

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Using Deep Learning and Computer Vision to enhance Human-Robot Collaboration**

**Duarte Alão Magalhães Silva Moutinho**



**International Master in Computer Vision**

**Supervisors:**

**Prof. Luís Teixeira (PhD)**

**Prof. Germano Veiga (PhD)**

**Eng. Luís Rocha (PhD)**

**April 1, 2022**





# **Using Deep Learning and Computer Vision to enhance Human-Robot Collaboration**

**Duarte Alão Magalhães Silva Moutinho**

International Master in Computer Vision

Approved in oral examination by the committee:

Chair: Prof. Jaime Cardoso

External Examiner: Prof. João Neves

Supervisor: Prof. Luís Teixeira

April 1, 2022



# Abstract

The trends of the modern industrial sector indicate a transition to more flexible production systems that are quickly adjustable to changing production requirements. Human-robot collaboration is one of the central components of this paradigm shift, which was brought forward by the technological breakthroughs of Industry 4.0 and is seen as critical to increase enterprise competitiveness, particularly in developed countries.

With these ideas in mind, the current dissertation aims to increase the natural collaboration level of a robotic engine assembly station by developing a cognitive system powered by computer vision and deep learning to interpret implicit communication cues of the operator. Such system should focus on obtaining assembly context through action recognition of the tasks performed by the operator. This will provide the robot with the ability to respond to the actions of the operator without the need of being prompted to do so, thus improving the collaboration level of the assembly station.

A dedicated dataset was proposed and recorded for the considered collaborative engine assembly task, allowing for deep learning models to be trained for the use case at hand. Due to limitations of availability of human resources, only one operator participated in the recording, compromising the ability of the trained models to work with any operator without prior adjustments. Nevertheless, solutions were proposed to surpass this issue, such as the recording of a more variable dataset or the implementation of transfer learning to fine-tune the proposed model. Regarding the cognitive system itself, several deep learning model architectures were proposed and tested to perform the intended human action recognition, including a 3D convolutional neural network, and two different variations of 2D convolutional neural networks and recurrent neural networks. The selected model architecture for the final system was a residual convolutional neural network with 34 layers and a long-short term memory residual neural network (*ResNet-34 + LSTM*).

This model showed a great performance, achieving an accuracy of 96.65% and a temporal mean intersection over union (mIoU) of 94.11%. Moreover, a task-oriented evaluation showed that the proposed cognitive system was able to leverage the performed human action recognition to command the adequate robot actions with near-perfect accuracy. As such, the proposed system was considered as successful at increasing the natural collaboration level of the considered assembly station.

**Keywords:** Computer vision, Deep learning, Human action recognition, Human-robot collaboration



# Resumo

O setor industrial moderno tem vindo a transitar para sistemas de produção mais flexíveis com capacidade de resposta a requisitos de produção em constante alteração. A colaboração entre humanos e robôs tem sido um dos principais componentes desta mudança de paradigma, que teve origem nas inovações tecnológicas da Indústria 4.0 e que é vista como crucial para melhorar a competitividade das empresas, particularmente em países desenvolvidos.

A presente dissertação tem como principal objetivo aumentar o nível de colaboração natural de uma estação colaborativa de montagem de um motor de combustão interna através do desenvolvimento de um sistema cognitivo com base em visão por computador e *deep learning* para interpretação de meios de comunicação implícita do operador. Tal sistema deverá ser capaz de obter contexto sobre o processo de montagem através do reconhecimento das ações do operador. Munido desta capacidade, o robô poderá responder às ações do operador de um modo eficiente, sem que seja necessário dar-lhe uma indicação de forma explícita.

Um *dataset* dedicado à aplicação em causa, montagem colaborativa do motor, foi proposto e gravado, permitindo treinar e avaliar modelos de *deep learning*. Devido a limitações de recursos humanos, apenas um operador participou na gravação do *dataset*, comprometendo a capacidade dos modelos em serem utilizados com outro operador sem qualquer ajuste. Não obstante, foram propostas soluções para ultrapassar esta limitação, desde a gravação de um *dataset* com maior variabilidade ou a utilização de *transfer learning* de modo a ajustar o modelo ao operador em causa. No que diz respeito ao sistema cognitivo desenvolvido, diversas arquiteturas de modelos de *deep learning* foram propostas e testadas para realizar o pretendido reconhecimento de ações humanas, incluindo uma rede neural convolucional 3D e duas redes neurais convolucionais 2D com *recurrent neural networks*. A arquitetura selecionada para o modelo final foi uma rede neuronal convolucional residual de 34 camadas e uma *long-short term memory residual neural network* (*ResNet-34 + LSTM*).

O modelo proposto revelou uma excelente performance, atingindo uma *accuracy* de 96.65% e uma *mean intersection over union* temporal (mIoU) de 94.11%. Adicionalmente, foi realizada uma avaliação da aplicação do modelo na estação colaborativa de montagem do motor, onde foi demonstrado que o sistema cognitivo proposto foi capaz de utilizar o reconhecimento de ações do operador para comandar as ações adequadas do robô, atingido uma precisão perto do ideal. Deste modo, o sistema proposto foi considerado um sucesso no que diz respeito ao aumento da capacidade de colaboração natural da célula de montagem em causa.

**Keywords:** Visão artificial, Deep learning, Reconhecimento de ações humanas, Colaboração homem-robô



# Acknowledgements

Writing this dissertation was a challenging yet fulfilling experience, filled with opportunities for professional and personal growth. Without the continuous support and guidance of the following people, this would not have been possible. For that, I owe them my sincere gratitude.

First and foremost, I would like to express my gratitude towards my supervisors - Prof. Luís Teixeira, Eng. Luís Rocha, and Prof. Germano Veiga. Not only was their technical contributions essential to achieve the proposed goals but also their personal guidance and motivation were of utmost importance.

I would also like to acknowledge the colleagues at INESC TEC with whom I had the pleasure to work alongside. A special word of gratitude goes to my colleague Carlos Costa not only for all his help with the collaborative assembly station but also for his priceless contributions and suggestions.

Finally, I could not end these acknowledgments without expressing my gratitude towards my friends and family whose continued support keeps me going. Thank you all.

Duarte Moutinho





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background and motivations . . . . .	1
1.2	Project exposition and goals . . . . .	2
1.3	Summary . . . . .	3
<b>2</b>	<b>State of the art</b>	<b>5</b>
2.1	Human-Robot Collaboration . . . . .	5
2.2	Human Action Recognition . . . . .	6
2.2.1	Resources for human skeleton data estimation . . . . .	8
2.2.2	Related work . . . . .	10
2.3	Eye Gaze . . . . .	11
2.3.1	Resources for human eye gaze estimation . . . . .	12
2.3.2	Related work . . . . .	14
2.4	Summary . . . . .	16
<b>3</b>	<b>Project Description</b>	<b>17</b>
3.1	Assembly scenario and hardware . . . . .	17
3.2	Previous work on the assembly station . . . . .	19
3.3	Project contributions . . . . .	22
<b>4</b>	<b>Collaborative plan proposal and integration of hardware and software</b>	<b>23</b>
4.1	Proposed collaborative assembly plan . . . . .	23
4.2	Hardware integration . . . . .	25
4.3	Skeleton tracking software . . . . .	27
<b>5</b>	<b>Collaborative Assembly Dataset</b>	<b>29</b>
5.1	Dataset Recording . . . . .	29
5.2	Dataset Labeling . . . . .	31
5.3	Dataloader implementation and optimization . . . . .	33
<b>6</b>	<b>Proposed Deep Learning Pipeline</b>	<b>35</b>
6.1	Implemented models . . . . .	35
6.1.1	Proposed architectures . . . . .	35
6.1.2	Model Variations . . . . .	37
6.2	Training and evaluation details . . . . .	39

<b>7</b>	<b>Results and Discussion</b>	<b>43</b>
7.1	Model validation . . . . .	43
7.2	Model testing . . . . .	47
7.3	Generalization assessment . . . . .	51
<b>8</b>	<b>Conclusions and Future Work</b>	<b>53</b>
<b>A</b>	<b>Camera support prototype</b>	<b>55</b>
<b>B</b>	<b>Video Labeling Application</b>	<b>57</b>
<b>C</b>	<b>Training and validation details</b>	<b>59</b>
C.1	ResNet 3D 18 . . . . .	59
C.2	ResNet-18 + LSTM . . . . .	62
C.3	ResNet-34 + LSTM . . . . .	65
	<b>References</b>	<b>69</b>

# List of Figures

2.1	Typical human action recognition framework. . . . .	7
2.2	Typical methodology for 3D skeleton estimation from images. . . . .	8
2.3	Bottom-up methodology employed by OpenPose . . . . .	9
2.4	Example of results obtained with <i>OpenFace 2.0</i> . . . . .	13
2.5	Examples of results obtained with <i>RT-GENE</i> . . . . .	14
3.1	Relevant components to the considered assembly operation. . . . .	18
3.2	Hardware components of the assembly station. . . . .	19
3.3	Example of the operation of the collaborative assembly station with augmented reality developed in the scope of the <i>Scalable 4.0</i> project. . . . .	20
3.4	Overview of the collaborative assembly station with an augmented reality system developed during the <i>Scalable 4.0</i> project. . . . .	21
4.1	Proposed collaborative assembly plan. . . . .	24
4.2	Robot's state machine. The states correspond to a Robot Task and the transitions are modeled by the detection of the corresponding Operator Tasks. . . . .	25
4.3	<i>Intel Realsense</i> camera mounted on the support prototype. . . . .	26
4.4	Example of tested camera configurations within the assembly station. . . . .	26
4.5	Example of the skeleton detection in the use case of the engine assembly task. . . . .	28
5.1	Types of transitions between tasks. . . . .	30
5.2	Overview of the recorded collaborative assembly dataset. . . . .	32
6.1	Residual learning block employed by residual networks (ResNets). . . . .	36
6.2	3D CNN model architecture. . . . .	36
6.3	2D CNN + RNN model architecture. . . . .	37
7.1	Confusion matrix of the final model (ResNet-34 + LSTM) on the test set. . . . .	48
7.2	Task-oriented evaluation of the final model (ResNet-34 + LSTM). . . . .	49
7.3	Task-oriented evaluation of the final model (ResNet-34 + LSTM) using a temporal consistency filter of 2.5 seconds. . . . .	50
7.4	Confusion matrices of the final model (ResNet-34 + LSTM) with data collected the other operators. . . . .	52



# List of Tables

3.1	Main manipulator specifications - <i>KUKA LBR iiwa 14R 820</i> . . . . .	18
3.2	Collaborative assembly plan developed in the <i>Scalable 4.0</i> project. . . . .	21
4.1	<i>Intel Realsense D435</i> specifications. . . . .	25
7.1	Validation of the <i>ResNet 3D 18</i> model variations. . . . .	44
7.2	Validation of the <i>ResNet-18 + LSTM</i> model variations. . . . .	45
7.3	Validation of the <i>ResNet-34 + LSTM</i> model variations. . . . .	45
7.4	Validation of the final models for each proposed architecture. . . . .	46
7.5	Performance of the final model (ResNet-34 + LSTM) on the test set. . . . .	48
7.6	Performance of the final model (ResNet-34 + LSTM) with data collected with other operators. . . . .	51



# Abbreviations

API	Application Programming Interface
CNN	Convolutional Neural Network
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mIoU	mean Intersection over Union
OT	Operator Tasks
ResNet	Residual Network
RNN	Recurrent Neural Network
ROS	Robot Operating System
RT	Robot Tasks
SDK	Software Development Kit





# Chapter 1

## Introduction

### 1.1 Background and motivations

Traditional industrial robotic applications mainly focus on replacing human workers in monotonous and repetitive operations, allowing them to focus on tasks with greater added value. This approach of full automation is often characterized by high operation speeds, great quality, and low production costs, at the expense of high initial costs and limited adaptability [39].

The technological breakthroughs of the Fourth Industrial Revolution, commonly referred to as Industry 4.0, combined with the growing market pressure for increased product variability and decreasing batch sizes, have brought a significant paradigm shift to the automation of industrial processes, favoring more flexible production systems that can be effortlessly reconfigured to constantly changing production requirements. Within this context, the concept of human-robot collaboration is crucial as it allows the increase of the automation level of a given process without compromising its flexibility. As such, the employment of human-robot collaboration in industrial scenarios allows enterprises to be more competitive in comparison to countries where cheap labor is abundant, provides better product quality, lowers requirements for quality control, increases production throughput, and reduces occupational injuries arising from non-ergonomic tasks [39].

Nevertheless, due to the significant impact that the industrial sector has on the European economy, such a revolution in industry is expected to have ripple effects on society. On an individual level, this is especially true for factory workers who may see their employment threatened by automation. In turn, from a collective perspective, a sharp increase in automation may subvert the traditional role of industry as an employer and engine of prosperity. To address these issues, the concept of Industry 5.0 has been emerging, not as a replacement for Industry 4.0 but, instead, as its natural evolution with a focus on making industries more resilient, sustainable, and human-centered [7]. To achieve these goals, human-robot collaboration is, once again, seen as an essential component that allows industries to evolve while conserving their societal role.

