each incoming skeleton frame is embedded into a latent space with spatial positional embeddings.

2. **Encoding Observations:** The encoder processes the sequence of embedded frames, producing a spatio-temporal representation Z_t.

3. **Predicting Future Representations:** The future motion predictor extrapolates Z_t to predict future representations $\hat{Z}^{(t)}_{t+1:t+N}$.

4. **Action Recognition and Motion Prediction:** The task-specific decoders classify the action and predict future skeleton frames.

As new frames become available, the process repeats, continuously refining the predictions and classifications.

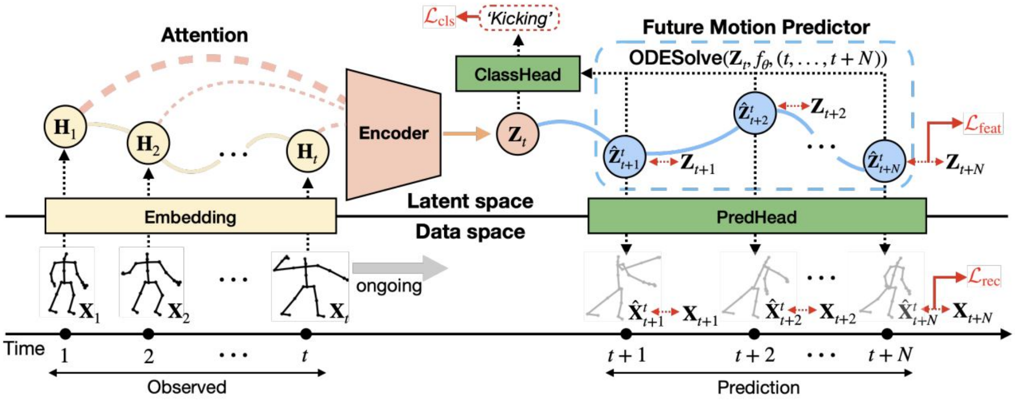


Figure 3.3: An overview of the InfoGCN++ framework workflow

Now we will introduce more in detail each of the four main elements of *InfoGCN++*.

1. Embedding Layer The embedding layer is responsible for transforming 3D skeleton data into a latent space that can be further processed by the encoder. Each skeleton frame, represented as $X_t \in \mathbb{R}^{V \times 3}$, where V denotes the number of joints, and 3 are the spatial coordinates, is linearly projected into a latent space of dimension D. This process is mathematically described as:

$$H_t^{(0)} = \text{Linear}(X_t) + \text{PE} \tag{3.1}$$

where PE $\in \mathbb{R}^{V \times D}$ represents learnable spatial positional embedding that incorporate joint position information and their relations. This step is essential for encoding the spatial relationships among the joints in a way that can be utilized for further processing.

2. Encoder The encoder converts the features from the embedding layer into a spatio-temporal representation, $Z_t \in \mathbb{R}^{V \times D}$, of the sequence up to the current time t. The encoder consists of two main modules:

- *Spatial Modeling Module:* Employs Self-Attention Graph Convolution (SA-GC)) layers to model spatial dependencies among joints.
- *Temporal Modeling Module:* Implements a Transformer encoder with a causal mask, enabling it to focus only on past frames, which is crucial for real-time applications.

The encoder processes the sequence of embedded frames $H_{1:t}$, producing a representation Z_t according to the following formula:

$$Z_t = \text{Encoder}(H_{1:t}) \tag{3.2}$$

The temporal modeling in the encoder relies on the causal mask, which ensures that only past and present frames are considered, simulating a real-time scenario. The attention mechanism, which is applied to individual joints i, is computed as follows:

$$\text{Attention}_t[i] = \text{Softmax}\left(\frac{Q_t[i]K_{1:t}^{\top}[i]}{\sqrt{D}}\right) V_{1:t}[i] \tag{3.3}$$

where Q_t, K_t, V_t are the queries, keys, and values computed by the SA-GC layers. The attention is calculated separately for each joint i, aggregating spatial features for the temporal modeling process.

3. Future Motion Predictor The future motion predictor is a novel element introduced in *InfoGCN++* that allows the model to anticipate future movements based on the current observation. By framing the prediction task as an extrapolation problem, the future motion predictor utilizes Neural Ordinary Differential Equations (Neural ODEs) to model the continuous latent dynamics of the skeleton, implemented thanks to *torchdiffeq* the Differentiable ODE Solvers for pytorch.

The prediction of future frames is formulated as an Initial Value Problem (IVP) and is solved using the observed representation Z_t as the initial condition:

$$\hat{Z}^{(t)}_{t:t+N} = \text{ODESolve}(f_\theta, Z_t, (t, \ldots, t+N)) \tag{3.4}$$

Here, $\hat{Z}^{(t)}_{t+n}$ represents the predicted representation of the n-th future frame, and f_θ is the ODE function parameterized by a neural network. The ODE function is designed to capture the temporal evolution of the latent features using SA-GC layers and temporal positional embeddings $\text{PE}_{\text{temporal}} \in \mathbb{R}^{T \times D}$. The temporal positional embeddings are used to inject temporal information into the ODE function, allowing the model to account for the time-variant properties of the sequence.

4. Task-Specific Decoders *InfoGCN++* employs multi-task learning, with two separate decoders that work simultaneously:

- The *action classification decoder* is responsible for predicting the action category from the observed and predicted representations.
- The *future motion prediction decoder* reconstructs future skeleton frames from the predicted representations, enhancing the model's ability to anticipate future movements.

By training the model to perform both tasks action recognition and future motion prediction *InfoGCN++* learns to create rich and robust representations of the observed sequence.

Data loading and preprocessing

The data feeder is an algorithm which goal is to prepare and supply data to the model. It ensures that the model receives data in the correct format, at the appropriate time, and with any necessary preprocessing or augmentation. It is responsible for loading, batching, transforming and normalizing, windowing and masking skeletons. This element of the model is designed from the authors starting from implementation of Semantic-Guided Network (SGN) [40], Hierarchical Co-occurrence Network (HCN) [41] and a system for skeleton-based action recognition called Predict and Cluster [42].

The data loading and preprocessing pipeline is crucial for preparing skeleton-based action recognition models. As reported in algorithm 1, the data initialization begins by determining whether the data is for training or testing. Depending on the split, either the training or test data is loaded into a dictionary structure (data_dict) that at this point contains list of file names - one file for each action - with their length and their label. The algorithm also initializes skeleton bone connections and gathers the corresponding labels for each data sample. The load_data() function is then invoked to load the raw skeleton data from files into memory.

Algorithm 1 Feeder Data Loading

```
Input: data_path, split
if split is 'test' then
    Set train_val to 'test' and load test data into data_dict
else
    Set train_val to 'train' and load training data into data_dict
end if
Initialize skeleton bone connections and an empty list label
for each data_sample in data_dict do
    Append the label from data_sample to the list
end for
Call load_data() function

Function: load_data()
Initialize an empty list data
for each data_sample in data_dict do
    Load skeleton data from the corresponding file and append to data
end for
```

Once the data is loaded, algorithm shown in 2 handles essential transformations, normalization, windowing, and masking operations to ensure the model receives well-prepared inputs. For training, random transformations are applied to each skeleton sample to introduce variations that help the model generalize better. These transformations include random rotations and scaling. The skeleton data is centered on the first joint, transformed using these sampled parameters, and then normalized to ensure consistency across samples. A random windowing strategy is applied to select temporal segments from the data. For the test set, a deterministic approach is followed, with no random transformations and windowing, ensuring a consistent evaluation process.

In addition to these operations, the algorithm handles optional bone and motion feature extraction. If bone information is required, it computes vectors between connected joints to capture the relative positions of bones. Similarly, if motion features are needed, the algorithm calculates frame-to-frame differences to represent movement patterns.

The final data is reshaped and transposed into the (C, T, V) format, where C represents the number of input channels, which is the number of coordinates, T represents the number of frames, and V is the number of joints. A mask of ones is also created to mark the valid data points, ensuring that the model can differentiate between real data and padding when processing sequences of varying lengths.

This preprocessing approach is designed to prevent overfitting by introducing random variations during training and by maintaining consistency during testing, allowing the model to generalize across different subjects and scenarios.

Algorithm 2 Preprocessing

```
Input: index
Output: data, label, mask, index
Retrieve label and value corresponding to index
if train_val is 'train' then
    Randomly sample transformation parameters:
    agx ~ Uniform(−60, 60),
    agy ~ Uniform(−60, 60),
    s ~ Uniform(0.5, 1.5)
    Center value on the first joint
    Apply random transformation using rand_view_transform
    Reshape and normalize the transformed skeleton data
    Sample window_size random indices and extract data
else
    Set agx, agy, and s to 0, 0, and 1.0, respectively
    Center value on the first joint
    Apply deterministic transformation using rand_view_transform
    Normalize the transformed skeleton data
    Perform windowing using evenly spaced indices
end if
if data contains bone information then
    Initialize data_bone
    for each bone index do
        Compute bone vector by subtracting corresponding joint pairs
    end for
    Set data to data_bone
end if
if data contains motion information then
    Initialize data_motion
    Compute motion vectors by taking the difference between consecutive frames
    Set data to data_motion
end if
Reshape and transpose data into format (C, T, V)
Create a mask of ones with the same shape as data
return data, label, mask, index
```

Training algorithm

The training algorithm for skeleton-based action recognition is designed to process sequences of skeletal data and predict the action category for each observation in real time. The model uses an encoder to extract relevant features from the input sequence and a prediction mechanism to anticipate future frames. An ordinary differential equation (ODE) solver is employed to model temporal dynamics, ensuring robust predictions over time. The training procedure is detailed below:

Algorithm 3 Training Procedure for Skeleton-based Action Recognition

```
Input: $X_{1:T}$: Full observation of skeleton sequence.
Output: $\tilde{y}_{1:T}$: Action category of each observation.
for epoch ← 1 to $MaxEpoch$ do
  for $t$ ← 1 to $T$ in parallel do
    $H_t \leftarrow \text{Linear}(X_t)$ {Embedding of each skeleton}
    $Z_t \leftarrow \text{Encoder}(H_{1:t})$ {Feature encoding of the sequence up to time $t$}
    $\hat{Z}_{t:t+N} \leftarrow \text{ODESolve}(f_\theta, Z_t, (t, \ldots, t+N))$ {Solve ODE to predict future features}
    $\hat{X}_{t:t+N} \leftarrow \text{PredDecoder}(\hat{Z}_{t:t+N})$ {Decode predicted skeleton sequence}
    $\hat{y}_t \leftarrow \text{ClassDecoder}(\hat{Z}_{t:t+N})$ {Classify action for current time step $t$}
  end for
end for
```

The algorithm operates as follows:

- **Input and Initialization:** The input is a sequence of skeleton data $X_{1:T}$, where X_t represents the skeletal pose at time t. The objective is to predict the action category $\tilde{y}_{1:T}$ for each time step in the sequence.

- **Epoch Loop:** The training process runs for a predefined number of epochs, $MaxEpoch$, during which the model iteratively improves its predictions.

- **Time Step Processing:** For each time step t, the following operations are performed:

  1. *Feature Encoding:* The input sequence up to the current time step, $X_{1:t}$, is passed through an encoder to obtain a high-dimensional feature

representation, Z_t, which captures both spatial and temporal features of the skeleton data.

2. *ODE Solver:* To model temporal dynamics and predict the future movement over the next N time steps, the ODE solver, parameterized by f_θ, takes the encoded feature Z_t and generates predicted features $\hat{Z}_{t:t+N}$ for future time steps.
3. *Prediction Decoder:* The predicted features $\hat{Z}_{t:t+N}$ are then decoded to reconstruct the skeleton sequence $\hat{X}_{t:t+N}$ for the future frames.
4. *Action Classification:* The predicted features are passed through a classification decoder to obtain the predicted action label $\hat{y}_t$ for the current time step t. This allows the model to classify actions based on both historical and anticipated future data.

- **Parallel Processing:** The operations for each time step are performed in parallel, which enables efficient processing of the full sequence during training.

- **End of Training:** After processing the entire sequence for all epochs, the model updates its parameters by minimizing the classification loss, aiming to reduce the difference between the predicted labels $\hat{y}_{1:T}$ and the ground truth labels $\tilde{y}_{1:T}$. The process repeats until the model converges or reaches the maximum number of epochs.

This training procedure leverages temporal modeling and future prediction to enhance action classification performance. The ODE solver models the dynamics of skeleton movements, improving the model's ability to anticipate future actions and thus providing more accurate real-time recognition.

Multi-task Learning

Multi-task learning is a technique in deep learning where a model addresses multiple related tasks simultaneously, rather than learning each task independently. The goal is to leverage shared parameters and features between the tasks, which can improve the model's performance across all tasks by allowing it to generalize better. The multi-task learning strategy in *InfoGCN++* strengthens the model's action recognition capabilities by integrating future motion prediction as an auxiliary task. This approach has several advantages:

- **Enhanced Representation Learning:** By predicting future movements, the model learns richer representations that capture not only the observed dynamics but also potential future trajectories.

- **Improved Discriminative Power:** Early anticipation of actions improves the model's ability to distinguish between similar actions that may diverge later in their execution.

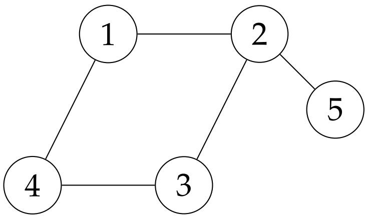
- **Reduced Recognition Latency:** The ability to recognize actions from partial observations reduces latency in time-sensitive applications.

3.2.2 Model customizations

Graph

The human skeleton model used in the MVN Analyze software is represented as a graph, where vertices correspond to the skeleton's joints, and edges represent the bones connecting them. This graph is implemented using a *Graph* class, which manages the relationships between joints, including self-links, inward connections, outward connections, and neighboring joints. These connections are stored as ordered pairs of joint indices. Specifically, the inward connections are defined based on the bone structure in the MVN skeleton, and the outward connections are derived by reversing these pairs. Neighbor connections are then obtained by combining both inward and outward connections.

The adjacency matrix, which captures the structure of the graph, is computed based on these relationships, as remarked in figure 3.4. The *Graph* class contains methods to compute various forms of the adjacency matrix, such as a binary adjacency matrix, a normalized version, and a k-scale graph, using utility functions from an external tools module. All these methods are borrowed from the *Graph* class included in the *InfoGCN++* code. The adjacency matrix plays a critical role in capturing the spatial configuration of the skeleton and is essential for tasks like action recognition and motion analysis.



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3.4: Example of graph and its adjacency matrix: rows and columns indexes represent nodes. If cell (i,j) is a 1, it means there is an edge between node i and node j

In the MVN skeleton model, bones are represented by ordered pairs of joints, and only the inward connections need to be explicitly defined. From these, the class computes the outward and neighbor connections, ensuring that the full graph structure is efficiently constructed. The details of this model can be found in the MVN User manual, which provides the specifications for the joints and bones used in the software.

Fine Tuning

Fine-tuning is the process where a pre-trained model is adapted to perform well on a new, often smaller, dataset by continuing the training process, but typically only on the final layers of the model. The idea is to leverage the knowledge the model has already learned from a larger and more generic dataset and adjust it to better suit the specific task or data to be used for inference. This approach is highly efficient because it allows you to adapt powerful, pre-trained models to your own needs without needing to collect large amounts of data or spend excessive time training from scratch. In our case, this is really helpful to address the problem of inter-class similarity, being capable of discrimate between similar movements and also to have a better knowledge of actions which are performed in a slightly different way compared to the ones seen by the model during training.

Fine tuning has three main steps:

- **Pre-training on a large dataset:** The model is first trained on a large, diverse dataset. This pre-training allows the model to learn general features.
- **Freezing earlier layers:** In the fine-tuning phase, most of the earlier layers, which capture low-level features, are frozen, meaning their weights are kept unchanged. Only the final layers, which are more task-specific and responsible for the final outcome of the network, are updated based on the new dataset.
- **Adapting to new data:** The model is then trained on a smaller, task-specific dataset. During this process, only the final few layers are fine-tuned, allowing the model to specialize in the new task while conserving computational resources by avoiding full retraining.

More practically, the first point is achieved with the training procedure of the model on a large dataset. The second step needs an adjustment to the function which is responsible to load the model, which is called during the initialization of the main process. Inspecting the model object, we can see the name of the blocks and the layers of the architecture, so, after freezing all them, we can unlock only the ones we are interested in continuing to train. Referencing to the model structure, we decided to fine-tune the classification decoder and the last three final convolutional layers. The code to achieve this is the following, and it is inserted in the function *load_model()*, after the loading of the model and before the loading of the pre-trained weights:

```python
# Freeze all layers first
for param in self.model.parameters():
    param.requires_grad = False

# Unfreeze only the ‘cls_decoder‘ layers
for param in self.model.cls_decoder.parameters():
    param.requires_grad = True

# Unfreeze the last Conv1d layers (c7, c8, c9 as the last layers based on model architecture)
for conv_layer in [self.model.c7, self.model.c8, self.model.c9]:
    for param in conv_layer.parameters():
        param.requires_grad = True
```

Code 3.1: Freezing layers for fine tuning

Then, the third point in achieved executing training, we only need to change some parameters: firstly we need to load the correct dataset to use for fine-tuning, and then we need to load weights of a pre-trained model. The training procedure will load the model, freeze the selected layers, load the pre-trained weights, load the new dataset and start the standard training process.

Real Time Evaluation Function

In our real-time action recognition system, we developed a function that processes a continuous stream of skeleton data, providing instant classification of movements without relying on pre-recorded sequences. The system is designed to handle incoming frames of skeleton data and stacking them into a buffer. When the buffer reaches a predefined size, the frames are subjected to a series of preprocessing steps including scaling, transformation, and normalization. Once these transformations are complete, the data is passed to the model, which performs action classification in real-time. This approach allows for the efficient handling of skeleton data streams while maintaining high classification accuracy.

The high-level pseudocode is shown in algorithm4, and outlines the steps involved in this process.

Algorithm 4 Real-Time Action Recognition

```
Initialize an empty buffer frame_buffer.
Set the model to evaluation mode.
while the stream is active do
    if frame_buffer is not full then
        Receive a new frame of skeleton data.
        Append the new frame to frame_buffer.
    end if
    for each new frame received do
        Append the frame to frame_buffer.
        if the buffer exceeds the predefined size then
            Remove the oldest frame from frame_buffer.
        end if
    end for
    if frame_buffer is filled to the required size then
        Preprocess the data: apply scaling, transformation, and normalization.
        Feed the preprocessed data into the model.
        Retrieve the predicted label from the model's output.
        Display or store the classification result.
    end if
end while
```

3.3 Ergonomics assessment

We developed a simple ROS Node capable of processing skeleton data in real-time to provide an ergonomic evaluation. Each skeleton is treated independently from previous ones, making this approach a natural fit for tools like RULA, which evaluates postures based on static snapshots. In our implementation, we adapted the traditional RULA tables and scoring system to not only generate a general risk factor but also to provide detailed feedback on how much each joint deviates from its optimal ergonomic range. The node works by reading relative joint angles from a skeleton, which is streamed in real-time over a dedicated ROS topic, and comparing those angles to predefined ergonomic thresholds for each joint.

The core of the analysis lies in a function which processes the joint angles for different parts of the body, such as the upper arms, lower arms, wrists, neck, and trunk. For example, the *locate_upper_arm_position* function determines whether the upper arm's angle falls within the ergonomic range of -20 to 20, which is considered a physiological interval and therefore low risk for the shoulder. If the joint angle is outside this range, the function calculates the degree by which the arm is out of the safe range, returning a zero if it is within bounds or the deviation value if its not. The same principle is applied to other joints, using the range described in the RULA worksheet, reported previously in figure 2.2.

The node outputs a more granular evaluation than a standard RULA score. For each joint, it either outputs a zero if the joint is within its ergonomic range or a value in degrees showing how much the joint is outside the no-risk interval. This detailed joint-specific feedback offers more meaningful information than a single risk score and can be used by industrial robots to dynamically adjust the work environment, as we will better discuss in section 6.3. For instance, robots could adapt in real-time the height of a component to prevent workers from raising their arms too high or change object orientation to avoid wrist twisting. By providing this continuous, real-time feedback, the ROS node enables robots to help workers maintain ergonomic postures, thereby reducing the risk of musculoskeletal injuries.

4

Experiments and results on AnDy dataset

This chapter provides a comprehensive exploration and motivation of the experiments conducted using the AnDy dataset, alongside their results and a critical analysis of the model's performance. The AnDy dataset served as a crucial foundation for training and initial validation of the action recognition model. It also provided a valuable benchmark for evaluating the models ability to learn various actions and generalize its performance to new subjects. Additionally, the model trained on the AnDy dataset will be used in further experiments, serving as a base for fine-tuning, both of which will be presented in the following chapter. We begin by detailing the experimental setup, including the processes of data preparation, training, and evaluation, to assess the models ability to generalize across different subjects and environments.

The results of these experiments, presented through both quantitative metrics and qualitative insights, offer a deeper understanding of the models behavior when applied to pre-existing public datasets. We analyze how the model handles variability in movements and how well it recognizes complex and subtle actions that are critical for accurate classification. The chapter is structured to not only present the results but also to provide a critical discussion on the models strengths and limitations, laying the groundwork for potential improvements. This sets the stage for evaluating the model's performance in real-world scenarios, which will be addressed in the following chapter.

4.1 Dataset

For the purposes of the study, we required a dataset collected using the Awinda system, to replicate data in our laboratory, and which contains actions attributable to an industrial context. The research ended with the choice of the AnDy Dataset [13].

4.1.1 AnDy Dataset

The AnDy Dataset is a comprehensive collection of human motion data, specifically collected for research in industrial settings, with a focus on enhancing workplace ergonomics and improving the interaction between humans and collaborative robots. The dataset was meticulously designed to address the need for high-quality labeled data in the fields of robotics and human motion analysis, particularly in the context of industrial tasks. The dataset captures a range of postures and actions commonly observed in industrial environments, such as those found on assembly lines. These activities were selected and structured to reflect realistic industry-related scenarios. They are:

- Re (Reach): Moving an arm towards a target without holding an object.
- Pi (Pick): Picking up an object, starting from the moment of contact with the object until the arm stops moving relative to the body.
- Pl (Place): Placing an object, similar to a reach but with an object in hand.
- Rl (Release): Returning the arm to a neutral position after manipulation.
- Ca (Carry): Transporting an object, starting after picking it up and ending before placing it.
- Fm (Fine Manipulation): Dexterous manipulation of an object.
- Sc (Screw): A specific type of fine manipulation involving a rotational screwing motion.
- Id (Idle): No activity with the hands.

The AnDy data collection procedure involves six sequences of this actions designed by the authors, and each of the thirteen participant performed three of them at random, each repeated five times. This approach ensured diversity in the data, and resulted in a total of 15 trials per participant, with each trial lasting approximately 90 seconds. In addition to action labeling, the AnDy Dataset is distinguished by its detailed posture annotations. Three independent annotators labeled the data, ensuring high reliability and accuracy in the recorded actions. These posture annotations were inspired by the Ergonomics Assessment Worksheet (EAWS) [33], a tool widely used in industries to evaluate the ergonomic impact of various postures. This dual annotation of actions and postures makes the AnDy Dataset particularly valuable for research focused on improving workplace ergonomics and developing robotics solutions that enhance human well-being in industrial contexts. The dataset provides a rich variety of data formats to facilitate reuse in different research domains. It includes whole-body kinematics recorded, finger pressure force, video recordings, and comprehensive annotations.