The employment of robotic systems in collaborative scenarios has seen a continuously increasing research effort over the past two decades [2], showing that it may assume a pivotal role in the future of robotics. These research endeavors were crucial for the establishment of structured

technical specifications for collaborative robots, as defined in ISO/TS 15066 [20], in which four different safe collaborative modes are detailed. However, only two of these modes are suited for continuous and active collaboration between robots and humans - ‘Speed and Separation Monitoring’ and ‘Power and Force Limiting’. The former allows any traditional robot to be employed in a collaborative setting by using external sensors to guarantee a safe distance between the robot and the humans. In turn, the latter ensures the safety of the humans by limiting the force output in such a way that any potential collision with a human will not result in injury. Dedicated robotic systems with specific design considerations (e.g. elimination of sharp corners and accessible electric current) and the ability to detect collisions are required to employ a Power and Force Limiting collaborative strategy [6]. Even though this may result in a more costly final product, it may be the only ‘fully collaborative solution’ for situations where close collaboration between the operator and robot is desired [39].

Moreover, the impact of the rise of human-robot collaboration has not been limited to industrial applications, having contributed to the proliferation of robots in previously non-viable environments such as homes, offices, and hospitals [6]. It is believed that human-robot collaboration can revolutionize modern society in numerous different fields such as medicine, schools, entertainment, space exploration, and military [11].

## 1.2 Project exposition and goals

The present dissertation took place at the Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), which is a private non-profit Portuguese research institution, whose main mission is to advance science and technology, enabling science-based innovation through the transfer of new knowledge and technologies to industry, services and public administration. Particularly, the project was developed in collaboration with INESC TEC’s Centre for Robotics in Industry and Intelligent Systems (CRIIS), whose focus is to develop advanced solutions in automation and industrial robotics.

With these ideologies in mind, the current dissertation focuses on the development of a cognitive system based on computer vision and deep learning approaches to perform human action recognition and, thus, extract context awareness of a real industrial engine assembly station. This assembly context can then be integrated into a collaborative assembly plan, allowing for a more natural and efficient collaboration that relieves the operator of the burden of commanding the actions of the robot. Moreover, a real industrial assembly scenario will be used for this purpose, allowing the developed cognitive system to be evaluated with a task-oriented approach, and conclusions to be drawn on its performance in a real industrial operation.

The main goals of the project are as follows:

- i. Research and integration of existing computer vision tools for visual recognition of the operator and its intentions.
- ii. Development of deep learning methodologies to perform human action recognition and, thus, extract information about the performed assembly task.
- iii. Development of a collaborative robotic strategy that takes advantage of the extracted data and culminates in an increase in assembly efficiency.

## 1.3 Summary

This dissertation is organized as follows. Chapter 2 showcases the performed literature review, giving a natural emphasis to the interpretation of implicit communication cues for human action recognition. In turn, Chapter 3 introduces the considered use case, highlighting the assembly task at hand, the hardware present in the assembly station, and the previous work performed on the cell. Chapter 4 presents the proposed collaborative assembly plan and both the hardware and software that were integrated into the station. Chapter 5 introduces the dedicated dataset that was developed for the considered use case, outlining the implemented labeling methodology and the performed optimization of the data loading procedure. Chapter 6 discloses the implemented model architectures and the carried out training and evaluation procedures. In Chapter 7 the obtained results of the proposed human action recognition system are presented and discussed. Finally, Chapter 8 outlines the main conclusions of the dissertation and enumerates the future work on the further development of the proposed system.



## Chapter 2

# State of the art

This chapter summarizes the performed literature review. Firstly, an overview of human-robot collaboration will be made, highlighting different forms of communication. This will be followed by a more in-depth review of the implicit communication types that were deemed relevant for the considered use case of the collaborative engine assembly station.

### 2.1 Human-Robot Collaboration

Human-robot interaction is a broad term that encompasses any action involving a human and a robot. In turn, human-robot collaboration is a specific type of human-robot interaction, in which the human and the robot work together and continuously share the same workspace to accomplish a common goal. To achieve efficient collaboration, all partners must be integrated into a common work plan with a shared objective. Moreover, collaborative members should be able to recognize the tasks performed by their partners and evaluate their intentions so that they can anticipate their actions and adapt accordingly [19]. As such, a good collaborative partner should be equipped with some level of cognitive abilities, namely environment perception, decision making, action planning, and communication [6].

In ideal human-robot collaboration, the human is in charge of selecting a goal and performing most high-level decisions, whereas the robot is tasked with assessing the behavior of its partner, understanding his communication cues, and deciding on the best way to assist the human to achieve his goal. The communication between the human and the robot, which takes a pivotal role in the collaboration [11], may be achieved explicitly or implicitly [10, 31]. The former encompasses communication types that deliberately convey complex information, such as speech, communicative gestures (e.g. hand signs and pointing), and haptic signals (e.g. force/torque inputs). In turn, the latter refers to more instinctive types of communication such as facial expressions, action recognition (e.g. recognition of manipulative gestures or object usage), and physiological signals (e.g. heart rate, brain activity, or muscle activity). Nevertheless, the boundary between implicit and explicit types of communication may not always be distinct, as it may be seen by the

use of speech, which is an explicit form of communication but may also convey emotion implicitly. Additionally, using head pose and eye gaze to hint at what object a partner is referring to may be accomplished explicitly whenever the human purposefully focuses on it, or implicitly when the human is focused on other tasks [6].

Generally, it is considered that implicit forms of communication are more suitable for human-robot collaboration tasks, as they do not require any active effort from the human. However, to achieve this level of communication, the robot must be capable of deriving the intention of the human from some sort of signal (e.g. video footage, or brain/muscle activity), which is a complex task that requires a high level of cognition [6].

Moreover, it is also important to consider potential communication from the robot to humans. This type of communication is crucial for a robot to inform the humans of its intentions, thus increasing their perceived safety (i.e. their perception of danger level) and comfort level during the interactions, which is essential to improve the acceptability of robots in the workspace [10].

For the current use case of a collaborative assembly station, intending to relieve the operator of the added task of explicitly providing the robot with adequate communication cues, it was considered that the communication between human and robot should be accomplished implicitly. Additionally, communication cues that have no direct impact on the operator were prioritized, meaning that implicit types of communication, such as brain or muscle activity, that require the operator to be equipped with dedicated sensors were also discarded. As such, the current project focuses on implicit types of communication that may be evaluated remotely through computer vision techniques, relieving the operator of any additional burden. Facial expressions, human action recognition, and eye gaze are examples of implicit communication cues that may be evaluated with an external camera and, thus, represent no burden to the operator. From these, eye gaze and, particularly, human action recognition were seen as the most relevant for the considered use case as they are task-focused. Thus, these two communication types and their relevance to human-robot collaboration will be discussed further in the following sections.

## 2.2 Human Action Recognition

Human action recognition is one of the most active topics of research in computer vision, with numerous applications such as video surveillance, robotics, and human-computer interaction. Typically, an action recognition problem can be divided into three main steps, as represented in Figure 2.1 - feature extraction (extraction of spatial and temporal features from raw frames), action representation (transformation of the extracted features into a feature vector), and classification (attribution of a class to the considered action). It is believed that the success of action recognition problems is highly reliant on the feature extraction process. Traditionally, human action recognition methods relied on hand-crafted feature extraction methods that struggled to extract significant high-level discriminative information from raw frames, mainly due to the high complexity of the problem, noise, high degree of variability, etc. The deep learning revolution that has been dominating nearly all major computer vision fields since 2012 has changed this paradigm, allowing

for complex feature extraction models to be learned directly from raw data. This major shift has allowed the development of models with unprecedented levels of performance [22].

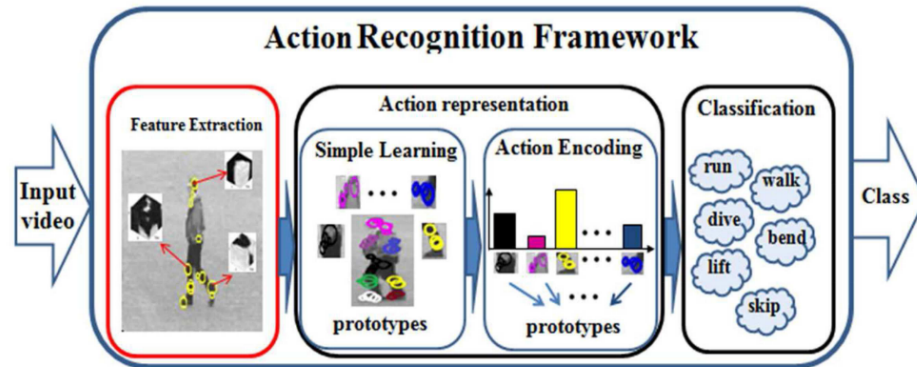


Figure 2.1: Typical human action recognition framework. Extracted from [22].

In computer vision, the most common approach using deep learning is a Convolutional Neural Network (CNN). However, since these models use 2D convolutions, they can only be used with images as input. This is a clear limitation for human action recognition, as the temporal information (evolution throughout the frames) is extremely relevant. As such, for a model to be as accurate as possible, it should be able to interpret not only the spatial but also the temporal information of a video feed. There are three main methods to perform human action classification directly from raw videos:

- i) **3D CNN:** Using this approach, the 2D convolutions are replaced by their 3D counterpart (convolution with a 3D kernel), allowing the feature extraction of a 3D CNN to convey information about several continuous frames. Even though this is one of the most typical approaches for video classification tasks, it has some shortcomings like its high computational cost and the need for a large amount of training data.
- ii) **Two-stream convolutional networks:** With this methodology, the model is divided into two distinct paths - a spatial stream that receives single frames as input, and a temporal stream that uses a multi-frame optical flow to extract temporal information. Both paths are then merged into a single class score to perform the classification of the video.
- iii) **Recurrent Neural Networks (RNN):** Using this approach, a 2D CNN is used as a feature extractor for each frame, and a recurrent neural network (e.g. a long short-term memory - LSTM) models the temporal information. Typically, these methods do not use the entire video due to high computational costs. Instead, a pre-defined window with a given frame length is used.

Moreover, instead of using raw video footage, different approaches may be employed to perform human action recognition. As an example, high-level skeleton data is often used to leverage human poses and identify the performed actions. The following subsections expand on these topics, enumerating resources to extract skeleton information from images and showcasing several

examples of applications of human action recognition, with a clear emphasis on human-robot collaboration scenarios.

### 2.2.1 Resources for human skeleton data estimation

As was previously highlighted, high-level 3D skeleton data is often used as a way to perform or enhance human action recognition. For this reason, the current subsection intends to elucidate how this data may be obtained using computer vision and deep learning methodologies.

The vast majority of existing depictions of humans are two-dimensional (images or videos) and, thus, contain depth ambiguity. Even though humans are great at understanding complex spatial arrangements in the presence of such ambiguities, this is a challenging task for computers [30]. As such, the task of obtaining 3D skeleton data is often divided into two sub-tasks: the prediction of 2D skeleton data and the conversion of said data into the three-dimensional space. Figure 2.2 displays an example of these sub-tasks being performed to obtain three-dimensional skeleton information.

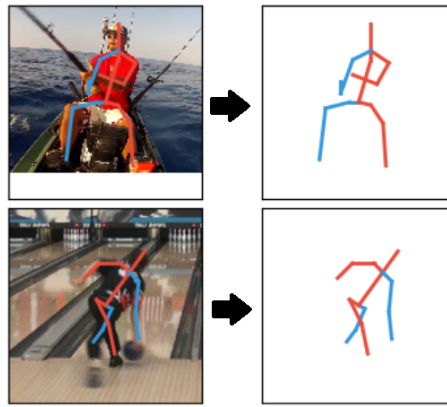


Figure 2.2: Typical methodology for 3D skeleton estimation from images. First the 2D skeleton is extracted from a given image. Subsequently, the 2D skeleton data is converted into the three-dimensional space. Extracted from [30].

Currently, 2D skeleton estimation tasks are performed almost exclusively using deep learning models, as their performance drastically surpassed the previously available methods. Nevertheless, the deep learning techniques may still be divided into two main approaches:

- i) **Top-down approach:** This methodology divides the keypoint detection problem into two sub-tasks. Firstly, it performs person detection using a dedicated detection CNN (e.g. a Region-based Convolutional Neural Network). Subsequently, it performs skeleton keypoint detection within the identified bounding boxes. This approach may not be competitive in real-time applications, as runtime complexity tends to grow with the number of people in the images. Moreover, the keypoint detection is reliant on the person detection model, which tends to fail in challenging scenarios (e.g. close interaction between two or more people).
- ii) **Bottom-up approach:** This methodology does the opposite of top-down methods. It starts by identifying potential keypoint regions and, subsequently, constructs the skeletons using



said keypoints. Recent work has underlined the potential of this approach, which is currently considered to be a more modern and efficient methodology. For these reasons, the current subsection will focus on bottom-up skeleton detection resources.

One of the most known and utilized resources for skeleton tracking is *OpenPose* [8, 9], an open-source library for real-time multi-person detection of keypoints of the human body, hands, face, and feet. *OpenPose* implements a bottom-up method to detect said skeleton keypoints by employing two different branches: one branch to predict confidence maps of body part locations, and another to predict Part Affinity Fields (PAFs), i.e. sets of 2D vectors that encode the degree of association between parts. The PAFs are then used to evaluate the relationship between the joints detected in the confidence map and perform a final decision on which joint belongs to each person. This approach allows for a greedy inference to efficiently estimate the keypoints of multiple people within an image. Figure 2.3 displays an example of the methodology employed by *OpenPose*.