4.1.2 Data rearrangement

As input for the model, we needed sequences of skeletons structured in a specific format and annotated with their labels and stored in JSON files, one for each action. So, starting from the MVNX acquisitions of AnDy Dataset and their labeling - labeling files are available in the dataset - we created new files to rearrange skeleton data and stored them as expected by the model. The flow chart of the data rearrangement is visible in figure 4.1

The modification of AnDy dataset was performed by a Python script that reads as input the MVNX acquisitions and gives as output the new annotations ready to be used by the neural network. The logic is straight-forward: the algorithm reads the labeling file and looking for changes in the labels it stores timestamps and labels, obtaining as result a time segmentation of the acquisitions corresponding to single actions. Then, it extracts skeletons joints positions corresponding to an action, which means skeletons whose timestamps are inside each of the intervals previously computed. A dictionary is created with the structure expected from the model. After all the segments are processed and the annotations created, they are stored as a JSON file. Pseudo-code of the dataset creation algorithm could be seen in algorithm 5.

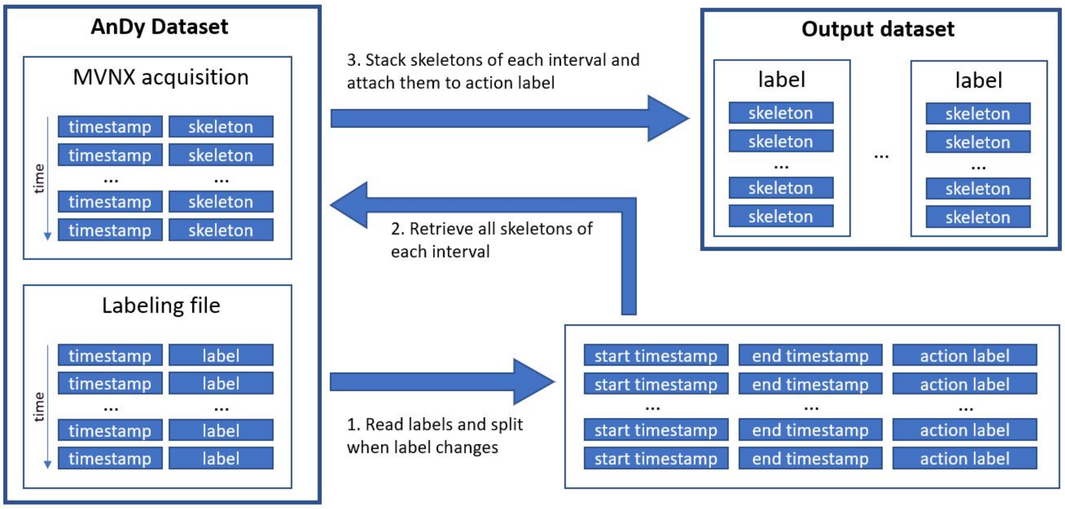

AnDy Dataset
MVNX acquisition
timestamp
skeleton
time
Labeling file
timestamp
label
time
3. Stack skeletons of each interval and attach them to action label
2. Retrieve all skeletons of each interval
1. Read labels and split when label changes
Output dataset
label
skeleton
start timestamp
end timestamp
action label

Figure 4.1: The high-level workflow of the data rearrangement from AnDy to the right format for our model

Algorithm 5 Dataset creation

```
Initialize an empty list annotations.
Assume the labels for each timestamp are given.
for acquisition in acquisitions do
    Identify changes as the timestamps where label changes occur, along with the associated labels.
    Set last_timestamp to 0.
    for current_timestamp in changes do
        Extract action_frames between last_timestamp and current_timestamp
        Initialize an empty list skeletons.
        for frame in action_frames do
            Append the skeleton data of frame to skeletons.
        end for
        Create annotation as a list containing skeletons and the current label.
        Append annotation to annotations.
        Update last_timestamp to current_timestamp.
    end for
end for
Store annotations
```

The format used to store the annotation is the same for every experiment presented in this thesis. Since the model was originally tested on the UCLA dataset [43] it was by default able to deal with the annotation format used by that dataset, and we exploit it for our purposes. The format is a folder containing the annotations, one stored in a different JSON file. The structure of the JSON is very simple, it is has only three fields: *file_name*, *skeletons*, and *label*. *file_name* is just a string with the name of the file, *skeletons* is an array containing the sequence of all skeletons, with shape *T x V x C*, where T (time) is the length of the sequence, V (vertices) the number of joints tracked and C (channels) the number of coordinates for each joint. *label* is just a number which identify the action related to this annotation, and in our case it goes from 0 to 7. More particularly, label 0 is associated with "Idle", 1 with "Reaching", 2 with "Picking", 3 with "Placing", 4 with "Release", 5 with "Carry", 6 with "Manipulate" and 7 with "Screwing".

```
{
    "file_name": "sa_s1_r3_1_a0_id6f6", //name of the file
    "skeletons": [ //array of skeleton
        [ // array of joints
            [0.05901, 0.033815, 0.9560675], // joint's coordinates
            ... ,
            [0.06543, 0.034362500000000004, 1.1704875000000001]
        ],
        ... ,
        [
            [0.055725000000000004, 0.0364675, 1.3764575],
            ... ,
            [0.0869425, 0.037567500000000004, 1.6061874999999999]
        ]
    ],
    label: 0 // label of action represented by skeletons
}
```

Code 4.1: Snippet of example of annotation file

The following step is to split all the annotations into train and test data, and it is accomplished using another python script which read all the files and split all the files into two arrays following different logics, for example randomly or by name, to achieve subject cross-validation. The script while splitting can consider a balance factor that for our case was 0.7, so tho have 70% of the annotations in the training set, and the other 30% in the validation and testing set.

4.2 Train and Evaluation

The first experiment is to train and evaluate the model on the AnDy dataset. This will help understanding how accurate the model can be on high quality annotations, and underline immediately strengths and weaknesses on the action recognition of the eight actions of AnDy.

This experiment involves all the trials available within the AnDy dataset, shuffled and randomly split between train set and validation set, with 70% of the annotations in the training set and the other 30% on the test set.

We used the following command for the training, provided by the model's authors and adjusted for our case:

```
python main.py --half=True --batch_size=32 --test_batch_size=64 --
    step 50 60 --num_epoch=70 --num_worker=4 --dataset=andy --
    num_class=8 --datacase=andy --weight_decay=0.0003 --num_person=1
    --num_point=23 --graph=graph.andy.Graph --feeder=feeders.
    feeder_andy.Feeder --base_lr 1e-1 --base_channel 64 --window_size
    120 --lambda_1=1e-1 --lambda_2=1e-3 --lambda_3=1e-3 --n_step 3
```

Code 4.2: Command to train model on AnDy dataset

As you can see, there are many hyper-parameters, and now we detail them explaining their meaning and their role.

- `-half=True`: Activates mixed-precision training, which allows faster computations and reduced memory usage by using half-precision floating-point operations where appropriate.

- `-batch_size=32`, `-test_batch_size=64`: These define the number of samples per batch for training and testing respectively. A larger test batch size can help speed up evaluation since it is less memory-intensive than training. This parameter is taken from the authors of the model.

- `-step 50 60`: Specifies at which epochs the learning rate should decay, helping to fine-tune the model by gradually reducing the learning rate to stabilize training. In this case, the learning rate decays at epoch 50 and 60. This parameter is taken from the authors of the model.

- `-num_epoch=70`: Sets the number of training epochs to 70, defining how many times the model will iterate over the entire training dataset.

- `-num_worker=4`: Defines the number of worker threads used for loading data during training. Increasing this number can speed up data loading, especially on systems with multiple cores.

- `-dataset=andy`, `-datacase=andy`: Specifies that the `andy` dataset and its corresponding data configuration will be used during training and testing. This parameter needs to have an exact match on the `data` directory.

- `-num_class=8`: The number of output classes in the classification task, indicating that the model will classify the data into one of 8 possible categories.

- `-weight_decay=0.0003`: A regularization parameter used to prevent overfitting by penalizing large weights during training, helping to maintain generalization. This parameter is taken from the authors of the model.

- `-num_person=1`, `-num_point=23`: The number of people and the number of joint points considered in the skeleton data. These parameters define the structure and representation of the input data.

- `-graph=graph.andy.Graph`: Defines the graph structure used in the model to represent the relationships between the joints. In this case, the `andy` graph is employed. In this case we load the graph we modify the suit the MVN skeleton.

- `-feeder=feeders.feeder_andy.Feeder`: Specifies the data feeder class, which handles data loading, and preprocessing.

- `-base_lr 1e-1`: Defines the initial learning rate for the optimizer, setting the pace at which the model adjusts weights during training. This parameter is taken from the authors of the model.

- `-base_channel 64`: Sets the base number of channels for the models convolutional layers. This determines the width of the network and its capacity to learn features. This parameter is taken from the authors of the model.

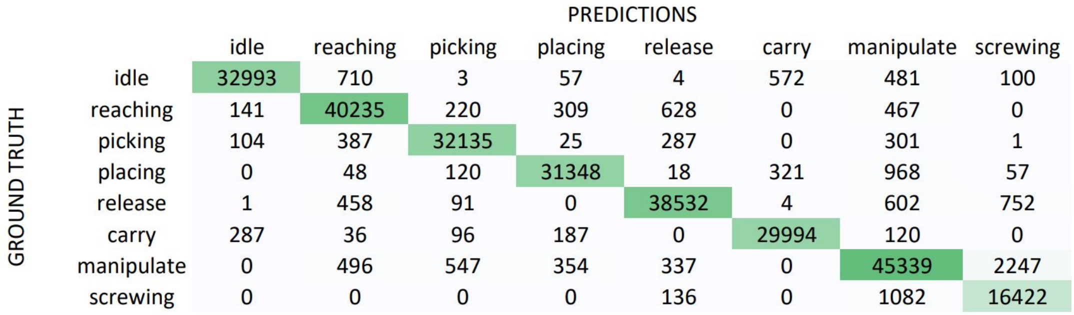
- `-window_size 120`: Refers to the size of the sliding window used for temporal data sequences, controlling how much context from the time dimension is used. The choice of 120 depends on the frame rate of our skeletons, which is 240Hz. From the results published in [44], we know that the temporal windows to perform accurate action recognition is between 250ms and 500ms, and that larger windows don't lead to a more precise classification. so we choose 120 frames, which means considering windows of 0.5 seconds.

- `-lambda_1=1e-1`, `-lambda_2=1e-3`, `-lambda_3=1e-3`: These hyper-parameters specify different regularization terms or weights for specific loss components, balancing their contributions during training. These parameters are taken from the authors of the model.

- `-n_step 3`: Defines the number of steps in sequence prediction, referring to how many future time steps the model aims to predict.

4.2.1 Results and discussion

First significant results came from the experiments on AnDy dataset. As you can see in Figure 4.2 and Table 4.1 the model achieves high performance across all eight action labels, with accuracy rates consistently exceeding 0.90 for each category. The model excels particularly in recognizing the labels "Picking" and "Carry," with respective accuracies of 0.97 and 0.98. This suggests that the model effectively distinguishes between these actions even when they involve similar movements. For instance, the high accuracy of "Reaching" (0.96) and "Placing" (0.95) demonstrates the model's ability to differentiate between these actions, which often appear visually similar. However, the model struggles a bit more with distinguishing between "Manipulate" and "Screwing," as evidenced by their lower accuracy of 0.92 and 0.93, respectively. The confusion between these two labels highlights the challenges in classifying actions that require similar hand movements and object interactions. Overall, the results reflect a robust performance in action recognition, with a total accuracy of 95%.



PREDICTIONS

GROUND TRUTH

	idle	reaching	picking	placing	release	carry	manipulate	screwing
idle	32993	710	3	57	4	572	481	100
reaching	141	40235	220	309	628	0	467	0
picking	104	387	32135	25	287	0	301	1
placing	0	48	120	31348	18	321	968	57
release	1	458	91	0	38532	4	602	752
carry	287	36	96	187	0	29994	120	0
manipulate	0	496	547	354	337	0	45339	2247
screwing	0	0	0	0	136	0	1082	16422

Figure 4.2: Confusion Matrix of inference on AnDy dataset

Idle	**Reaching**	**Picking**	**Placing**	**Release**	**Carry**	**Manipulate**	**Screwing**	**Total**
0.94	0.96	0.97	0.95	0.95	0.98	0.92	0.93	0.95

Table 4.1: Accuracy of each label for AnDy dataset evaluation

Beside of accuracy, there are other interesting metrics that give us details about the model behaviour. The AUC is a performance metric often used to evaluate classification models by summarizing the trade-off between true positives and false positives. In the context of binary classification, AUC is computed based on the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (sensitivity) against the false positive rate (1 - specificity, where sensitivity is the true negative rate) as the classification threshold is varied. In multi-label classification, AUC is extended to measure the model's ability to discriminate between multiple classes, calculating the AUC for each label separately and then averages the results. A model with an AUC of 0.5 is equivalent to random guessing, while an AUC of 1.0 represents perfect classification. The metric is meaningful because it provides a single value that summarizes the trade-offs between true positives and false positives, making it especially useful in imbalanced datasets.

We reported AUC plots for AnDy experiment in figure 4.3, where the left plot shows the training AUC over the course of training steps, while the right plot shows the evaluation AUC. The training AUC plot demonstrates a steady increase early on, reaching around 0.95, indicating that the model is learning to discriminate between classes during training. The evaluation AUC plot follows a similar trend, starting lower but stabilizing around 0.93, which is slightly lower than the training AUC. This suggests that the model generalizes well to unseen data, though there is minor fluctuation in the evaluation AUC, indicating variability during evaluation. Both plots show strong overall performance with minimal overfitting, as the gap between training and evaluation AUC is small.

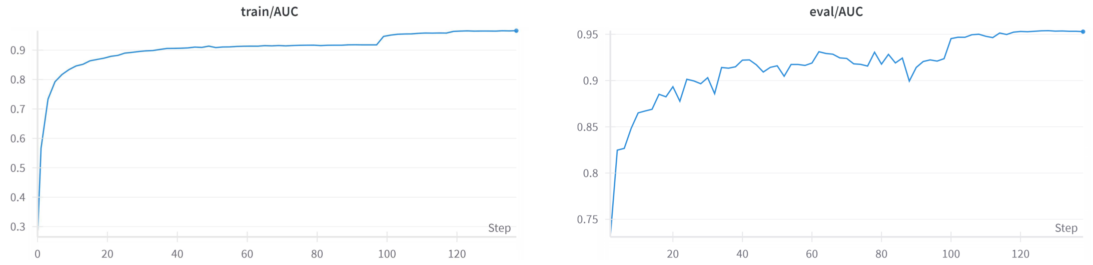


Figure 4.3: AnDy dataset train and evaluation AUC

4.3 Subject cross-validation

The second set of experiments we conducted involved subject cross-validation to evaluate how well the model generalizes to unseen subjects. This approach allowed us to investigate the model's performance when faced with new data not involved in the training phase. For this, we divided the twelve participants in the AnDy dataset to train six different models. Each model was trained on a set of ten participants, with the remaining two reserved for evaluation. For instance, in the first configuration participants 1 and 2 were used for evaluation, while the other ten were used for training. This process was repeated across the six models, with the final model being trained on the first ten subjects and evaluated on the last two. These experiments were crucial not only for assessing the models ability to generalize to new subjects but also for identifying potential overfitting or underfitting issues during training. By examining the model's performance across different subject combinations, we could gain insights into its robustness and adaptability.

4.3.1 Results and discussion

The next important result is from the cross-validation experiment. Here, the main objective was not to asses whether the model is capable to learn and recognize actions, but to emulate and investigate how the model deals with new subjects. The cross-validation results, shown in figure 4.2, provide valuable insights into the model's performance when evaluated on subjects that were not part of the training data. The accuracies range from 0.87 to 0.95 across different subject pairs, with the highest accuracy observed for Subjects 11 and 12 (0.95). This suggests that the model generalizes well to new subjects, indicating a low risk of overfitting, where the model would perform exceptionally well on the training data but poorly on unseen data. The relatively consistent accuracy across various subject groups also implies that the model has not fallen into the trap of underfitting, which would be reflected in uniformly low performance. Instead, the results demonstrate that the model effectively captures the underlying patterns in the action recognition task, suggesting a robust ability to recognize actions even when presented with novel subjects. These findings reinforce the model's effectiveness and reliability in real-world applications, where variability in the subject population is expected.

Subjects for validation	**Accuracy**	**Subjects for validation**	**Accuracy**
Subjects 1,2	0,88	Subjects 3, 4	0,87
Subjects 5,6	0,87	Subjects 7,8	0,90
Subjects 9,10	0,93	Subjects 11,12	0,95

Table 4.2: Accuracy of cross-subjects validation experiments between the 12 participant of AnDy.

In figure 4.4, we show the AUC metrics for both training and validation during the first experiment of the cross-validation set, where two subjects are used for validation and the remaining ten for training. This figure represents the first of six experiments, and since the AUC plots follow similar trends across all experiments, the observations here are generally applicable.

The training AUC demonstrates an initial rise before leveling off around 0.97, indicating that the model quickly learns to distinguish between the multiple labels in the training data and improves steadily as training progresses. The high final value suggests that the model has learned to separate the labels effectively within the training set.

On the other hand, the validation AUC shows more fluctuation, starting around 0.76 and improving to about 0.87 by the end. These fluctuations reflect the model's varying performance on the validation set at different stages of training. This variability is common in real-world data, where the validation set may contain more challenging or ambiguous examples, and the model's performance can vary depending on which patterns it has learned so far. However, the overall upward trend suggests that the model is progressively learning to classify correctly actions and generalizing better to unseen data, even though the validation AUC remains slightly lower than the training AUC. The differences from the final values is about 10% which mean there is a small overfitting of the model, and the generalization over evaluation data could be slightly improved, even though the performance is already high.

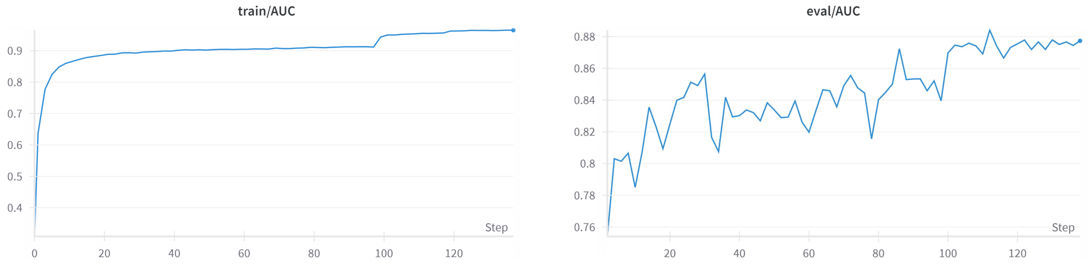


Figure 4.4: Train and evaluation AUC of cross-validation with subjects 1 and 2 of AnDy

5

Experiments and results on custom dataset

This chapter focuses on the experiments conducted using the custom data acquired in our laboratory, designed to evaluate the action recognition models performance in real-world, industrial scenarios. The dataset, which captures assembly tasks performed by multiple subjects, serves as a challenging benchmark for testing the model's adaptability, generalization, and fine-tuning capabilities. Throughout the chapter, we emphasize the rationale behind each experimental setup, explaining how each was intended to test specific hypotheses regarding the models performance in diverse contexts. We outline the processes of data collection and annotation, detailing the specific experimental setups employed to assess the models functionality in this applied setting. By illustrating the step-by-step processes involved in training and evaluating the model, we aim to provide a comprehensive understanding of the experimental workflow.

The results presented in this chapter include a detailed analysis of the model's quantitative performance and its ability to handle new, unseen data. We complement these metrics with qualitative observations, emphasizing how the model performs in recognizing the intricate and dexterous movements associated with real-world tasks. By delving into these different aspects, the chapter not only highlights the technical performance of the model but also addresses broader questions about its generalization capabilities, robustness to new data, and practical applicability in real-world scenarios. Through a critical discussion, we highlight both the successes and limitations of the model in this practical application, with a particular focus on its robustness and generalization capacity. The goal of this comprehensive analysis is to provide a deeper understanding of the model's functionality and its potential to deliver accurate action recognition across diverse environments and tasks. This evaluation also lays the groundwork for identifying future improvements and optimizations, discuss later on section 6.3 to enhance the model's effectiveness in even more challenging applications. The chapter concludes with an analysis of the models potential for further development, suggesting avenues for future refinement and optimization based on the insights gained from these experiments.

5.1 Custom data collection

To further enhance the evaluation of our action recognition model, we developed a custom dataset tailored to a specific real-world scenario: the assembly of a chair. Figures 5.1 and 5.2 show a stage of the assembly of the chair and the laboratory setup used for the dataset collection. This dataset is crucial for testing the model's performance in an environment distinct from the AnDy dataset, ensuring its ability to generalize across different tasks. Even if it is always an assembly process, as actions in AnDy, the scenario is changed and variations in the movements are introduced. In fact, the chair assembly process involves a complex and sequential series of actions that, while similar to the original AnDy data in nature, are executed in a significantly different manner. This divergence presents a novel challenge for the model, offering a valuable opportunity to assess its adaptability and robustness in handling variations in how the same actions are performed.

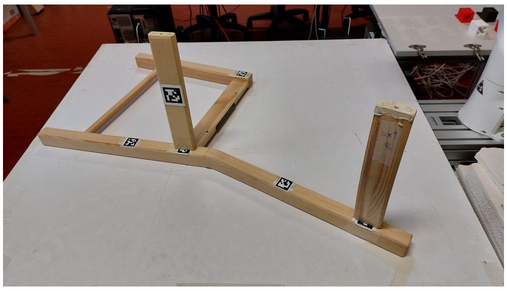


Figure 5.1: The chair used for our sequence of actions, at an intermediate stage of the assembly process

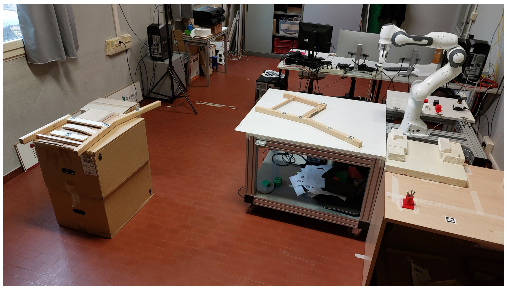


Figure 5.2: Experimental setup: in particular this is the initial situation, with one side part on the work table, other components on the left and screws is the red container on the right

The custom dataset was created by simulating a complete chair assembly process, designed to represent a practical test case for the model. The chair used for the experiment consists of two side parts, visible in Figure 5.3 which include the legs, and four crossbars that connect these side parts, visible in Figure 5.4. All the pieces are held together by three screws.

The experimental setup, illustrated in Figure 5.5, begins with one side part of the chair already in place, while the remaining five parts must be retrieved and assembled. These parts are placed on a table approximately three meters away from the assembly station. The sequence has been carefully designed, and the actions to be performed in order are as follows:

- Idle position
- 5x reach, pick, carry, and place part of the chair.
- 3x reach, pick, and place screws.
- Pick the Allen key.
- 3x screwing.
- Place the Allen key.
- Idle position

This sequence of actions strikes a balance between the original AnDy sequences and a real-world assembly process, providing a meaningful and realistic evaluation scenario for the model.