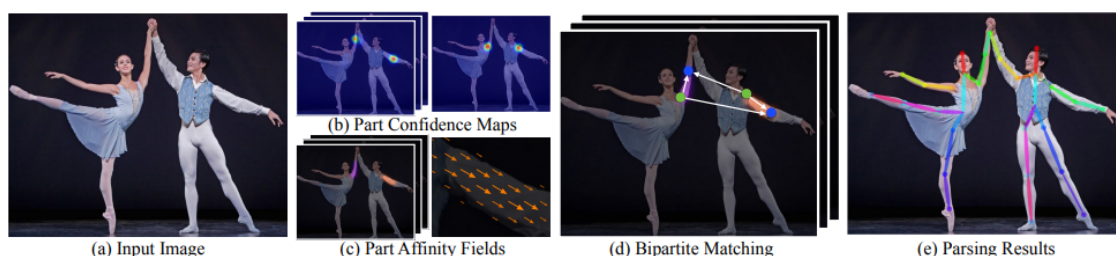


Figure 2.3: Bottom-up methodology employed by *OpenPose* [8]. Extracted from [9]

Another commonly used open-source resource for 2D skeleton tracking is *PifPaf* [23]. This library was developed to tackle the problem of challenging human pose estimation in low-resolution images, which is particularly relevant for autonomous navigation applications. Similarly to *OpenPose*, *OpenPifPaf* also implements a bottom-up approach divided into two main components: Part Intensity Fields (PIFs) and Part Association Fields (PAFs). The former, which intends to detect and locate body parts, has a composite structure composed of a confidence score, a vector map that points to the closest body part of a given component, and the size of the joint. In turn, PAFs are responsible for the association of the detected joints from multiple people into adequate poses. The authors claim that *PifPaf* has state-of-the-art performance for high-resolution images and outperforms existing methods for low-resolution images. This resource has been continuously developed, culminating in *OpenPifPaf* [24], a semantic keypoint detection that uses temporal information to enhance and accelerate the keypoint detection between frames.

In turn, the conversion from the 2D skeleton joints to the three-dimensional space may be achieved either by employing computer vision techniques or, once again, using deep learning models. As an example of computer vision techniques, the depth channel of an RGB-D camera may be used to solve the depth ambiguity between the camera plane and the object plane, culminating in three-dimensional coordinates of the skeleton data. The *Intel RealSense* or *Microsoft Kinect* are two of the most popular RGB-D cameras for these applications. Alternatively, it is

possible to achieve the same effect with the use of triangulation between two or more cameras. In fact, the most recent version of *OpenPose* [8] has a module to use camera triangulation to directly obtain 3D skeleton coordinates. These methodologies are typically not too complicated but have the disadvantage of only working in controlled scenarios with specific data acquisition hardware. As such, it cannot be applied to already existing data with uncontrolled acquisition settings.

Regarding the deep learning methods, there are several models that try to predict 3D skeleton data using only its 2D representation as input. The first widespread implementation of such a solution was *3D Pose Baseline* [30], in which a deep multilayer neural network with residual connections was developed for this task. Even though this approach is quite simple and lightweight, it was extremely effective at the task at hand. The authors considered that the majority of errors of the tested 3D pose estimation system arose from the 2D skeleton estimator and not the developed network. This work from Martinez et al. inspired numerous developments in this field, among which is *VideoPose3D* [33], an open-source project from *Meta Research* which uses a fully convolutional architecture with residual connections, and employs dilated convolutions to model long-term temporal dependencies. The authors concluded that *VideoPose3D* achieved great results, surpassing the previously best metrics on commonly used datasets.

### 2.2.2 Related work

In [40], Wang et al. employed a deep learning approach to perform action recognition and object classification in a collaborative assembly task. The goal of this model was to obtain context information about the assembly procedure that could be leveraged to increase the efficiency of the collaboration. The authors used a CNN to perform the classification directly on each raw frame. This is a simple and computationally inexpensive approach, yet it has the downside of not taking advantage of the available temporal information. Similarly, Liu et al. [28] employed a methodology to obtain context awareness and, consequently, increase the production efficiency of a collaborative disassembly station. The authors of this study used a CNN just for feature extraction, relying on a recurrent neural network (more specifically an LSTM) to explore the temporal information of the sequence of frames and identify the performed actions. Moreover, a comparison was performed between the implemented network, a simple CNN (no temporal information), and a 3D CNN, showing that the selected model significantly outperformed these two alternatives.

In [3], Amin et al. employed 2D and 3D CNNs to perform action recognition to evaluate dangerous interactions between a human and a robotic mobile platform. Before being fed to the network, the frames were enhanced with 2D skeleton data (skeleton representation overlaid on top of the image) to give an added emphasis to the human. Even though the authors reported great accuracy at detecting possibly dangerous situations for the simplified laboratory scenario (where the mobile platform is in a fixed location), it was highlighted that the 3D representation of both the human skeleton and the robotic arm should be added to the network input to ensure its viability in a real industrial scenario.

In [37], Song et al. proposed a deep LSTM network for action recognition based on high-level 3D skeleton data. The authors explore the concept of spatial and temporal attention modules, which allow the model to attribute additional importance to more relevant joints (e.g. the joints of the hand, elbow, and head are particularly important to identify a drinking action) and frames, as some frames convey more information than others. This methodology achieved state-of-the-art results at the time of publishing. In turn, Zhang et al. [41] proposed a deep learning method to forecast human's future motion trajectory, which may be used for online robot action planning and execution. This approach feeds 3D skeleton data to a recurrent neural network to learn the time-dependent mechanisms underlying human motions. To enhance the network's structure, the authors used the novel concept of functional units, which correspond to dedicated RNN layers for individual components (e.g. arms or spine) and for the coordination between them (e.g. coordination between arms and spine). The authors reported that the network achieved a great prediction accuracy, allowing its output to be used to enhance human-robot collaboration.

Finally, in [25], Lee et al. proposed a novel method to perform human action recognition in a collaborative robot scenario. The main idea behind this study was to create an efficient and computationally inexpensive model to perform human action recognition while taking advantage of temporal information. To achieve this goal, the authors used high-level 3D skeleton data from multiple frames and converted it to an RGB image, with each pixel corresponding to the coordinates of a single joint of the human and each row corresponding to data from a single frame. Thus, the final RGB image includes spatial and temporal data from the performed human action, allowing the classification to be made using a simple and efficient 2D CNN. The authors reported great accuracy and efficiency, allowing the model to be executed in real-time on embedded systems.

## 2.3 Eye Gaze

Eye gaze is believed to be a particularly important non-verbal signal, as evidence from psychology studies suggests that humans have unique 'hard-wired' pathways in the brain dedicated to its interpretation. The complex nature of interpreting and employing eye gaze data in human-robot collaboration tasks makes it a multidisciplinary problem spanning multiple fields such as robotics, artificial intelligence, and psychology [1].

In [1], Admoni et al. present a thorough review on the social impact of eye-gaze in several activities such as conversation and object manipulation. It includes considerations about how this type of communication may be used to enable a more natural human-robot collaboration. Moreover, the authors differentiate between various types of eye-gaze:

- **Mutual gaze**, which is often referred to as 'eye contact', is a gaze directed from one agent to the face or eyes of another. As the name indicates, reciprocity is a crucial component of mutual gaze.
- **Referential gaze** is a gaze directed at an object or location in space.

- **Joint attention** is characterized by the sharing of attention on a common object. It often involves several phases, starting with a mutual gaze to establish attention followed by a referential gaze to draw attention to a certain object.

The use of robotic eye-gaze in human-robot collaboration is reliant on the application and the type of interaction the robot wishes to have with the human. As an example, a tutoring robot mainly employs mutual gaze to express engagement with the user, whereas an assembly-line robot prioritizes task-focused gazes such as joint attention and referential gazes [1].

Due to the bilateral nature of the communication between two collaborative partners, the studies using eye-gaze to enhance human-robot collaboration tend to be divided into two main groups. On the one hand, multiple studies focus on the interpretation of human eye-gaze and how to use it to predict the human's intention with the ultimate goal of enhancing the robot's ability to collaborate. On the other hand, other studies aim to generate natural and 'human-like' robotic eye-gaze, with the intention of developing a robot that can intuitively communicate its intentions with humans. The following subsections will provide examples of both, putting a clear emphasis on the former due to the nature of the use case of the project at hand. Additionally, a review of the available resources for estimation of human eye gaze will be provided, as it is deemed essential to understand the related work.

### 2.3.1 Resources for human eye gaze estimation

There are multiple technological approaches to extract human eye gaze data. The first major division between these systems is whether they are active or passive. While the former approach uses infrared illumination to highlight the eyes, the latter uses only natural light, making them more natural, but more sensitive to changes in environmental conditions. Another possible distinction is between head-mounted and remote solutions. Head-mounted systems, which require the human to wear cameras either on a helmet or on glasses frames, can achieve high accuracy but may be impractical and not comfortable for extensive use. In turn, remote solutions can track the eye gaze from a distance, meaning that they require no effort from the human side. Finally, eye tracking solutions may also be differentiated on their need for a calibration procedure. Generally, calibrated systems can achieve higher accuracy, but require dreary calibration procedures and lose the ability to interact with people with no prior preparation [31].

With these considerations in mind, the solution that is more suitable for robotics is a passive remote system without the need for a calibration procedure, as such a system guarantees a high degree of naturalness and puts no burden on the human during the interaction. These considerations are even more crucial for situations where the robot is employed in a public space to interact with multiple people. For these reasons, this subsection will focus only on passive remote systems without the need for calibration.

Nevertheless, implementing such a natural and intuitive system is oftentimes difficult for several reasons. Firstly, such a system would benefit from high resolution and narrow field-of-view cameras, while robotic applications tend to prioritize wide field-of-view options to capture a

larger environment. Additionally, the high computation power required to process high-resolution footage may be limiting, particularly for situations where other real-time tasks are a priority (such as navigation, balancing, etc.) [31]. On the other hand, head pose, i.e. the direction in which the head is pointed, is believed to be a good approximation of eye gaze, while being considerably easier to compute. As such, in the field of human-robot interaction, head pose is often used instead of eye gaze. However, it is important to emphasize that even though good results have been obtained using head pose, eye gaze still holds the most relevant information, as both signals do not always coincide. This is especially true in situations where objects are close to each other, in which people tend to switch their focus just through eye movement. In [31], Palinko et al. performed a direct comparison between head pose and eye gaze in a human-robot collaboration scenario. The authors reported that even though both approaches performed similarly in social situations, the use of eye gaze lead to a significant efficiency increase in object selection tasks. Moreover, this approach was deemed more likable, smart, and effective by the participants in the study.

In recent years, there has been an emergence of passive, remote, and calibration-free solutions for eye gaze tracking, including several open-source projects. Amongst these, the most well known is *OpenFace* [4] and its sequel, *OpenFace 2.0* [5]. This project, developed by Baltrusaitis et al., aimed to create a state-of-the-art framework for facial behavior analysis, including facial landmark detection, head pose estimation, eye gaze estimation, and facial action unit recognition. *OpenPose* uses an instance of a Constrained Local Model to perform facial landmark detection by building a 3D representation of said landmarks, which is then used to estimate the head pose by solving a Perspective-n-Point problem. For the eye gaze estimation, no dedicated model is employed. Instead, it is obtained by projecting a ray between the center of the pupil (obtained in the facial landmark detection phase) and the camera origin. Figure 2.4 showcases examples of the results obtained with *OpenFace*.

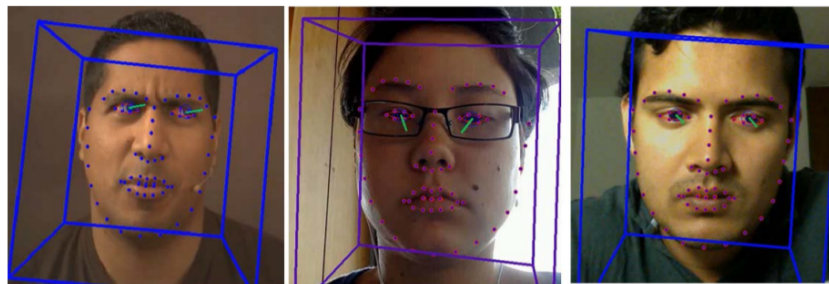


Figure 2.4: Example of results obtained with *OpenFace 2.0*. The blue prisms represent head pose, the green lines represent eye gaze, and the dots represent the facial landmarks. Extracted from [5].

Another open-source project of particular interest is *RT-GENE* [14], developed by Fischer et al., which intends to obtain robust eye gaze estimation in natural environments, i.e. large camera-subject distances and less constrained subject motion, making it particularly suited for human-robot collaboration applications. To achieve this, the authors created a dataset under these conditions and extracted the ground truth using a head-mounted system, which was then removed from the images using a GAN-based image generation model (semantic inpainting). A Multi-task Cascaded Convolution Network was used to directly extract the eye gaze data. Figure 2.5 showcases examples of *RT-GENE* being used to predict human eye gaze in challenging situations where the resolution is low.

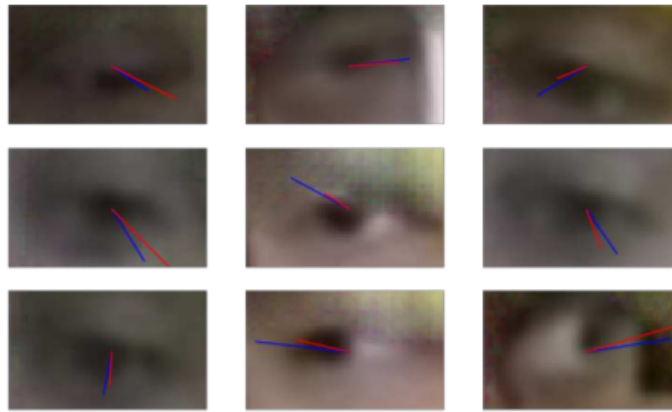


Figure 2.5: Examples of results obtained with *RT-GENE* showcasing its capability to work with low resolution images. Estimations are represented in red and ground truth in blue. Extracted from [14].