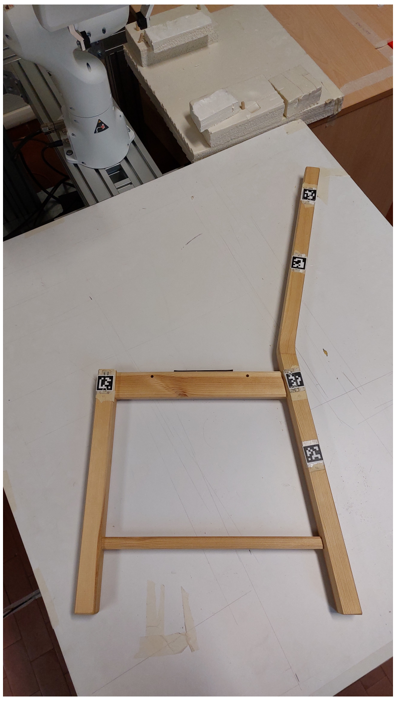


Figure 5.3: Example of side part component of the chair

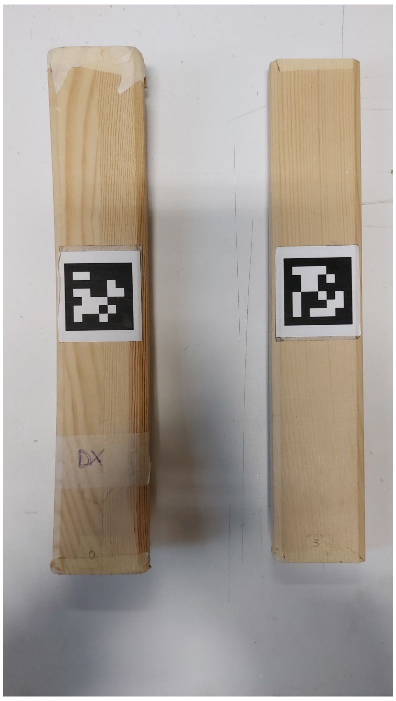


Figure 5.4: Example of crossbars component of the chair

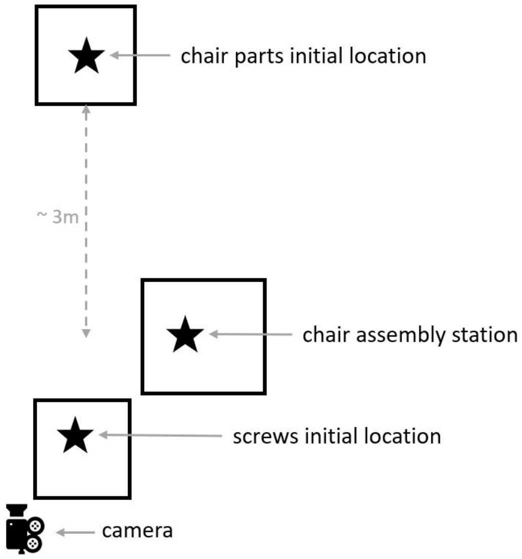


Figure 5.5: Laboratory setup schema for the acquisition of the custom dataset

We collected data from 7 subjects, each of whom performed the entire assembly sequence 10 times. The data was captured using a combination of inertial sensors and a single RGB camera, ensuring comprehensive coverage of the movements. All recordings were stored in ROS bag files, containing topics for RGB video, timestamps, and skeletal data. These recordings were meticulously annotated using a custom Python script, which assigned labels to each frame of the RGB video data and added a new topic to the ROS bag containing the corresponding labels for each timestamp. The ROS bag was then processed using a modified version of the script described in section 4.1.2, which was used to generate the necessary JSON files for further experimentation. This allowed us to maintain consistency with the previously established data preparation procedures, ensuring the dataset was ready for use in subsequent model evaluations.

Our dataset introduced new challenges that required the model to adapt to a larger variety of action execution. Unlike the more structured sequences in the AnDy dataset, our data involved a greater degree of variability in how the same actions were performed. This increased complexity hindered the model's ability to generalize and accurately classify the actions, yet it aligns more closely with real-world industrial scenarios. In such environments, each worker develops unique movement patterns based on their individual preferences, physical characteristics, and the specific demands of their tasks. Therefore, incorporating this diverse and realistic data is crucial for our study, as it enables the model to better reflect the variability encountered in actual industrial settings. This adaptation is essential for enhancing the model's applicability and effectiveness in monitoring and improving worker performance, ultimately contributing to more efficient operations and improved safety protocols. As an example of this variability in the subjects' actions, the reach action was not limited to a single form, as shown in Figure 5.6. Depending on the scenario, it may involve moving closer to or further away from the body, using either one hand or both hands. The specific object to be picked and its position further influenced how the reaching action was executed, introducing additional variability for the model to account for. Another notable example is the *placing* action, as shown in figure 5.7, where subjects exhibited different strategies. A subject might use their dominant hand, either the left or the right, or both hands together, depending on the ease and comfort of the task. The model had to recognize and differentiate between these variations, complicating the learning process. Additionally, the "screwing" action involved even more subtle distinctions, as you can notice in figure 5.8. Not only did the hand positions vary, but the orientation of the tool also shifted based on the subject's preferences. These small yet critical variations in execution posed significant challenges for the model's ability to correctly identify and classify the actions across different subjects and instances.

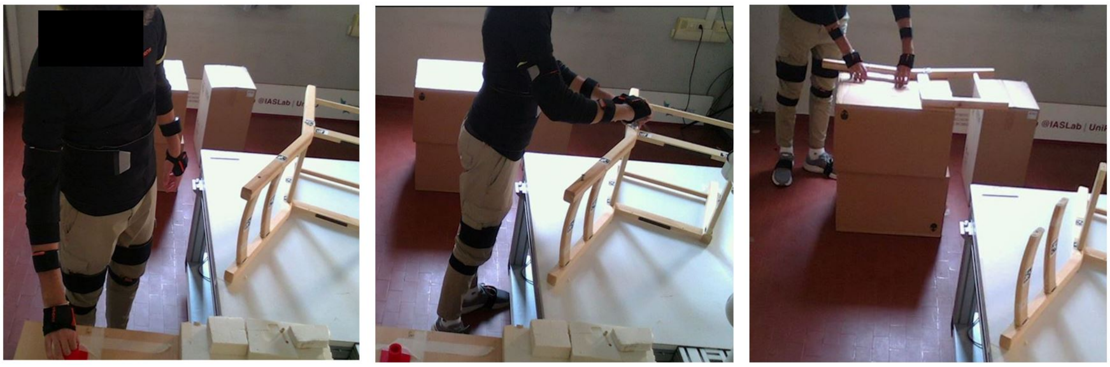


Figure 5.6: Examples of variations of reaching action: reaching a screw with one hand, reaching with screwdriver, reaching chair component with two hands.

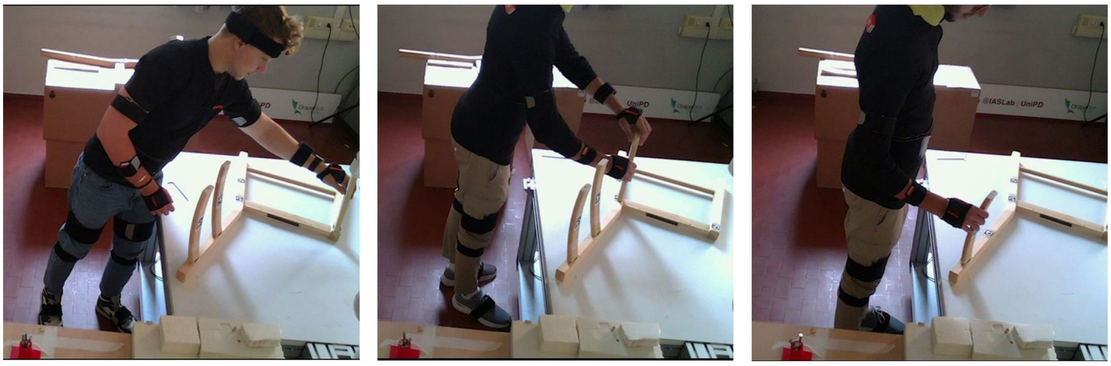
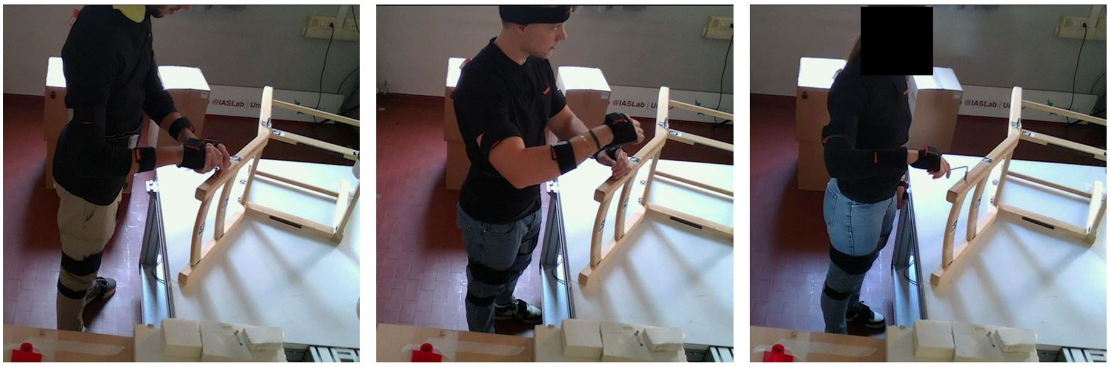


Figure 5.7: Examples of variations of placing action: place object with left hand far away from the body, place object with two hands, place object with right hand close to the body.



[redacted]

Figure 5.8: Examples of variations of screwing action: two hands with horizontal key, two hands with vertical key, one hand

5.2 Inference on custom dataset using AnDy weights

The experiment on the data collected in our lab used the first two subjects for evaluation to see how performance changes under different model conditions. This will hold for all the experiment on our data, to have a more objective benchmark when comparing different models: we will see for example how classification accuracy will increase on these subjects under different configurations.

After creating our custom dataset, we tested the model trained on the AnDy dataset. We tried firstly to test on data annotated in a very strict way, but this resulted in a low accuracy, so we relabeled the data trying to match more the actions in AnDy. At the end, we tried also to consider only the first part of the sequence, which means to test the model only on the pick and place of the chair components, excluding the screwing part. This is because the action of screwing is significantly different between AnDy, where it is the fasten of a nut on a shelf, and our sequence, where it is about the screwing of a screw with an Allen key.

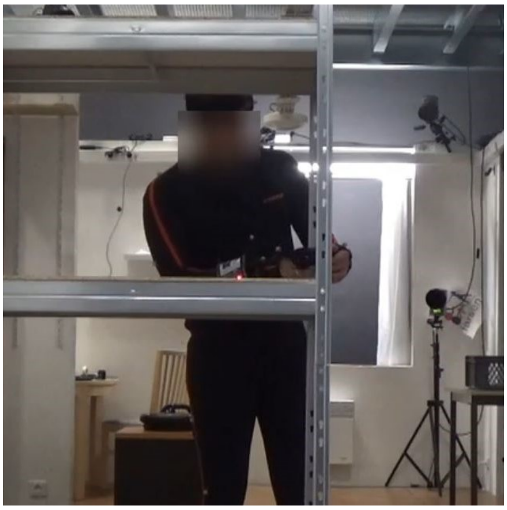


Figure 5.9: Screwing action in AnDy dataset

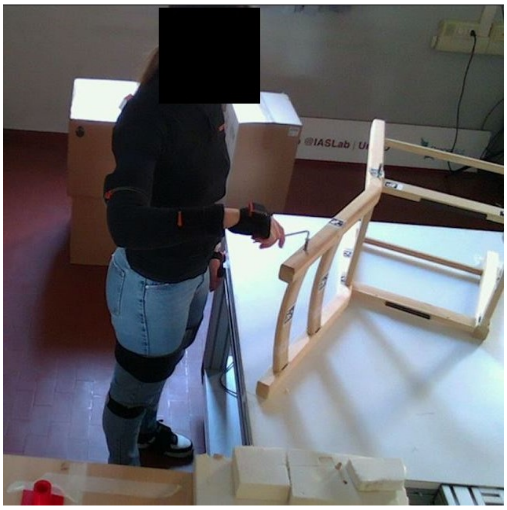


Figure 5.10: Screwing action in our routine

5.2.1 Results and discussion

This set of experiments was conducted on our custom dataset, using the model trained on AnDy and evaluated on subjects 1 and 2. The results, shown in Figure 5.11 and Table 5.1, underline several issues with the model's performance. The overall accuracy for each label varies significantly, highlighting areas where the model struggles to generalize to new data.

For the label "Idle," the model achieves an accuracy of only 0.56, misclassifying many samples as "Carry". This happens due to similar body motion, since both these actions include the walking phase between stations, and the differences are nuances in the arm position.

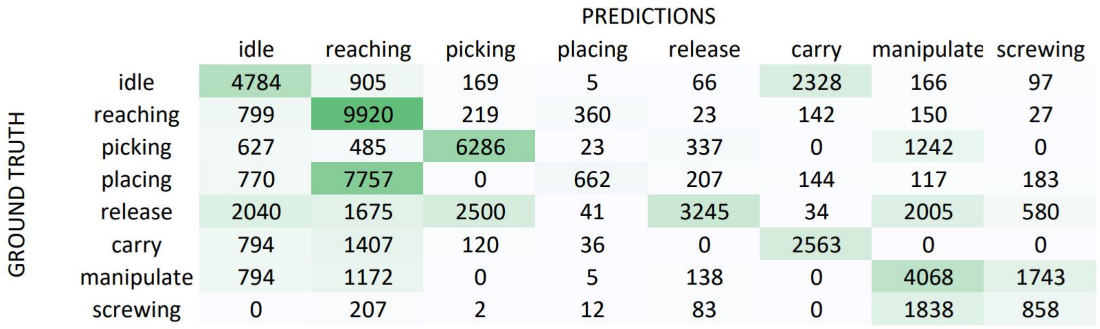
"Reaching" shows a much better performance with an accuracy of 0.85, suggesting that the model can effectively recognize this action. However, some confusion still occurs with "Picking" and "Placing," which is understandable given the similarity between these actions in terms of arm and hand movements. This holds also for the label "Picking", which achieves a moderate accuracy of 0.70. though the confusion with the manipulation action is notable.

One of the most concerning results is for the label "Placing," which has an extremely low accuracy of 0.07. The vast majority of "Placing" samples are being confused with "Reaching". This suggests that the model is almost entirely failing to recognize "Placing" as a distinct action, treating it more like the initial reach toward the object, which is a critical problem in the model's performance. This problem stems from the difference between the place action of AnDy and the one performed in our laboratory.

Similarly, "Release" is another weak point, with an accuracy of 0.27. The model confuses this action with "Idle" and "Carry," suggesting that the transition from object handling to releasing is not being effectively captured by the model, perhaps due to the fact that this movement is often performed at the same time of some other body movements.

"Carry" has an accuracy of 0.52, which is somewhat better than the previous labels but still underperforms. The confusion with "Idle" and "Reaching" points to the model's difficulty in recognizing the continuity of carrying actions.

"Manipulate" and "Screwing" also show concerning results, with accuracies of 0.51 and 0.29, respectively. "Manipulate" is frequently confused with "Screwing," which is because, they have similar body position and subtle hands movements, differentiated only by different object handling. In addition, AnDy participants perform the screwing action of a horizonal axis, while for the purposes of the chair assembly we screw nuts vertically. This means that the model has to evaluate an action different from the one used during training.



		PREDICTIONS							
		idle	reaching	picking	placing	release	carry	manipulate	screwing
GROUND TRUTH	idle	4784	905	169	5	66	2328	166	97
	reaching	799	9920	219	360	23	142	150	27
	picking	627	485	6286	23	337	0	1242	0
	placing	770	7757	0	662	207	144	117	183
	release	2040	1675	2500	41	3245	34	2005	580
	carry	794	1407	120	36	0	2563	0	0
	manipulate	794	1172	0	5	138	0	4068	1743
	screwing	0	207	2	12	83	0	1838	858

Figure 5.11: Confusion Matrix of inference on subjects 1 and 2 of custom dataset, evaluated using weights computed after 70 epochs of training on 70% of AnDy

Since accuracy of the model is very low, we try re-annotate all the data with more focus on labeling actions to match the ones on the training dataset. This means not considering only the exact movements, but more the intention behind.

Idle	Reaching	Picking	Placing	Release	Carry	Manipulate	Screwing	Total
0.56	0.85	0.70	0.07	0.27	0.52	0.51	0.29	0.47

Table 5.1: Accuracy of each label for custom dataset evaluation

We provide a practical example for more clarity: consider the action of move the screwdriver to the screw and start screwing. For the definition provided in the AnDy paper, since we have an object in the hand it should be "Placing", but since the intention is not to drop something, the movement is more similar to "Reaching". Then, consider you start screwing and then you help yourself with the other hand to keep the screw in place or adjust the parts positions: this should be "Manipulation" since it is a free movement of the arm, but at the end it is part of the screwing action. After this relabeling of our sequences, even if we obtain just a small increase of the overall accuracy, from 0.47 to 0.50, there are some interesting improvements to notice: the biggest is for the "idle" label, that reach an accuracy of 0.76 starting from 0.56. Also reaching, placing, release and manipulate have some smaller gains, while carry and screwing remain stable.

In summary, while the model performs well for certain actions like "Reaching," it struggles significantly with more nuanced or transitional actions like "Placing," "Release," and "Screwing." This indicates potential overfitting to the training data and poor generalization to the custom dataset, where specific movements are either too subtle or too similar for the model to differentiate accurately. The confusion between labels suggests the need for further refinement in the model architecture or the training process, particularly for actions that involve fine motor control or transitions between states.

5.3 Inference on custom dataset using AnDy weights after fine tuning

To improve the model's accuracy, we applied the technique of fine-tuning, focusing specifically on adjusting the final layers of the model using our custom dataset. The aim was to assess how adding new data influences the model's performance. To do this, we fine-tuned five separate models, all initialized with the same pretrained weights, gradually increasing the number of subjects used for fine-tuning from one to five. The initial weights for these experiments were taken from the 70th epoch of training on 70% of the AnDy dataset. The fine-tuning involved the last three fully connected convolutional layers and the decoder responsible for the action classification. For consistency, we evaluated the performance of all models on the same two subjects as in previous experiments, allowing us to make direct comparisons and observe the impact of incorporating additional data.

5.3.1 Results and discussion

In the fine-tuning experiments, there was a significant improvement in accuracy, rising from 0.5 to 0.76 as the number of subjects used for fine-tuning increased from zero to five as shown in figure 5.12. AUC gained a remarkable improvement too, as reported in figure 5.13, starting from 0.55 without any fine-tuning and reaching 0.79 when the model is tuned on all five subjects. While this marked a notable enhancement in overall performance, certain action labels were sacrificed in the process. As reported in figure 5.14 the accuracy trend for individual actions is not consistently increasing. In several cases, such as "Idle," "Reaching," "Carry," and "Screwing," the final accuracy obtained is not the highest in this experiment set. A special case is the "Manipulate" action, which shows significantly lower accuracy compared to the others due to its complex nature. The dexterous movements involved in fixing chair parts were often misclassified as actions like reaching, placing, or even screwing, especially when the hands were positioned close together, leading to ambiguity in the recognition process. Without considering this outlier label the overall accuracy of the model with the maximum data used in the fine tuning goes up to 0.85. During the fine-tuning phase, it is interesting to observe that the model appears to deliberately lower its accuracy on specific actions in favor of improving overall performance. This behavior suggests that the model is prioritizing the recognition of more complex or generalizable features at the cost of precision in certain tasks. By doing so, it enhances its ability to perform well across a broader range of actions, reflecting a strategic adjustment aimed at optimizing global performance rather than focusing narrowly on individual actions. This indicates that the model prioritizes actions that contribute more significantly to general accuracy. This trade-off suggests a balancing mechanism within the model's optimization process, where improving general performance comes at the cost of certain specific actions. Another important result concerns the amount of data needed for effective training. We have demonstrated that fine-tuning is beneficial, but for a complex model like this, a large amount of data is essential to train it properly, even in the fine-tuning phase. The model is capable of distinguishing between very similar movements, which is one of its strengths. However, this also means that it can interpret the same action differently when it is performed with minor variations. To overcome this challenge, considering data-augmentation, which is generating new data from the collected one, adding noise or geometrical transformations, and collecting a larger dataset are both valid solutions. By providing the model with extensive examples, we teach it the full range of ways an action can be performed, helping it to generalize better and recognize the action consistently despite small differences in execution.

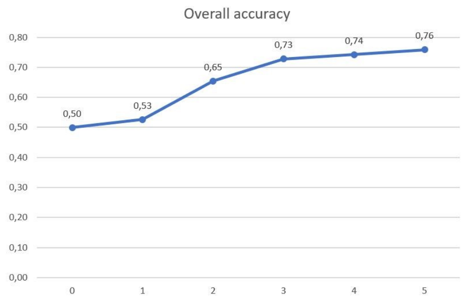


Figure 5.12: The evolution of overall accuracy over different fine-tuned models. In the y-axis it is shown the accuracy value, in the x-axis the number of subjects used during fine-tuning

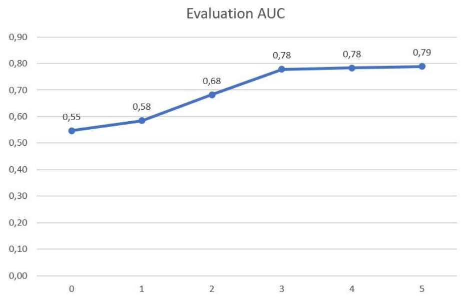


Figure 5.13: The evolution of AUC over different fine-tuned models. In the y-axis it is shown the AUC value, in the x-axis the number of subjects used during fine-tuning