In turn, Lorenz et al. [29] proposed a tracking system based on two cascaded CNNs to remotely extract head pose and eye gaze of humans. The authors reported improved accuracy compared to *OpenPose*, especially in extreme conditions and dynamic environments. However, this project is not open-source, nor have the models been released for inference, thus, the proposed model cannot be easily integrated into other research efforts.

### 2.3.2 Related work

#### Robotic eye gaze

In [13], Duque-Domingo et al. proposed a novel method to control a robot's eye gaze during social situations with multiple humans, i.e. decide to whom the robot should pay attention when multiple people are competing for its attention. The outcome of this study was a competitive network that receives multiple stimuli (including human eye gaze) and decides who the robot should pay attention to and establish mutual gaze. As such, this research includes both main situations where eye-gaze may be used for enhancing human-robot interaction (interpretation of human eye gaze and generation of a natural robotic eye gaze).



### Human eye gaze

Eye gaze plays a crucial role in object reference and manipulation, as humans tend to look at objects in their environment before naming or manipulating them. When referring to objects, humans tend to fixate on them about a second beforehand [1]. In turn, when manipulating an object, the human's eye gaze typically meets the object of interest even before any movement of the hands is registered. Humans are great at intuitively picking up on these implicit cues and predicting the intention of their partner [31]. As such, it is expected that human eye gaze information can be used by robots to predict their partner's intention and enhance the collaboration.

Generally, most available studies that employ eye tracking in a collaborative robotics scenario rely on head-mounted systems, as they are easier to implement and tend to achieve higher accuracy. In [19], Huang et al. proposed an anticipatory control mode that monitors user actions, predicts their intentions, and proactively controls the robot actions in a collaborative robot station. The authors used eye gaze data (tracked by a head-mounted system) and speech recognition to evaluate the human's implicit and explicit communication cues. The intention prediction module uses a support vector machine and is based on the method presented in [18]. The authors compared the implemented anticipatory control mode with a reactive approach, having reported a decrease in the robot's response and grasping times, better teamwork, and a more positive user experience. In turn, Shi et al. [35] address a human-robot interaction problem using the human's eye gaze to select a certain object for the robot to pick up. The authors used eye-tracking glasses and an object detection CNN to explore the concept of visual saliency and determine which object the human intends to interact with. Finally, in [10], Chadalavada et al. propose a bi-directional communication plan between humans and an autonomous guided vehicle, where the robot interprets the human's eye gaze (obtained with eye-tracking glasses) and adapts its trajectory according to the human's intentions of passing through its left or right side. Additionally, the robot informs the human of this trajectory change by projecting it on the floor.

Alternatively, Saran et al. [34] proposed a novel approach to predict referential and mutual gazes to assess the attention focus of the human in a collaborative scenario. In this study, the authors used a less intrusive remote eye-tracking system, which has the added benefit of not impacting the human subject. The proposed algorithm employs several modules such as a face detector, eye gaze estimation, and object detection to compute mutual and referential gaze prediction scores, which may be used to infer social cues and action intent.

Even though the aforementioned related work focuses on the use of eye gaze as an implicit form of communication, several studies use it explicitly. This is extremely common in the medical field, particularly in robot-assisted surgeries, where the communication interface between the surgeon and the robot is often troublesome. This problem is particularly critical in situations where the surgeon's hands are occupied by tools (rendering the use of physical buttons impossible) or in long and strenuous procedures where the comfort level of the surgeon is critical. As such, the use of explicit eye gaze input from the surgeon is seen as a good solution to establish a communication tool with the robot. In [27], Li et al. presented an automated robotic laparoscope system that uses

a fuzzy logic interpretation of the surgeon's eye gaze data on the laparoscopic monitor to guide the tool. Similarly, Li et al. [26] developed a Dense CNN to extract eye gaze information that is used to control a surgical robot by specifying either its moving direction or its target position. Finally, in [15], Guo et al. introduced a novel robot system for needle-based percutaneous interventions, where the needle guidance is achieved through eye gaze data obtained from a CNN with a *ResNet-18* backbone.

## 2.4 Summary

In this section, an extensive literature review was made on the use of implicit communication cues in human-robot collaboration. Particularly, it focused on two types of communication - human action recognition and eye gaze - as they are task-focused and, thus, were considered as the implicit communication cues with greater potential for the use case of the collaborative engine assembly station. Each of these types of communication was introduced, enumerating related research studies and listing potential useful resources for their implementation. This literature review was essential for this dissertation as it defined the building blocks which will be used for the development of the present study.



## Chapter 3

# Project Description

The current chapter intends to present the considered use case. To do so, the engine assembly task will be showcased and the hardware present in the station will be enumerated. Additionally, the previous work performed on the considered station will be detailed with the intention of exposing not only its current state but also how it may be built upon. Finally, the contributions of this project to the considered station will be exposed.

### 3.1 Assembly scenario and hardware

The considered use case, which was derived from a real application scenario presented by the PSA Group, corresponds to a collaborative station for the assembly of an internal combustion engine. It is considered that the pistons and crankshaft were previously assembled on the engine block, while all other subsequent tasks should be performed in the assembly station. The assembly components, which are displayed on Figure 3.1, are the following:

1. Oil pump components
2. Crankshaft caps
3. Crankshaft cap screws
4. Windage baffle
5. Crankcase
6. Crankcase screws
7. Oil sump
8. Oil sump screws
9. Oil filter
10. Engine block

The station is also equipped with a collaborative robotic manipulator - *KUKA LBR iiwa 14R 820*. It is an industrial manipulator with 7 joints, each with a torque sensor, allowing it to achieve a compliant behavior. As such, this robot may be used alongside humans, safely sharing the same workspace without the need for physical barriers between the manipulator and operators. Moreover, *KUKA LBR* is extremely adequate for assembly tasks due to its traits of dexterity and kinematic redundancy [32]. Table 3.1 summarizes the main specifications of the used robot.



Figure 3.1: Relevant components to the considered assembly operation.

Table 3.1: Main manipulator specifications - *KUKA LBR iiwa 14R 820*

<b>Rated Payload</b>	14kg
<b>Weight</b>	29.91kg
<b>Reach</b>	820mm
<b>Positioning accuracy (ISO 9283)</b>	$\pm 0.1\text{mm}$
<b>Positioning repeatability (ISO 9283)</b>	$\pm 0.15\text{mm}$

The manipulator is equipped with two tools mounted on its end effector (the specifications and integration details of these tools will not be delineated as they fall outside the scope of the current dissertation):

- i) **Robotiq's 2F-85**, an adaptive gripper particularly suited for collaborative robotic applications, as it can control its exerted force. The gripper allows the robot to perform pick and place tasks of components.
- ii) **Kolver's Pluto 15CA/N**, an electric screwdriver with a DC motor and solid-state controls, which is adequate for high volume and high duty industrial applications. This tool allows the manipulator to tighten screws autonomously.

Figure 3.2 shows the considered assembly station, where it is possible to observe the fully assembled engine, the *KUKA LBR* manipulator, and its tools.

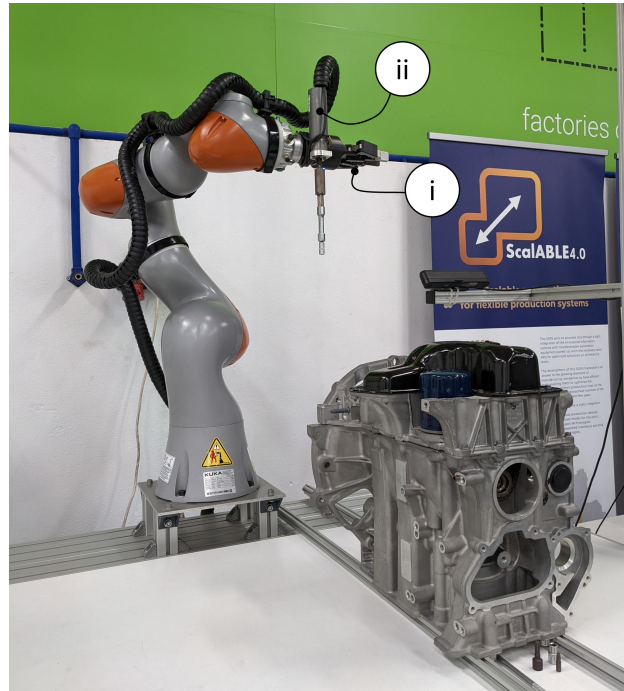


Figure 3.2: Hardware components of the assembly station - KUKA LBR manipulator, Robotiq's 2F-85 adaptive gripper (i), Kolver's Pluto 15CA/N electric screwdriver (ii).

### 3.2 Previous work on the assembly station

The presented assembly station was utilized by previous projects in INESC TEC's Centre for Robotics in Industry and Intelligent Systems. Particularly, it was developed in the *Scalable 4.0* project, which is part of the European Union's Horizon 2020 research and innovation program, and whose main objective was the development and demonstration of an open scalable production system framework (OSPS) that can be used efficiently and effectively to visualize, virtualize, construct, control, maintain, and optimize production lines.

The assembly scenario introduced in the previous section is composed of complex components that are difficult to manipulate and must be placed in hard-to-reach locations, requiring the dexterity, flexibility, and cognition that a human operator offers. Nevertheless, robotic systems may be used to increase the productivity of the assembly line by aiding the operator through the use of intuitive human-robot collaboration. Moreover, the traditional teaching methods that operators go through tend to not be very intuitive and require long training times. Thus, augmented reality systems that display concise three-dimensional information in real-time throughout the assembly process would be a good solution to expedite the training process of the operator and minimize possible operator mistakes. A portion of the aforementioned *Scalable 4.0* project focused on solving these two issues through the following contributions:

- i) Development of a **collaborative assembly plan** that combines the adaptability and dexterity of the operator with the repeatability and precision of robots, culminating in a flexible, productive, and cost-effective production line.

- ii) Development of an **immersive augmented reality system** capable of projecting three-dimensional information on the assembly station. The projected information is not only task-oriented, such as displaying relevant work zones, but it is also used to guide and train the operator by providing text and video instructions.

Figure 3.3 displays an example of the full system being employed, where the operator is assembling the oil pump and the robot is performing a pick and place operation of the crankshaft caps. The green and red regions, which correspond to the work zones of the operator and robot, respectively, provide information to the operator not only about which task to perform but also about where the robot motions are constrained, improving the perceived safety of the operator. Finally, on the bottom right corner, it is possible to see the projected interface, where the operator can receive instructions about the current assembly step and control the stages of the process. The projection mapping system for interactively teaching assembly operations employed in this project is outlined by Costa et al. in [12].

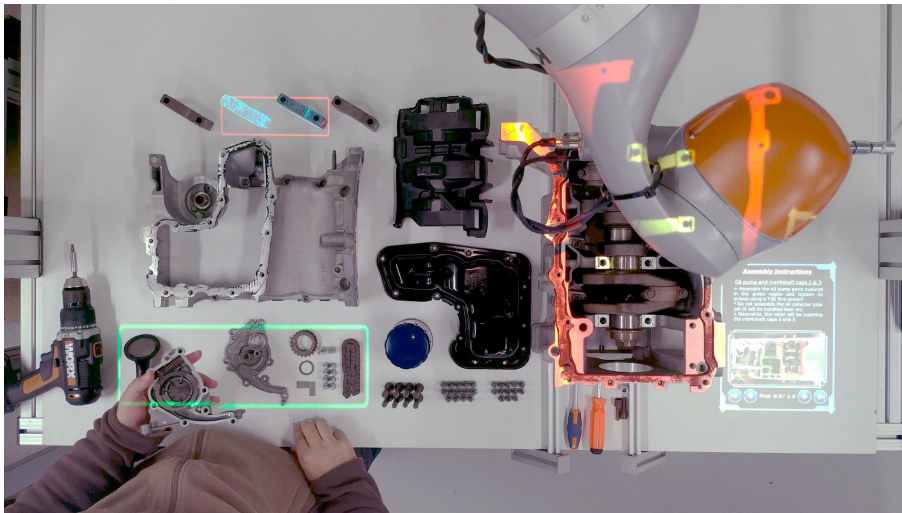


Figure 3.3: Example of the operation of the collaborative assembly station with augmented reality developed in the scope of the *Scalable 4.0* project.

In turn, Figure 3.4 shows an overview of the hardware components employed in the developed solution. The projector is in charge of projecting the information on the assembly station, whereas the 3D sensor is responsible for probing the environment around the projected interface to detect the commands provided by the operator. With this interface, the user may perform the synchronization between the operator and robot tasks.

Regarding the developed collaborative assembly plan, the required operations were divided into sequential assembly steps where the robot and operator are responsible for performing their respective tasks. Table 3.2 displays the defined division between robot and operator tasks. As it may be seen, the listed steps are composed of only one or two simple tasks. Nevertheless, they cannot be aggregated into more complex steps as the augmented reality system, which requires a different projection for each smaller task, is controlled by the assembly step.