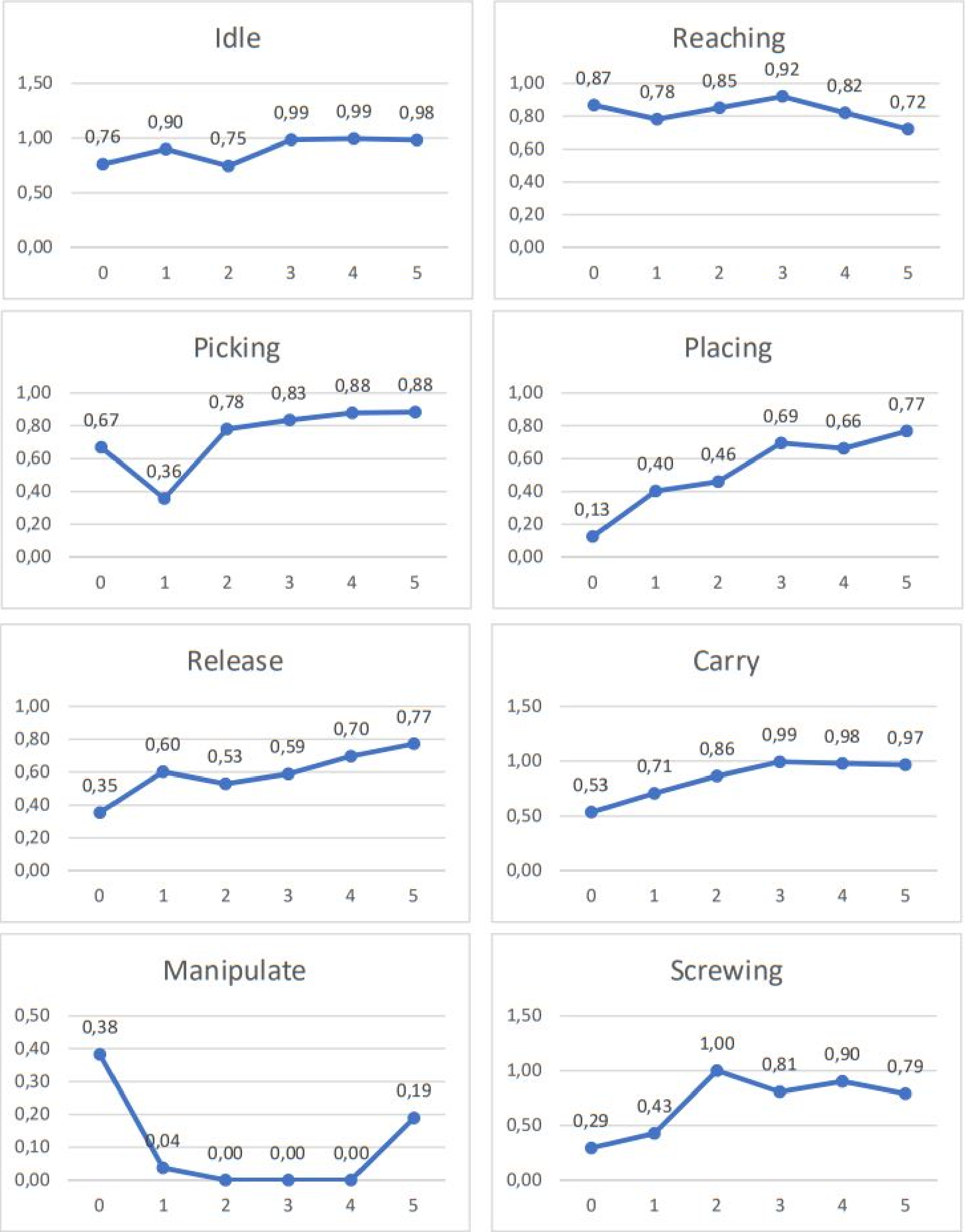


Figure 5.14: The evolution of accuracy of each label over different fine-tuned models. In the y-axis it is shown the accuracy value, in the x-axis the number of subjects used during fine-tuning

5.4 Data augmentation and dataset balancing

As previously discussed, the performance of action recognition models can be limited by two critical issues: insufficient data and an imbalanced dataset. In such cases, as previously discussed, certain labels may dominate the dataset, skewing the models learning process and and consequently leading to biased metrics during evaluation. To address these issues, we employed data augmentation to artificially increase the number of samples, ensuring a bigger quantity of data and at the same time a more balanced distribution across all categories.

The data augmentation process was carried out by first calculating the number of samples per label, which show that some labels are over or under represented, as reported in Table 5.2. A multiplier was then computed for each label based on its relative frequency, which was used to determine the probability of generating new samples for that label. New samples were created by applying a combination of a rotation around the z-axis and the addition of normal noise to the each skeleton data, simulating natural variations in movement. This augmentation process was repeated until the number of samples for each label was approximately equal: Table 5.3 reports the final number of samples for label.

Idle	Reaching	Picking	Placing	Release	Carry	Manipulate	Screwing
315	625	477	459	429	271	78	161

Table 5.2: Initial distribution of samples per label

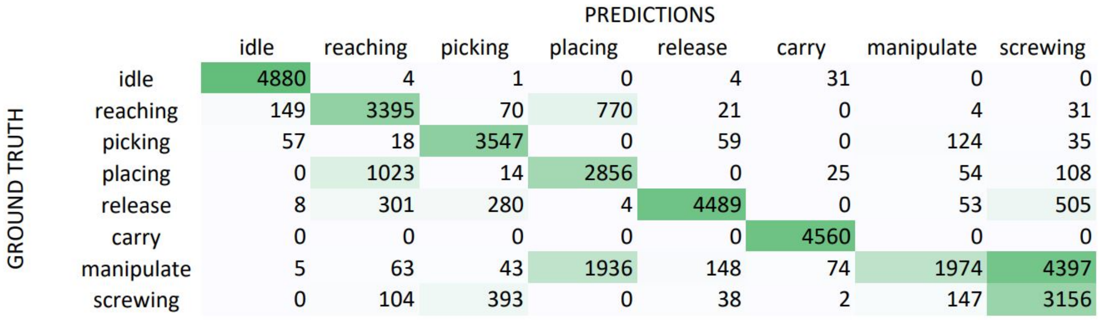
Idle	Reaching	Picking	Placing	Release	Carry	Manipulate	Screwing
702	840	774	766	748	691	624	664

Table 5.3: Final distribution of samples per label after data augmentation

Then, using this enlarged dataset we fine tune the model starting from the weights used also for the other fine tuning experiments, following the same procedure already outlined.

5.4.1 Results and discussion

The results obtained after fine-tuning the model using the augmented dataset demonstrate a substantial improvement in overall performance, with an accuracy of 0.78 . These metrics indicate another slight improvement compared to earlier fine-tuning experiments without data augmentation, where the model achieved an accuracy of 0.76. Despite this improvement, the accuracy across individual action labels reveals some variation, as shown in Table 5.4 and in Figure 5.15. Certain actions, such as "Idle" (0.99) and "Carry" (1), exhibit near-perfect recognition, suggesting that the model excels in identifying relatively static or clearly defined tasks. However, more actions, like "Manipulate," show significantly lower accuracy (0.23), similar to the previous fine-tuning experiments. This underperformance could be explained by the spreading of systematic problems, such as the nature of the movement, also in the augmented data.



PREDICTIONS

GROUND TRUTH	idle	reaching	picking	placing	release	carry	manipulate	screwing
idle	4880	4	1	0	4	31	0	0
reaching	149	3395	70	770	21	0	4	31
picking	57	18	3547	0	59	0	124	35
placing	0	1023	14	2856	0	25	54	108
release	8	301	280	4	4489	0	53	505
carry	0	0	0	0	0	4560	0	0
manipulate	5	63	43	1936	148	74	1974	4397
screwing	0	104	393	0	38	2	147	3156

Figure 5.15: Confusion Matrix of inference on subjects 1 and 2 of custom dataset, evaluated using model fine-tuned on augmented custom dataset

Idle	Reaching	Picking	Placing	Release	Carry	Manipulate	Screwing	Total
0.99	0.76	0.92	0.70	0.76	1	0.23	0.82	0.78

Table 5.4: Accuracy per label of model fine-tuned on augmented dataset

The augmented data appears to have improved recognition of most actions compared to earlier fine-tuning without augmentation, where the model struggled to consistently classify such movements. However, the augmentation did not fully resolve issues with the "Manipulate" label, where the model continued to struggle due to the ambiguous nature of hand movements during chair assembly. These results reflect and confirm the model’s tendency to sacrifice accuracy on more complex actions in favor of improving performance on easier or more generalizable ones. This observation is consistent with the behavior noted during the initial fine-tuning phase, where the model appears to prioritize actions that have a greater influence on overall accuracy, and shows that this behaviour is not fully driven by the balance of the dataset used.

Despite these challenges, the use of data augmentation has proven beneficial in improving the model's ability to generalize across a range of actions, particularly for those that were underrepresented in the original dataset. The augmented dataset allowed the model to better recognize actions but also remark the challenges of this industrial scenario and underline difficulties in learn ambiguous actions, suggesting that further refinements, possibly involving more targeted augmentation strategies or additional training data, are necessary to achieve better accuracy.

6 Conclusions and Future Works

6.1 Summary

This thesis has explored the application of online human action recognition in industrial scenarios. The research aimed to develop a robust system that can classify human actions based on skeleton data, with the goal of achieving an high accuracy in real-time action classification. To meet the real-time demands of action recognition in such a context, we adopted and adjusted the state-of-the-art *InfoGCN++* model. This graph convolutional neural network was chosen for its capability to effectively model complex skeleton data and capture spatio-temporal dependencies crucial for recognizing human actions, and for being able to handle classification without the whole action sequence. Given the industrial setting, the system needed to handle both repetitive, well-structured movements as well as slight variations in movement patterns that occur in real-world environments. The model was trained on the AnDy dataset, which contains sensor acquisition and labeling of sequences of industrial actions and was finally fine-tuned with a custom dataset collect in our laboratory to optimize performance for real-time action recognition. Experiments were conducted to assess the model's performance under various conditions. Results showed that *InfoGCN++* achieved high accuracy, with the model demonstrating strong learning capabilities and generalizing well to previously unseen subjects. Although the model performed well in most cases, action variations and differences in movements introduced inconsistencies in the recognition performance. This was a critical challenge in the industrial context, where workers often perform the same task differently based on individual habits, fatigue, or other factors. Initial inference on actions that featured such variations showed a noticeable drop in performance, which was not satisfactory for real-world application. To address this, we exploited fine-tuning technique. By refining the model using this method, the performance improved significantly. Fine-tuning allowed the model to adapt better to the nuances of movement variation, ultimately achieving a maximum accuracy of 0.85. This marked improvement highlighted the importance of model adaptability in real-time, real-world environments.

6.2 Limitations

One of the primary limitations of our project lies in inter-class similarity, which led to poor classification accuracy for certain labels, such as "manipulation." The actions within this class shared many overlapping characteristics, making it difficult for the model to clearly distinguish between subtle variations in movement and actually limiting the capability of the model to classify well all the actions. Our small dataset used for fine-tuning suffer also of unbalance: some actions for example *manipulation* has less training samples with respect to *reaching*.

Furthermore, our dataset included only a small set of actions, which limited the scope of the project. While the results were promising for the actions studied, expanding the action set could introduce more confusion for the model, as additional classes might share similarities or present new challenges, potentially leading to reduced accuracy.

Another critical limitation is the system's lack of context-awareness. The inertial sensors we used could only capture motion data, without considering any environmental factors. This significantly limited classification in scenarios like "pick and place", where differentiating between picking up different objects or performing similar tasks in varying contexts could be important. To overcome this, future work could integrate the system with a camera and incorporate computer vision algorithms to provide context-awareness, enabling more accurate and reliable action recognition across diverse scenarios.

6.3 Future developments

Future developments of this project could focus on several critical areas to enhance both performance and applicability. One area of improvement is hyperparameters tuning and optimizing training parameters such as the learning rate, batch size and weight decay. These adjustments could help refine the model's learning dynamics, enabling faster convergence and better generalization. In this project we trust results published by the model's authors without testing ourselves different configurations of the model.

Additionally, since fine-tuning improved performance in this project, further improvements could be achieved through fine-tuning on a larger dataset. A common technique to generate more data is data-augmentation. This process aims to generate new artificial data from the real skeleton, adding noise to joint positions and rotate and translate the skeleton sequence. The drawback of this technique is that if original data contains some systematic error, it will propagate to new data. So, could be necessary to collect a bigger dataset, with new sequences and more subjects. Another experiment at this point could be comparing performances of a model trained on a general-purpose dataset and then fine-tuned between a model directly trained on a dataset tailored to the specific industrial environment where to use this system. Expanding the action set to include more complex or nuanced movements is another option to explore, though this would require careful consideration to avoid inter-class confusion. The integration of a camera and a computer vision algorithm to recognize objects will help in differentiating similar actions that interact with different objects.