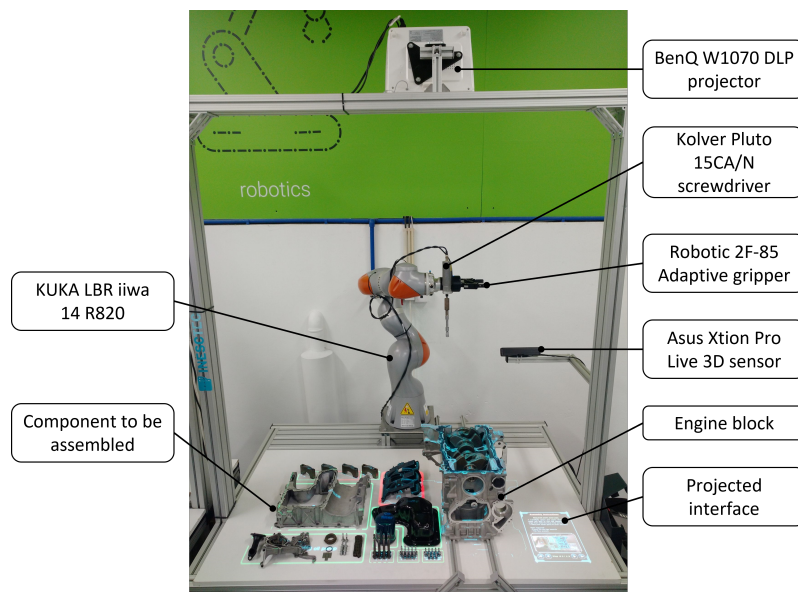


Figure 3.4: Overview of the collaborative assembly station with an augmented reality system developed during the *Scalable 4.0* project.

Table 3.2: Collaborative assembly plan developed in the *Scalable 4.0* project.

	Operator Tasks	Robot Tasks
Step 1	Preparation of the assembly station (placing of the parts on predetermined locations).	-
Step 2	Assembly of the oil pump outside of the engine block.	Insertion (pick and place) of crankshaft cap 2 and 3.
Step 3	Insertion of crankshaft cap 1 and 4 (this step cannot be performed by the robot due to spatial constraints in the location of the caps).	-
Step 4	Place screws on crankshaft caps. Attach 10mm head to the robot screwdriver.	-
Step 5	Tighten 4 crankshaft screws.	Tighten the remaining 4 crankshaft screws.
Step 6	Install the oil pump into the cylinder block. Assemble the oil collector tube.	Pick and place the windage baffle.
Step 7	Pick up the crankcase and place it on top of the cylinder block.	-
Step 8	Place the 14 crankcase screws. Attach 8mm hex head to robot screwdriver.	-
Step 9	Tighten 6 crankcase screws.	Tighten the remaining 8 crankcase screws.
Step 10	Pick the oil sump and place it on top of the crankcase.	-
Step 11	Place the 13 oil sump screws.	-
Step 12	Tighten 6 oil sump screws.	Tighten remaining 7 oil sump screws.
Step 13	Place the oil filter in its correct region and tighten it.	-



### 3.3 Project contributions

With the considered assembly station in mind, the current dissertation aims to build on top of the contributions of the *Scalable 4.0* project by introducing computer vision and deep learning techniques to enhance the collaboration between the operator and the robot. To reiterate, the main contribution of the project is the development of an artificial intelligence pipeline to increase the cognitive capabilities of the robot, allowing it to interpret implicit communication cues provided by the operator. These communication cues should then be integrated into a plan to enhance the current collaboration between human and robot. To achieve this, the implicit communication cues outlined in Chapter 2 were considered and evaluated, with the intention of being integrated into the contemplated use case.

The main identified potential improvement of the considered assembly station was the transitions between assembly steps. Currently, the assembly station transitions to the next assembly step only when prompted to do so by the operator through the projected interface. This may be considered a drawback as it requires the operator to manually command the required transitions. Thus, it would be beneficial for the robot to be able to monitor the operator, recognize its actions, infer the current assembly stage, and react accordingly. This may be achieved using human action recognition, as described in Section 2.2, to interpret the implicit actions of the operator. Thus, the development of a deep learning system to perform human action recognition in the context of the assembly station will be the main contribution of the current dissertation.

Nevertheless, even though the utilization of eye gaze, as seen in Section 2.3, is not as relevant as human action recognition for the current use case, it still holds valuable information that may lead to interesting applications for this station. Hence, even though the utilization of this communication cue is not seen as a priority to this dissertation, it will not be completely disregarded. Instead, it was seen as a secondary objective to further enhance the collaboration.

## Chapter 4

# Collaborative plan proposal and integration of hardware and software

The current chapter details all the developments that were made before starting with the data acquisition and model development. Firstly, a collaborative assembly plan will be proposed, using detectable operator actions as triggers for the robot tasks. This will be followed by the hardware integration of an RGB-D camera on the assembly station, which will be used to collect the required data. Finally, the used software to collect high-level skeleton data of the operator will be exposed.

### 4.1 Proposed collaborative assembly plan

As it was presented in Section 3.2, the previous work developed at INESC TEC employed a simple collaborative assembly plan, in which the planned tasks were divided into steps and attributed to either the robot or the operator. However, to employ the planned cognitive system to automate the transitions between assembly steps, it was necessary to adapt the assembly plan, ensuring that certain human actions serve as triggers for the robot to react accordingly.

To do so, a new assembly plan was developed, conserving the task division between the robot and operator, but using certain human actions to trigger the robot to start its tasks. For the development of the new assembly plan, it was assumed that the assembly station was previously prepared (placement of the parts in predetermined locations) and the robot's screwdriver starts with a 10 mm head attached.

Figure 4.1 displays a scheme of the proposed collaborative assembly plan, divided by Operator Tasks (OT) and Robot Tasks (RT). The defined operator tasks are purely sequential, i.e. the next task may start when the previous one finishes. However, the established Robot Tasks are sequential but also depend on a trigger provided by a given Operator Task. As an example, *RT2* can only start when *RT1* is finished and *OT3* has started. Additionally, even though the order of most operations is not interchangeable, *OT 5* ( '*Attach 8mm head to the robot's screwdriver*' ) in particular has some freedom. The only requirement is that it is performed after *RT2* ( '*Tighten the*

remaining 4 crankshaft screws’) and before RT4 ( ‘Tighten the remaining 8 crankcase screws’), as the crankshaft cap and crankcase screws have different screw head sizes.

Moreover, it is important to underline how some operator sub-tasks were aggregated into larger Operator Tasks. This was done only for related sub-tasks with a common denominator. As an example, the sub-tasks ‘Place the crankcase on top of the engine block’ and ‘Place the 14 crankcase screws’ were combined into the OT6, as they both pertain to the placement of the crankcase components, preparing them for the screwing operation. Nonetheless, it is important to underline that this aggregation into larger tasks will hinder the developed cognitive system’s capability to differentiate between the sub-tasks of a given Operator Task. This may seem like a problem as the augmented reality system projects different information on the assembly station depending on the performed sub-task. However, since the projections represent an external artifact that may be used by the visual-based deep learning model to differentiate between assembly tasks, it was decided that they should be deactivated, ensuring that the model only learns the assembly context based on the operator’s actions and used assembly components. Thus, the aggregation of sub-tasks into larger Operator Tasks has no downside to the developed system, while simplifying the proposed collaborative assembly plan.

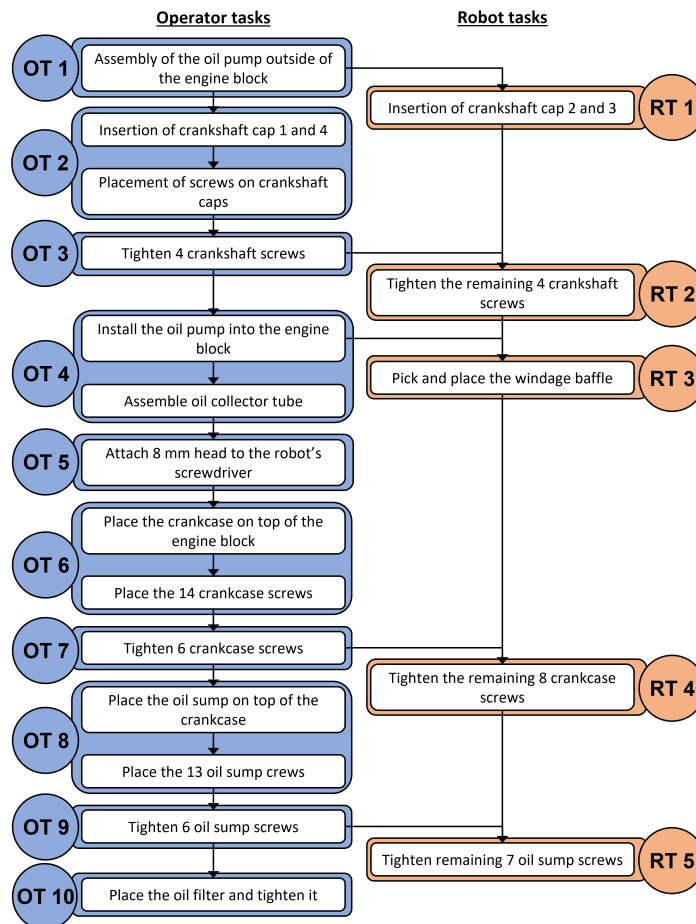


Figure 4.1: Proposed collaborative assembly plan.



From the perspective of the robot, it is possible to use this assembly plan to establish a state machine to control the robot tasks. Figure 4.2 displays the proposed state machine, where the states correspond to a given Robot Task and the transitions are modeled by the detection of the corresponding Operator Tasks. An additional constraint should also be implemented to ensure that the robot state cannot transition before its Robot Task has finished. With this state machine, the Robot Tasks may begin as soon as the corresponding Operator Task trigger starts being performed, thus improving the efficiency of the collaboration.

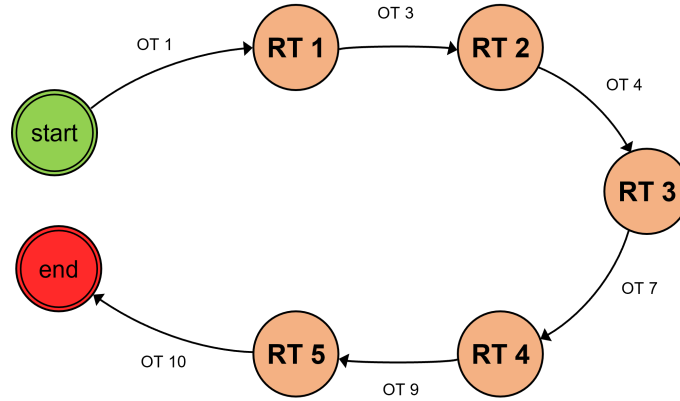


Figure 4.2: Robot's state machine. The states correspond to a Robot Task and the transitions are modeled by the detection of the corresponding Operator Tasks.

## 4.2 Hardware integration

In addition to the hardware components of the assembly station presented in Chapter 3, a sensor was selected to provide the cell with the ability to monitor its environment. The choice fell on *Intel's Realsense D435*, an RGB-D camera with widespread use in robotics applications. This camera allows not only the capture of regular RGB images that may be used for visual recognition but also depth data which may be used to solve the depth ambiguity problem that arises from the use of a single camera, allowing the capture of three-dimensional data. Table 4.1 displays the main specifications of the selected sensor.

Table 4.1: *Intel Realsense D435* specifications.

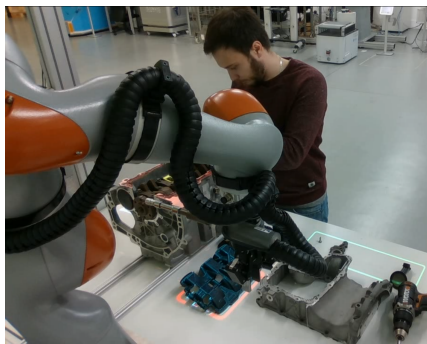
<b>Depth Technology</b>	Stereoscopic
<b>Depth FOV (H × V)</b>	87° × 58°
<b>Depth Resolution</b>	Up to 1280 × 720
<b>Depth Frame Rate</b>	Up to 90 fps
<b>Image Sensor Technology</b>	Global Shutter
<b>RGB Sensor Resolution</b>	2 MP
<b>RGB Sensor FOV (H × V)</b>	69° × 42°
<b>RGB Frame Rate and Resolution</b>	1920 × 1080 at 30 fps
<b>Ideal Range</b>	0.3 m to 3 m

Since the main goal of the project is to perform human action recognition to extract context on the performed assembly operation, the camera should be mounted in a location where it captures all operator movements in its field of view. Additionally, it is also beneficial that the assembly components are also captured by the camera, so that it may extract information from the interaction between the operator and said components. However, this task is somewhat challenging due to limitations of the cell structure and possible occlusions, particularly when the robotic manipulator moves. For this reason, and since it was important to test multiple camera configurations, it was beneficial to have a camera support that allowed the quick and effortless adjustment of the camera's position. Thus, an adjustable support prototype for the *Intel Realsense D435* was modeled and 3D printed. In Figure 4.3, the RGB-D camera may be seen mounted on the designed support prototype. Further details on the design of the camera support can be found in Appendix A.

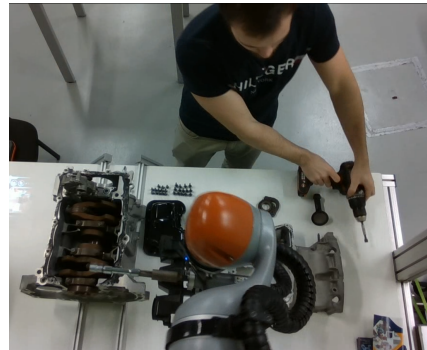


Figure 4.3: *Intel Realsense* camera mounted on the support prototype.