Furthermore, future work should aim to integrate the models ergonomics assessment capabilities directly into the action recognition framework. By embedding ergonomics evaluation, the system could not only classify human movements but also assess the ergonomic risks associated with each action in real-time. This could be achieved using the model to learn good and bad postures using labels inspired by some ergonomics assessment tools as EAWS, or to learn general body movements and assign them a risk factor, always computed with some ergonomics tool. Both approaches will provide information about healthy and risk-free actions and provide both insights into worker posture and an opportunity to intervene in unsafe or inefficient practices before injury occurs.

Ultimately, the broader vision of this project lies in advancing toward human-robot collaboration. In such systems, robots would work alongside human operators, dynamically adjusting to their actions while ensuring safety and optimal ergonomics. This means integrating our work in a more complex system, where our output become input for the algorithm responsible for the robot behaviour, to make it aware of what the human is doing and what his intentions are, and make it behave and react accordingly. To support this, future versions of the system could incorporate environmental context and object recognition, possibly through the integration of vision-based algorithms. This would enable the system to recognize not just human actions, but also the surrounding environment, allowing robots to anticipate and respond to both human behavior and external factors, creating a seamless, adaptive collaborative workspace.

6.4 Final conclusions

In conclusion, this thesis has demonstrated the potential of advanced action recognition models, specifically *InfoGCN++*, to be adapted for real-time applications in industrial settings. Through extensive experimentation and refinement, the system achieved high accuracy and generalization capabilities, particularly when dealing with repetitive, well-structured tasks. However, challenges remain in handling inter-class similarity and movement variation, as well as the lack of context-awareness, all of which are critical in complex, real-world environments. Despite these limitations, the integration of fine-tuning techniques improved performance, showcasing the model's adaptability. Looking forward, the incorporation of ergonomics assessments, expansion to larger datasets, and integration with vision systems offer promising avenues for enhancing the system's effectiveness. Ultimately, this work contributes to the ongoing effort to develop intelligent, context-aware systems for human-robot collaboration, with the potential to improve both productivity and worker safety in industrial environments. The results underscore the importance of continued research in this field, as the fusion of action recognition and ergonomics could play a crucial role in shaping the future of smart, collaborative workplaces.

References

[1] Vikas Pogadadanda et al. "Abnormal Activity Recognition on Surveillance: A Review". In: *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. 2023, pp. 1072–1077. DOI: 10.1109/ICAIS56108.2023.10073703.

[2] Venet Osmani, Sasitharan Balasubramaniam, and Dmitri Botvich. "Human activity recognition in pervasive health-care: Supporting efficient remote collaboration". In: *Journal of Network and Computer Applications* 31.4 (2008), pp. 628–655. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2007.11.002. URL: https://www.sciencedirect.com/science/article/pii/S1084804507000719.

[3] Bappaditya Debnath et al. "A review of computer vision-based approaches for physical rehabilitation and assessment". In: *Multimedia Systems* 28.1 (Feb. 2022), pp. 209–239. ISSN: 1432-1882. DOI: 10.1007/s00530-021-00815-4. URL: https://doi.org/10.1007/s00530-021-00815-4.

[4] Nicole Robinson et al. "Robotic Vision for Human-Robot Interaction and Collaboration: A Survey and Systematic Review". In: *J. Hum.-Robot Interact.* 12.1 (Feb. 2023). DOI: 10.1145/3570731. URL: https://doi.org/10.1145/3570731.

[5] Colin Dixon et al. "An operating system for the home". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. NSDI'12. San Jose, CA: USENIX Association, 2012, p. 25.

[6] Ronald Poppe. "A survey on vision-based human action recognition". In: *Image and Vision Computing* 28.6 (2010), pp. 976–990. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2009.11.014. URL: https://www.sciencedirect.com/science/article/pii/S0262885609002704.

[7] A.F. Bobick and J.W. Davis. “The recognition of human movement using temporal templates”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.3 (2001), pp. 257–267. DOI: 10.1109/34.910878.

[8] Ziming Zhang et al. “Motion Context: A New Representation for Human Action Recognition”. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 817–829. ISBN: 978-3-540-88693-8.

[9] Hong-Bo Zhang et al. “A Comprehensive Survey of Vision-Based Human Action Recognition Methods”. In: *Sensors* 19.5 (2019). ISSN: 1424-8220. DOI: 10.3390/s19051005. URL: https://www.mdpi.com/1424-8220/19/5/1005.

[10] Karen Simonyan and Andrew Zisserman. “Two-Stream Convolutional Networks for Action Recognition in Videos”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/00ec53c4682d36f5c4359f4ae7bd7ba1-Paper.pdf.

[11] Zehua Sun et al. “Human Action Recognition From Various Data Modalities: A Review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2023), pp. 3200–3225. DOI: 10.1109/TPAMI.2022.3183112.

[12] Chuankun Li et al. “DFN: A deep fusion network for flexible single and multi-modal action recognition”. In: *Expert Systems with Applications* 245 (2024), p. 123145. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2024.123145. URL: https://www.sciencedirect.com/science/article/pii/S0957417424000101.

[13] Pauline Maurice et al. “Human Movement and Ergonomics: an Industry-Oriented Dataset for Collaborative Robotics”. In: *The International Journal of Robotics Research* (2019).

[14] Md Golam Morshed et al. “Human Action Recognition: A Taxonomy-Based Survey, Updates, and Opportunities”. In: *Sensors* 23.4 (2023). ISSN: 1424-8220. DOI: 10.3390/s23042182. URL: https://www.mdpi.com/1424-8220/23/4/2182.

[15] Bin Ren et al. "A Survey on 3D Skeleton-Based Action Recognition Using Learning Method". In: *Cyborg and Bionic Systems* 5 (2024), p. 0100. DOI: 10.34133/cbsystems.0100. eprint: https://spj.science.org/doi/pdf/10.34133/cbsystems.0100. URL: https://spj.science.org/doi/abs/10.34133/cbsystems.0100.

[16] Guy Lev et al. "RNN Fisher Vectors for Action Recognition and Image Annotation". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 833–850. ISBN: 978-3-319-46466-4.

[17] Jun Liu et al. "Global Context-Aware Attention LSTM Networks for 3D Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[18] Jun Liu et al. "Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 816–833. ISBN: 978-3-319-46487-9.

[19] Seunghyeok Shin and Whoi-Yul Kim. "Skeleton-Based Dynamic Hand Gesture Recognition Using a Part-Based GRU-RNN for Gesture-Based Interface". In: *IEEE Access* 8 (2020), pp. 50236–50243. DOI: 10.1109/ACCESS.2020.2980128.

[20] Guilhem Cheron, Ivan Laptev, and Cordelia Schmid. "P-CNN: Pose-Based CNN Features for Action Recognition". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[21] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.12328. URL: https://ojs.aaai.org/index.php/AAAI/article/view/12328.

[22] Yong Du, Wei Wang, and Liang Wang. "Hierarchical recurrent neural network for skeleton based action recognition". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1110–1118. DOI: 10.1109/CVPR.2015.7298714.

[23] Amir Shahroudy et al. "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[24] Jun Liu et al. "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2020), pp. 2684–2701. DOI: 10.1109/TPAMI.2019.2916873.

[25] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[26] Farhood Negin et al. "A hybrid framework for online recognition of activities of daily living in real-world settings". In: *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2016, pp. 37–43. DOI: 10.1109/AVSS.2016.7738021.

[27] Xiang Wang et al. "OadTR: Online Action Detection with Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 7545–7555. DOI: 10.1109/ICCV48922.2021.00747.

[28] Lina Tong et al. "A Novel Deep Learning Bi-GRU-I Model for Real-Time Human Activity Recognition Using Inertial Sensors". In: *IEEE Sensors Journal* 22.6 (2022), pp. 6164–6174. DOI: 10.1109/JSEN.2022.3148431.

[29] Guoliang Liu et al. "Online human action recognition with spatial and temporal skeleton features using a distributed camera network". In: *International Journal of Intelligent Systems* 36.12 (2021), pp. 7389–7411. DOI: https://doi.org/10.1002/int.22591. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.22591. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22591.

[30] Marta Lorenzini et al. "Ergonomic human-robot collaboration in industry: A review". In: *Frontiers in Robotics and AI* 9 (2023). ISSN: 2296-9144. DOI: 10.3389/frobt.2022.813907. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022.813907.

[31] Filip Rybnikár et al. "Ergonomics Evaluation Using Motion Capture TechnologyLiterature Review". In: *Applied Sciences* 13.1 (2023). ISSN: 2076-3417. DOI: 10.3390/app13010162. URL: https://www.mdpi.com/2076-3417/13/1/162.

[32] Leandro Donisi et al. "Wearable Sensors and Artificial Intelligence for Physical Ergonomics: A Systematic Review of Literature". In: *Diagnostics* 12.12 (2022). ISSN: 2075-4418. DOI: 10.3390/diagnostics12123048. URL: https://www.mdpi.com/2075-4418/12/12/3048.

[33] B. Britzke K. Schaub G. Caragnano and R. Bruder. "The European Assembly Worksheet". In: *Theoretical Issues in Ergonomics Science* 14.6 (2013), pp. 616–639. DOI: 10.1080/1463922X.2012.678283. eprint: https://doi.org/10.1080/1463922X.2012.678283. URL: https://doi.org/10.1080/1463922X.2012.678283.

[34] Lynn McAtamney and Nigel Corlett. "Rapid upper limb assessment (RULA)". In: *Handbook of human factors and ergonomics methods*. CRC Press, 2004, pp. 86–96.

[35] Marta Lorenzini, Wansoo Kim, and Arash Ajoudani. "An Online Multi-Index Approach to Human Ergonomics Assessment in the Workplace". In: *IEEE Transactions on Human-Machine Systems* PP (Jan. 2022), pp. 1–12. DOI: 10.1109/THMS.2021.3133807.

[36] Adrien Malaisé et al. "Activity Recognition for Ergonomics Assessment of Industrial Tasks With Automatic Feature Selection". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1132–1139. DOI: 10.1109/LRA.2019.2894389.

[37] Monique Paulich et al. "Xsens MTw Awinda: Miniature Wireless Inertial-Magnetic Motion Tracker for Highly Accurate 3D Kinematic Applications". In: (May 2018). DOI: 10.13140/RG.2.2.23576.49929.

[38] Seunggeun Chi et al. *InfoGCN++: Learning Representation by Predicting the Future for Online Human Skeleton-based Action Recognition*. 2023. arXiv: 2310.10547 [cs.CV]. URL: https://arxiv.org/abs/2310.10547.

[39] Hyung-Gun Chi et al. "InfoGCN: Representation Learning for Human Skeleton-based Action Recognition". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 20154–20164. DOI: 10.1109/CVPR52688.2022.01955.

[40] Pengfei Zhang et al. "Semantics-Guided Neural Networks for Efficient Skeleton-Based Human Action Recognition". In: June 2020, pp. 1109–1118. DOI: 10.1109/CVPR42600.2020.00119.

[41] Chao Li et al. "Co-occurrence Feature Learning from Skeleton Data for Action Recognition and Detection with Hierarchical Aggregation". In: July 2018, pp. 786–792. DOI: 10.24963/ijcai.2018/109.

[42] Kun Su, Xiulong Liu, and Eli Shlizerman. "PREDICT & CLUSTER: Unsupervised Skeleton Based Action Recognition". In: 2019. arXiv: 1911.12409 [cs.CV]. URL: https://arxiv.org/abs/1911.12409.

[43] Jiang Wang et al. "Cross-View Action Modeling, Learning, and Recognition". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2649–2656. DOI: 10.1109/CVPR.2014.339.

[44] Oresti Banos et al. "Window Size Impact in Human Activity Recognition". In: *Sensors* 14.4 (2014), pp. 6474–6499. ISSN: 1424-8220. DOI: 10.3390/s140406474. URL: https://www.mdpi.com/1424-8220/14/4/6474.