Figure 4.4 shows the two most promising camera positions that were found for the camera. *Perspective A* was considered ideal for most situations. However, for some robot configurations (such as the one shown in Figure 4.4a), the robot occludes the actions of the operator, rendering it unviable. Thus, *Perspective B* was the selected position for the camera as it minimizes the occlusions of the robot (the robot may occlude some components on the assembly table but it does not occlude the actions of the operator). The position of the camera in relation to the station may be consulted in Appendix A.



(a) Perspective A



(b) Perspective B

Figure 4.4: Example of tested camera configurations within the assembly station. *Perspective B* was the selected camera position as it minimizes occlusions to the operator actions.

### 4.3 Skeleton tracking software

Oftentimes, high-level skeleton data is leveraged to perform human action recognition. Intending to explore this possibility, the technological solutions available to obtain said skeleton data were evaluated to find the best fit for the considered assembly station. As described in Subsection 2.2.1, the two main methods to extract three-dimensional skeleton information from images both start by performing 2D skeleton keypoint identification. The differentiating characteristic is in the conversion from 2D to 3D coordinates, which may be achieved either using computer vision methods or using a deep learning approach. For the considered use case, it was decided that the former approach was more beneficial, as it requires fewer resources (particularly GPU usage and memory) which may then be employed in other tasks such as the intended action recognition. For these reasons, the employment of an RGB-D camera was seen as a natural solution for the station at hand (thus the selection of the *Intel Realsense D435*).

Regarding the used software, *realsense* has a dedicated skeleton tracking Software Development Kit (SDK), developed by *Cubemos*, a German software company. This SDK uses an efficient and proprietary model to extract the 2D skeleton data and converts it to 3D using the depth data from the RGB-D camera. With this approach, the sought-after skeleton data may be obtained in real-time without the use of a GPU (at a framerate of  $\approx 10$  fps with the available hardware).

The employed skeleton tracker detects a total of 18 keypoints (nose, neck, eyes, ears, shoulders, elbows, wrists, waist, knees, and ankles). Figure 4.5 displays an example of the performed skeleton detection in the assembly station. As it may be seen, the body parts not present in the field of view of the camera (e.g. knees, ankles, and right ear) are automatically discarded by the skeleton tracking SDK. This is accomplished by setting a threshold on the confidence score of the output of the skeleton detection network, rendering the joints that do not surpass the defined threshold as undetected. Moreover, the used camera position makes the skeleton detection task challenging due to the angle of the camera (above-view of the human). This leads to a decrease in performance of the skeleton detection when compared to a regular front-facing angle such as the one shown in Figure 4.4a. Nevertheless, the used system is robust enough to get consistent results even with the challenging angle, only resulting in a few undetected joints in occasional frames.

Additionally, even though the used SDK can detect the skeleton keypoints of multiple humans simultaneously, it cannot track them, as it does not have any way of evaluating the obtained skeletons throughout the frames. This was seen as a limitation of the system as some person other than the operator may enter the field of view of the camera, influencing the obtained skeleton data. For these reasons, a simple tracking algorithm was developed to isolate the skeleton data of the operator. This was achieved by calculating the centroid of the torso keypoints of the detected skeletons. The obtained centroids are then used to compute a distance score between them and the obtained skeleton in the last five frames. The skeleton with the lesser distance is considered to be the tracked human, in this case, the operator. This simple tracking algorithm achieved great results at isolating the skeleton data of the operator for situations where the humans do not drastically change positions in between two consecutive frames. Nevertheless, this condition was assumed

to be always true since, when using a framerate of  $\approx 10$  fps, two consecutive frames have a time difference of  $\approx 0.1$  s, making it impossible for large movements to occur in that time interval.

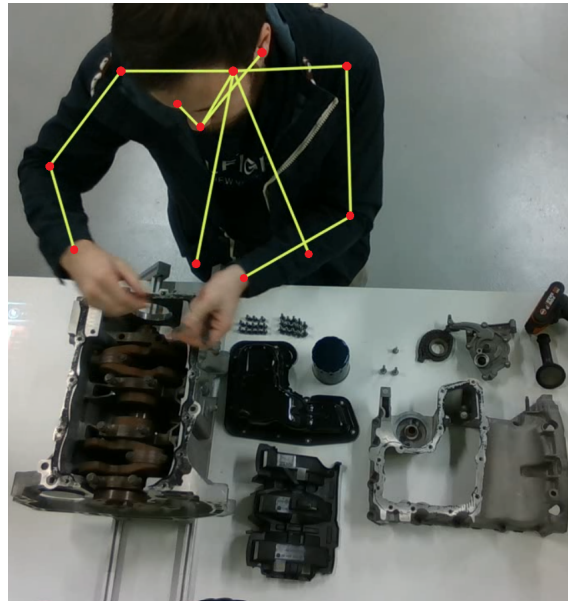


Figure 4.5: Example of the skeleton detection in the use case of the engine assembly task.

## Chapter 5

# Collaborative Assembly Dataset

This chapter introduces the dedicated dataset created for the collaborative engine assembly task. Firstly, the dataset will be presented, highlighting not only its main characteristics but also the details of the recording process. The dataset labeling procedure will also be explained, exposing the application developed for that purpose. Finally, the implementation and optimization of the data loading pipeline will be explained.

### 5.1 Dataset Recording

To develop a deep learning model based on supervised learning, it is necessary to have a labeled dataset so that the algorithm can be trained to accurately classify data (classification) or predict outcomes (regression). For general applications, large public datasets tend to be available for research and development purposes. As an example, the *Kinetics* dataset, whose latest iteration is the *Kinetics-700-2020* [36], is a human action dataset with 700 different human actions (each with at least 700 videos extracted from *Youtube*). However, for specific applications, such as the recognition of human actions in the context of a particular engine assembly task, a custom dataset needs to be recorded and appropriately labeled, which is often a complex and time-consuming task. Moreover, every detail that may influence the data must be considered and included in the dataset to ensure that the developed model is trained with data that is representative of the domain where it will be deployed.

With these ideas in mind, a dataset was recorded for the use case of the collaborative engine assembly plan showcased in Figure 4.1. The augmented reality collaborative assembly system described in Section 3.2 was used to record the dataset, with a few adjustments to ensure that the collected data is representative of the intended final use case.

The first major modification was the removal of the augmented reality system, as it projects information on the station that is indicative of the current stage of the assembly and, thus, may be considered as an external artifact with a direct impact on the recorded data. The removal of the projection should allow the network to focus only on the relevant operations, such as human actions and interactions with assembly components.

Another crucial characteristic of the dataset is how the transitions between assembly steps are made. In its current state, the transitions between the assembly steps are performed manually. However, since the goal is to automate the transitions between the assembly steps, the model is intended to be used with automatic transitions. This paradigm shift is crucial as it changes the timing when the robot starts its motion. As an example, Figure 5.1 shows the timeline of events in the transition between the Operator Tasks *OT2* and *OT3*, which triggers the Robot Task *RT2*. Using manual transitions, the natural behavior of the operator is that the transition trigger is commanded after finishing *OT2* and before starting *OT3*. This is the most efficient behavior with manual transitions as it maximizes the collaboration time between robot and operator. However, when working with automatic transitions, the operator should start *OT3* immediately after finishing *OT2*. As soon as the system detects that the transition was made, the robot starts its task (*RT2*). This has a major impact on the dataset since, if the regular manual trigger operation mode was used to record the data, the model could get focused on recognizing the movement of the robot instead of the human. In turn, this would cause the model to potentially have a good performance on the recorded validation and test sets, but perform poorly on the actual final application. This issue may be solved by simulating the behavior of an automatic transition operation mode. To achieve such behavior, the operator should start *OT3* immediately after finishing *OT2*, but trigger the manual transition in the middle of the task. Using this methodology, the recorded dataset will be more complete and representative of the actual task in which it is intended to be deployed, as its classes will have examples of both possible situations - operator performing the action independently, and operator and robot performing the action simultaneously. Moreover, since the operator will, most likely, finish the task earlier, there will be situations where only the robot is performing its tasks. These situations can be leveraged to teach the model that the actions of the robot are irrelevant to the classification, through the use of an adequate label (e.g. ‘no relevant operator action’).

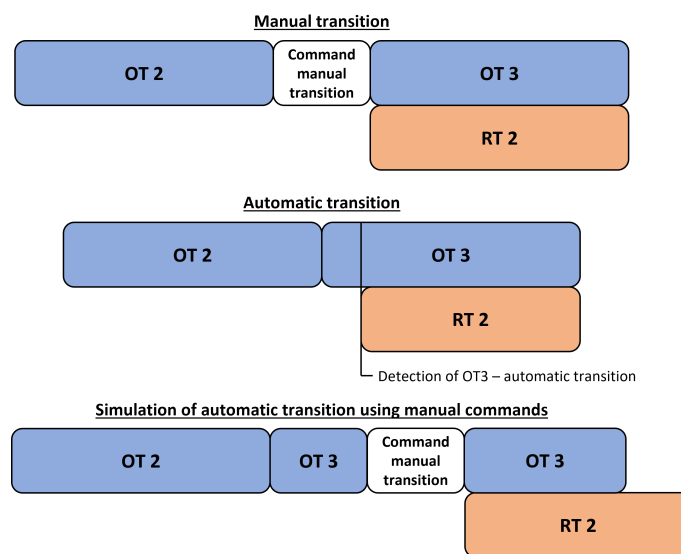


Figure 5.1: Types of transitions between tasks.

Yet another important attribute that should be accounted for when recording the dataset is its variability, i.e. how different its samples are from each other. Ideally, the dataset should include as much variability as possible to ensure that every situation that may arise during inference time is somewhat represented in the training dataset. However, for real applications, this is often challenging, as capturing a large and variable dataset may be a strenuous and time-consuming task with high associated costs. As an example, for the case of the collaborative assembly, it would be recommended that the captured dataset was recorded with several different operators, ensuring not only variability in terms of visual representation but also taking into account the fact that the same task may not be executed in the same way by different operators. However, due to limitations in the availability of human resources, such a thorough dataset could not be captured. Thus, as a compromise, it was decided that the dataset would be recorded only by one operator. Nevertheless, an effort was made to artificially introduce variability into the dataset, such as using different clothes, and performing tasks in different ways (e.g. change of dominant hand to perform certain tasks). Additionally, after the deep learning pipeline is fully developed and trained, the impact of this compromise should be evaluated by testing the model with data collected with different operators, and, if required, solutions to overcome this limitation should be proposed.

To record the dataset, a *Python* module that interacts with both the Application Programming Interface (API) of the RGB-D camera and the SDK of the skeleton tracking software was developed. The package starts by retrieving the RGB frames and depth map from the camera and performing the required alignment between them. The frames are then used to perform the 2D skeleton detection, which is then converted to three-dimensional coordinates using the data from the depth map. After the recording is stopped, the software processes and saves the data to the disk, using an *‘.mp4’* video file to store the RGB frames and a binary *Numpy* file to store an array with the skeleton information for each frame. The dataset was recorded with a resolution of  $1280 \times 720$  and a framerate of 8 fps due to the bottleneck introduced by the skeleton tracker (that can only be executed at  $\approx 10$  fps). Nevertheless, this framerate is deemed sufficient as it is not expected that humans can perform significant motions in less than 125 ms.

The recorded dataset is composed of 14 videos, each covering the full collaborative assembly process of the engine. On average, each video has a length of 7 minutes and 56 seconds and, in total, 53 264 frames were recorded. The dataset was divided using a 70%/15%/15% training/validation/test split. The split was performed randomly and reserves 10 videos for training and 2 for both validation and testing. Figure 5.2 displays an overview of the recorded videos.

## 5.2 Dataset Labeling

To implement supervised learning, the dataset needs to be properly labeled, which tends to be a cumbersome and labor-intensive task. Moreover, the nature of the application has a direct impact on the required labels and, thus, on the effort required to annotate the dataset. As an example, the annotation of images for a classification task requires that a label be attributed to the image as a whole, whereas the annotation of the same image for a semantic segmentation task requires a label





Figure 5.2: Overview of the recorded collaborative assembly dataset.

to be attributed to each pixel of the image. Naturally, this corresponds to a drastic difference in labor that is reflected in the cost of labeling the dataset.

For the recorded collaborative assembly dataset, the task performed by the model is the classification of video clips (i.e. a set of consecutive frames), attributing them with one of the following classes with a direct Operator Task (OT) correspondence in the proposed collaborative assembly plan (Figure 4.1):

- Class 0: No relevant action.
- Class 1: Assembly of oil pump outside the engine.
- Class 2: Placement of crankshaft caps and screws.
- Class 3: Screwing of crankshaft caps.
- Class 4: Assembly of oil pump inside the engine.
- Class 5: Placement of crankcase and screws.
- Class 6: Screwing of crankcase.
- Class 7: Change of robot screwdriver head.
- Class 8: Placement of oil sump and screws.
- Class 9: Screwing of oil sump.
- Class 10: Assembly of oil filter.

Since the classification is performed at the clip level, it would seem like a natural choice to attribute the labels directly to each clip. However, this would limit the flexibility of the dataset, as a labeling operation would have to be performed every time a different clip configuration was used (e.g. different clip sizes or clip overlap). Thus, the best option is to perform the annotation of each frame and use them to attribute a ground truth to the clips by selecting the most common frame label. It should be noted that this conversion is extremely accurate as most clips will be composed exclusively of frames from one class. Only clips in the transition of two human actions will be composed of frames with different labels. Nevertheless, the proposed conversion methodology is valid for both situations.



Having said that, since the full dataset is composed of 53 264 frames, the labeling process of all the frames is quite labor-intensive and time-consuming. To expedite this process, a *Python* console application was developed to perform the labeling of the frames. This application allows the user to select a video of the dataset, display the frames of the selected video, attribute a label to the frames of the video, and save the created annotations to a dedicated label file using a *Pandas DataFrame* data structure. Appendix B displays the developed application to facilitate the labeling of the frames.

One crucial characteristic of videos is the temporal relationship between frames. This means that consecutive frames will, most likely, correspond to the same class. This trait can be leveraged to perform a more efficient annotation of videos by aggregating them into frame blocks. As an example, if a video contains 100 frames of a given class and 200 frames of another, the labeling process of every individual frame would require 300 different annotations. However, they may be aggregated into frame blocks, requiring only 2 annotations to label the same 300 frames. This concept was applied to the developed video labeling application, through the implementation of breakpoints between frame blocks. The introduction of this simple concept allowed the recorded collaborative assembly dataset to be labeled using only 401 annotations, instead of the 53 264 that would have been required if the frames were annotated individually.

### 5.3 Dataloader implementation and optimization

The data preparation methodology is critical for any deep learning problem. As such, to complete the dataset implementation, it is necessary to develop a pipeline to quickly and efficiently retrieve the data. This is a crucial step as it will have a major impact on the training time of the models. Since the models will be implemented using *PyTorch* (an open-source machine learning framework), this was done using the recommended approach when working with this library:

- A **dataset class** that expands the *torch.utils.data.Dataset* abstract class and defines the dataset. This class is responsible for handling the data preparation and for loading the actual data.
- A **dataloader**, using the *torch.utils.data.DataLoader* iterator, which allows the effortless use of numerous data iteration features such as data batching, data shuffling, and parallel data loading using multiprocessing workers.

The dataset class is the most important component as it is responsible for loading the actual data. As such, it was implemented flexibly, so that different clip configurations could be obtained from the same videos. Particularly, the dataset class was developed in a way that allowed for the effortless configuration of the duration of the clip, i.e. the length of the clip (in seconds), and overlap, i.e. a ratio that defines how much two consecutive clips overlap (if the ratio is 0, there is no overlap between two consecutive clips), which may allow the clips to hold more temporal information or have increased temporal resolution, respectively. Additionally, the dataset class may provide only the video clips and their respective frame labels, or also include the three-dimensional skeleton data of the operator. The skeleton data is processed so that only the relevant

joints to the engine assembly (shoulders, elbows, and wrists) are used, resulting in a vector with 18 data points for each video frame (6 joints, each with 3 dimensions).

The aforementioned dataset splits were also implemented in the dataset class, allowing data to be retrieved from one of the training/validation/testing partitions or the dataset as a whole. Finally, a *torchvision.transforms* object may be passed to the dataset class, allowing data transformations to be applied to the clips before being retrieved. This feature may be used for data normalization or augmentation.

Regarding the methodology of loading the video clips, numerous approaches were considered and tested, including accessing the *.mp4* video directly with the use of appropriate video readers (such as the *VideoCapture* class of *OpenCV* or the *VideoReader* API provided by *torchvision*), or saving the video's frames as image files and using image processing libraries (such as *OpenCV*, *scikit-image*, or *Pillow*) to read them. All of these options had a similar performance, taking approximately 100 ms to retrieve a clip with 8 raw frames, i.e. frames with the full recorded resolution. However, this is quite inefficient, as the model will not receive the full resolution images as input but, instead, a scaled-down version. To solve this issue, and to speed up the clip loading times, the videos were pre-processed, saving the scaled-down versions of each frame to image files (using a *.png* format). The *Pillow* library was used to read the lower resolution frames, which are then converted to *PyTorch* tensors. Using this methodology, the same 8 frame clip can be loaded in approximately 9 ms, corresponding to a loading speed decrease of around 11 times. In turn, the loading efficiency of the skeleton information was not considered as it is negligible when compared to the clip data.

On the other hand, the implementation of the dataloader was quite simple. The only major contribution was the development of two custom collate functions, i.e. functions to combine several data samples into mini-batches. The difference between these functions is that one provides the labels for each frame of the clip (which are used for evaluation), while the other collapses the frame labels into a single class for each clip (which are used to train the models).

## Chapter 6

# Proposed Deep Learning Pipeline

This chapter presents the proposed deep learning models for the human action recognition task in video clips. The architecture of the models will be explained, highlighting the differences between them. Moreover, the developed training, validation, and evaluation process will be detailed.

### 6.1 Implemented models

Modern human action recognition methodologies mainly rely on deep learning approaches as they significantly outperform traditional methods using hand-crafted feature extraction techniques. Moreover, it was highlighted that an ideal model would not only use the spatial information of images but also leverage the temporal relationship between frames. As such, the developed human action recognition system focuses exclusively on deep learning approaches that use temporal information as input.

#### 6.1.1 Proposed architectures

From the main network architectures identified in the state-of-the-art to perform human action recognition directly on videos, only the ‘two-stream convolutional network’ approach was discarded due to the high computational complexity associated with the computation of the dense optical flow required for the temporal stream. This limitation is particularly relevant for the considered use case due to the real-time requirements of the collaborative assembly station. As such, the considered architectures are 3D Convolution Neural Networks (3D CNNs) and Recurrent Neural Networks (RNNs) using a regular CNN as a backbone (i.e. as a feature extracting network).

Even though these two architectures are quite different, they both employ convolutions (2D or 3D) to perform feature extraction. Thus, to ensure a fair and structured comparison between both architectures, they should have a common convolution structure. Even though several solutions could have been considered for this purpose, a residual network (ResNet) architecture, as proposed in [16], was selected. Figure 6.1 displays the residual learning block that serves as a basis for residual networks. The ‘shortcut connections’ do not add extra parameters or computational

complexity but allow the effortless training of deeper networks without noticeable accuracy saturation. In other words, the showcased residual learning block allows the implementation of deeper networks, generally resulting in a model with increased performance. This was the main reason why the residual network architecture was selected as the basis of the implemented models.

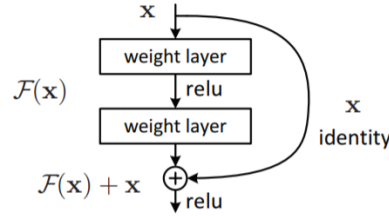


Figure 6.1: Residual learning block employed by residual networks (ResNets). Introduced in [16].

Regarding the 3D CNN model architecture, it is composed of the 3D CNN itself and a classification head. The former is a *ResNet 3D 18* (ResNet 3D with 18 layers), as proposed in [38], and is responsible for the feature extraction of the video clip as a whole, processing its spatial and temporal information and converting it into a single feature vector. In turn, the classification head, which is composed of three layers, converts the clip feature vector into an output vector (i.e. a vector with a confidence score for each class of the dataset). The hidden layer, which has a size of 200 neurons, is crucial as it allows the classification head to map any non-linear function between the feature vector and the output layer. Figure 6.2 shows a schematic view of the described model, which has a total of approximately 33 million trainable parameters and has an estimated total size of around 305 MB.

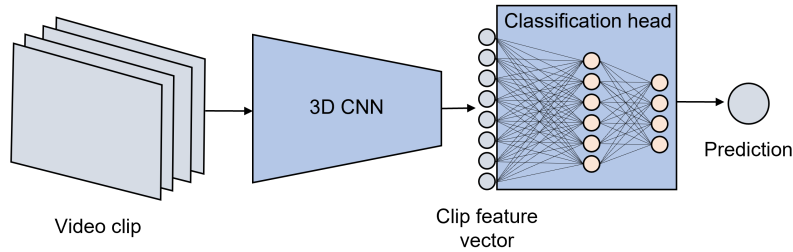


Figure 6.2: 3D CNN model architecture. A *ResNet 3D 18* [38] is used as the three dimensional convolutional neural network and the classification head is composed of three linear layers.

In turn, the proposed Recurrent Neural Network architecture is composed of a regular CNN, the RNN itself, and a classification head. The two-dimensional CNN serves as a feature extractor for each frame of the clip, converting them into frame feature vectors. The selection of the CNN could be made based on network depth or computation resource consumption (i.e. number of trainable parameters and total network size). Using the former approach, a *ResNet-18* model [16] was selected, as it has the same network depth as the considered 3D CNN. However, this network is much lighter and uses fewer computational resources when compared to its 3D counterpart (around 11.5 million trainable parameters and an estimated size of 130 MB). As such, a deeper CNN

(*ResNet-34* [16], with 34 layers) was also considered so that the comparison with the 3D CNN is somewhat fairer regarding the computational resource consumption. *ResNet-34* has around 22 million trainable parameters and an estimated size of 210 MB, which is still less than the *ResNet 3D 18*, allowing some resources to be allocated to the Recurrent Neural Network. Regarding the RNN, a Long Short-Term Memory (LSTM) [17] architecture was selected due to its ability to model long-term information through the use of a hidden state (the selected size of the hidden state was 100 features). The LSTM is responsible for processing the temporal information by converting the frame feature vectors into a single clip feature vector. Finally, the implemented classification head is the same as the one described for the 3D CNN model. It transforms the obtained clip feature vector into a prediction for the input clip. Figure 6.3 show the architecture of the proposed 2D CNN + RNN model.

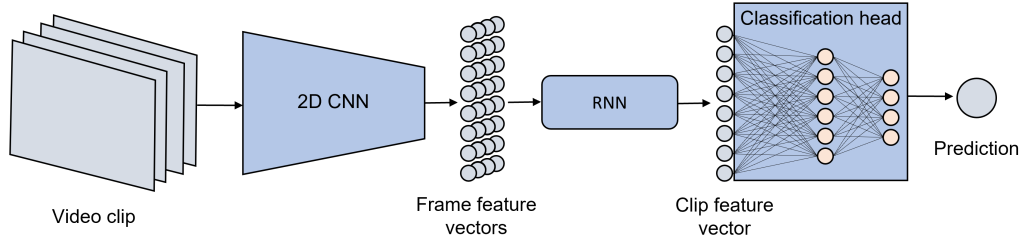


Figure 6.3: 2D CNN + RNN model architecture. A *ResNet-18* or *Resnet-34* [16] is used as the convolutional neural network, a *long short-term memory (LSTM)* is used as the RNN, and the classification head is composed of three linear layers.

### 6.1.2 Model Variations

Even though the three proposed architectures (*ResNet 3D 18*, *ResNet-18 + LSTM*, and *ResNet-34 + LSTM*) serve as a starting point for each of the models, there are several variations to the hyperparameters that may have an impact on their performance. Moreover, variations to the training methodology and data preprocessing may also influence the obtained results.

To measure the impact of these changes, a baseline was defined for each of the models, serving as a benchmark to evaluate the implemented variations. A direct comparison between the baseline and the models with each proposed variation was performed using the validation set to decide whether the considered variation will be employed in the final iteration of the model or not (the validation methodology will be further detailed in Subsection 6.2). With these considerations in mind, the implemented variations were the following:

#### Baseline

The defined baselines correspond to the simple model implementations described above. For the baseline models, only the video data was used (the collected 3D skeleton data was not utilized). The used video clips have a duration of 1s (8 frames per clip, considering the dataset has a frame-rate of 8 fps) and do not have any overlap (i.e. one clip begins where the previous one ended).

Moreover, the video clips were normalized using the mean and standard deviation of the *Kinetics-400*, a large dataset for human action recognition.

### **Transfer learning**

Transfer learning is a powerful tool in machine learning applications that allows the transfer of knowledge from a given task to a similar, yet different, problem. The idea behind this methodology is that large and variable datasets allow models to learn more accurate and complex relationships that cannot be obtained with a more limited dataset. Thus, transfer learning is particularly useful for improving the performance of a specific application by leveraging the knowledge obtained with a larger and more generic dataset.

This concept was applied to the 3D CNN model by initializing the weights of the 3D CNN with the final weights of an equivalent model trained on the *Kinetics-400* dataset. This is a direct employment of transfer learning as both applications are a task of human action recognition. Regarding the 2D CNN + LSTM models, transfer learning was applied by pre-training the CNNs on ImageNet, a large dataset for image classification. Even though the application is not exactly the same, it is expected that pre-training the CNN on such a large and variable dataset will allow the network to perform a better feature extraction on each frame, which is crucial for the human action recognition task.

### **Data augmentation**

Data augmentation is a methodology to artificially introduce variability into a dataset by applying random transformations (within defined limits) to the data. It is particularly relevant for smaller datasets with limited variability, as it has the potential to increase the quality of the dataset, making it more representative of any situation that may be encountered during inference.

The implemented data augmentation methodology employs small alterations of brightness, contrast, saturation, and hue in an attempt to increase the variability of the captured dataset. Moreover, a random horizontal flip with a probability of 50% was also implemented with the intention of making the model focus more on the performed action, instead of the spatial location of the operator or some assembly components (as the camera location is in a fixed position relative to the assembly station).

### **Clip duration**

The baseline model uses a duration of 1s for each clip. To assess the impact of using larger clips, with more temporal information on the performed action, this variation will use clips with a duration of 2s (which corresponds to 16 frames, given the dataset framerate of 8 fps). Nevertheless, it should be highlighted that, unlike the other variations, increasing the clip duration has a significant overhead as it requires more data to be continuously stored and used during inference. Moreover, it slows down the classification of the clips, as it requires more frames to be captured before being processed by the model.

### **Clip overlap**

Another variation that may be introduced at the clip level is their overlap, i.e. the ratio of shared frames between two consecutive clips. The baseline models use no overlap, meaning that each

frame is used in a single clip. Nevertheless, it would be interesting to assess if the use of clip overlap has any significant impact on the performance of the model, as it may provide more temporal resolution to the network predictions. To evaluate this theory, a clip overlap of 0.5 was implemented. Using this overlap, a clip with 8 frames will share its first 4 frames with the previous clip and its last 4 frames with the following clip.

### **Skeleton data**

In addition to the video clips, the captured 3D skeleton data may be used in an attempt to enhance the developed models and increase the performance of the human action recognition task. For the case of the 2D CNN + LSTM models, this addition is quite simple, as the 3D skeleton coordinates captured in each frame can be appended to the corresponding frame feature vectors before being processed by the LSTM. Using this method, the 2D CNN processes the spatial information from the video and the LSTM processes temporal information from both the video and the skeleton coordinates. However, for the proposed 3D CNN architecture, this is not as trivial as it would require an additional model (such as an RNN) to process the temporal information of the skeleton data. As such, to avoid the modification of the proposed models, the skeleton data was only used for the 2D CNN + LSTM architectures.

### **Stacked LSTM**

LSTM layers may be stacked with the intention of trying to improve their capability of learning the temporal relationships of the data and, thus, improve the network's performance. With this idea in mind, one of the implemented model variations used 3 stacked LSTM layers instead of the single LSTM used in the baseline models. Naturally, this variation is only relevant to the 2D CNN + LSTM models.

### **Class weights**

The considered collaborative assembly dataset consists of videos of the full assembly process of the engine. Since not all actions have the same expected length, it is normal that some classes are more represented in the dataset than others. This may result in a problem to the model, as the training process tends to give more importance to some classes when compared to others. To overcome this issue, it is possible to attribute the classes with a weight inversely proportional to their class distribution in the dataset. This method results in a higher training weight for the less represented classes, which should allow the training process to give approximately equal importance to all classes.

## **6.2 Training and evaluation details**

The three proposed model architectures (as well as all their relevant variations) were trained end-to-end using the training partition of the captured collaborative assembly dataset. The models were trained for 20 epochs, as experiments with additional training time did not reveal any advantage regarding the convergence of the model.

The training process was performed using an Adam optimizer [21] and a Cross-Entropy Loss function. A relatively small mini-batch size of 5 video clips was used due to memory limitations of the used GPU (*NVIDIA GeForce GTX 1650*).

The evaluation of the models was performed at the frame level. As such, the frame predictions must be inferred from the output of the networks, which provide a classification of the clips. This conversion is direct when no clip overlap is used, as each frame is present in only one clip. However, when clip overlap is employed, a decision must be made regarding the classification of the frame. In these situations, the confidence score of the clips was used to decide on the classification of the frame, i.e. the frame is classified with the same label as the clip with the highest confidence score.

The metrics used to evaluate the models were accuracy, confusion matrix, and Intersection over Union (IoU). The accuracy score is the most common evaluation metric for classification problems. Since it is a simple ratio of correct predictions over total predictions performed by the model, it provides an indication of the overall performance of the model. In turn, the confusion matrix provides a visual indication of the predicted classes in relation to the ground truth. This metric is not only extremely useful to identify potential performance discrepancies between classes but also possible systematic errors (e.g. the model cannot distinguish between two classes). Finally, the Intersection over Union metric was also implemented. This metric, which is mostly used to evaluate object detection tasks, is traditionally defined as the ratio between the area overlap between the model's prediction and ground truth, and its corresponding area of union. Nevertheless, this metric may be adapted to video classification tasks using the temporal domain. As such, the implemented temporal IoU is defined as the ratio between the intersection between the frame predictions and the ground truth, and their corresponding union. This metric was calculated individually for each label of the dataset, providing a measurement of the performance of the models for each class. Moreover, the mean Intersection over Union (mIoU) was also calculated as an average of the IoU of all classes. This provides an indication of the overall performance of the network but, unlike the accuracy score, gives equal importance to every class regardless of their distribution on the dataset.

All the model baselines and their respective variations were evaluated using the validation data partition. The model variations were compared to their respective baseline using the accuracy and mIoU metrics, as they provide an overall performance score of the model. To minimize the impact of the random variations to the model state at the end of the training process, the validation was performed at the end of the last 5 training epochs (epochs 16 to 20), as the models have already fully converged at this training stage. The average accuracy and IoU over those 5 evaluations is then calculated and used to select the variations that lead to increased network performance. A final iteration for each of the proposed architectures was then trained using the variations that lead to increased performance. A similar validation methodology was employed to perform a robust comparison between the final version of each architecture, allowing the selection of the best overall model.



After the full validation process is completed, resulting in the best developed architecture, the test data partition will be used to perform a final evaluation of the model. This is crucial as it allows the final network to be evaluated using data that was not used for training or validation, allowing for an unbiased and representative evaluation process with valid results. Additionally, the test set will also be used to simulate a real scenario of the collaborative engine assembly, allowing conclusions to be drawn on the performance of the proposed deep learning pipeline to enhance the collaboration. Finally, the network will also be tested using data collected with other operators to evaluate if the simplifications performed during the dataset collection stage have an impact on its ability to be used with different operators.



## Chapter 7

# Results and Discussion

In this chapter, the obtained results will be showcased and discussed. The results of the implemented validation procedure will be outlined, culminating in a final model architecture with the best variations for the task at hand. This final model will then be tested using not only performance metrics but also task-oriented approaches.

### 7.1 Model validation

The training and validation methodology described in Section 6.2 was implemented and the details will be presented in this section. Intending to simplify the exposure of the obtained outcomes, only an overview of the results will be showcased. The training details, in particular, will not be presented to avoid excess cluttering of information. Nevertheless, the full training and validation results for the baseline model of each architecture may be consulted in Appendix C.

As such, for each model architecture, only the average performance metrics of the baseline and respective variations will be shown, allowing for a comparison to be made between them. Additionally, a final model will be proposed for each architecture based on the results of each model variation. Ultimately, the final models for each architecture will be compared to each other, allowing for a decision to be made on the best architecture for the task at hand.

#### Resnet 3D 18

Table 7.1 shows the average results obtained during the validation of the baseline of the *ResNet 3D 18* model and its variations. The color of the cells is related to the comparison of the considered variation with the defined baseline (green cells correspond to a performance better than the baseline, whereas red cells correspond to a worse performing network when compared to the baseline). As it may be seen, the baseline model achieved satisfactory results, having reached an accuracy of 88.81% and a mIoU of 79.15%. However, the performance of the baseline model was quite different depending on the considered classes. This is particularly noticeable for class 7 ('Change of robot screwdriver head'), where the IoU was only 16.22%.

Regarding the model variations, the introduction of transfer learning, data augmentation, and clip overlap lead to improvements to the overall performance of the network, even if at the cost of the performance of some classes. This is particularly noticeable for the case of data augmentation, where the overall performance increased even when 5 different classes had a decrease of the IoU metric. Nonetheless, these variations will be included in the final model of the ResNet 3D 18 architecture. Moreover, it should be noted that the introduction of transfer learning lead to the best overall results for this model architecture by quite a significant margin.

In turn, the variation of clip duration and the introduction class weights during training lead to a decrease in the performance of the model. For the case of the clip duration, the decrease was particularly noticeable, especially since this model variation introduces an overhead during inference time. Moreover, the introduction of class weights did not contribute to attributing more importance to less represented classes, as may be seen by the decrease of the IoU of class 7 (which is the least represented class in the dataset).

Table 7.1: Validation of the *ResNet 3D 18* model variations.

		Baseline	Transfer Learning	Data augmentation	Clip Duration	Clip Overlap	Class weights
<b>Accuracy</b>		88.81%	93.51%	89.79%	87.96%	91.48%	88.75%
<b>mIoU</b>		79.15%	87.74%	80.89%	77.15%	82.13%	78.69%
<b>IoU</b>	<b>0</b>	66.09%	76.93%	69.43%	64.30%	75.67%	67.16%
	<b>1</b>	89.06%	91.25%	85.68%	85.38%	91.37%	89.95%
	<b>2</b>	96.00%	95.96%	94.50%	86.30%	95.44%	95.78%
	<b>3</b>	90.46%	95.68%	93.34%	86.29%	91.42%	93.56%
	<b>4</b>	91.77%	94.09%	94.31%	92.20%	94.16%	93.86%
	<b>5</b>	72.92%	84.67%	74.35%	76.15%	80.81%	70.55%
	<b>6</b>	91.40%	96.16%	92.89%	91.67%	94.71%	91.42%
	<b>7</b>	16.22%	68.50%	43.76%	24.22%	28.66%	5.75%
	<b>8</b>	91.18%	94.16%	77.04%	79.42%	94.17%	87.94%
	<b>9</b>	92.07%	93.75%	91.79%	94.51%	94.88%	89.70%
	<b>10</b>	73.50%	74.02%	72.69%	68.26%	62.10%	79.88%

## ResNet-18 + LSTM

Table 7.2 shows the average results obtained during the validation of the baseline of the *ResNet-18 + LSTM* model and its variations. The baseline network achieved an accuracy of 89.77% and a mIoU of 81.02%.

The introduced variations that lead to an increase in the overall performance of the model and, thus, will be conserved in its final iteration, were transfer learning, clip overlap, skeleton data, stacked LSTM, and class weights. Nevertheless, only the introduction of transfer learning lead to a significant increase in performance, while most variations only lead to a marginal performance improvement. In fact, even though the introduction of class weights contributed to an overall performance increase of the network, it slightly decreased the performance of the least represented class (class 7), meaning that it did not have the intended effect. Nonetheless, since these variations

still contributed to slight overall performance improvements, they will be employed in the final iteration of this architecture.

In turn, the variations of data augmentation and, especially, clip duration contributed to a significant decrease in the performance of the network. Thus, these variations will not be included in the final version of the *ResNet-18 + LSTM* model.

Table 7.2: Validation of the *ResNet-18 + LSTM* model variations.

		Baseline	Transfer Learning	Data augmentation	Clip Duration	Clip Overlap	Skeleton data	Stacked LSTM	Class weights
<b>Accuracy</b>		89.77%	93.54%	89.29%	87.28%	90.72%	90.65%	90.22%	90.28%
<b>mIoU</b>		81.02%	88.04%	79.86%	78.35%	81.77%	82.16%	81.28%	81.81%
<b>IoU</b>	<b>0</b>	67.82%	77.13%	67.73%	62.42%	73.03%	73.47%	70.44%	67.74%
	<b>1</b>	86.89%	90.05%	83.18%	73.91%	86.24%	90.42%	90.42%	91.13%
	<b>2</b>	96.37%	95.99%	96.16%	94.21%	96.05%	97.78%	94.34%	97.60%
	<b>3</b>	92.35%	95.68%	92.23%	87.43%	95.88%	93.88%	92.56%	94.11%
	<b>4</b>	94.12%	95.20%	93.87%	87.32%	92.72%	94.16%	93.85%	94.41%
	<b>5</b>	72.92%	83.21%	72.14%	78.70%	74.48%	70.84%	73.22%	73.20%
	<b>6</b>	92.52%	95.81%	94.85%	87.91%	96.44%	94.23%	95.19%	95.03%
	<b>7</b>	36.83%	68.44%	36.82%	52.20%	34.60%	31.69%	37.93%	35.57%
	<b>8</b>	89.69%	93.44%	80.04%	82.15%	90.79%	87.05%	83.62%	84.04%
	<b>9</b>	92.13%	94.40%	92.27%	84.26%	95.75%	94.93%	91.57%	93.60%
	<b>10</b>	69.60%	79.12%	69.18%	71.36%	63.45%	75.27%	70.93%	73.46%

### ResNet-34 + LSTM

Table 7.3 displays the average results obtained during the validation of the baseline of the *ResNet34 + LSTM* model and its variations. The baseline for this architecture reached an accuracy of 89.96% and a mIoU of 82.03%.

Table 7.3: Validation of the *ResNet-34 + LSTM* model variations.

		Baseline	Transfer Learning	Data augmentation	Clip Duration	Clip Overlap	Skeleton data	Stacked LSTM	Class weights
<b>Accuracy</b>		89.96%	93.89%	89.13%	88.53%	91.84%	89.32%	88.65%	89.01%
<b>mIoU</b>		82.03%	89.28%	79.28%	78.73%	84.37%	80.03%	79.39%	79.36%
<b>IoU</b>	<b>0</b>	67.19%	75.52%	67.43%	64.65%	74.66%	69.27%	65.92%	65.29%
	<b>1</b>	80.09%	91.93%	88.94%	88.12%	91.60%	85.04%	79.60%	87.68%
	<b>2</b>	97.78%	97.13%	95.55%	94.00%	97.23%	97.01%	93.74%	96.73%
	<b>3</b>	93.71%	95.45%	91.43%	88.69%	94.44%	94.61%	90.27%	88.39%
	<b>4</b>	93.61%	93.15%	93.46%	90.85%	94.04%	89.51%	93.65%	95.29%
	<b>5</b>	77.77%	93.16%	71.39%	78.92%	74.20%	72.60%	74.42%	73.38%
	<b>6</b>	94.40%	94.18%	92.08%	94.19%	95.74%	92.62%	91.02%	93.81%
	<b>7</b>	58.95%	82.29%	33.36%	36.00%	46.96%	28.96%	34.10%	29.59%
	<b>8</b>	80.26%	90.54%	88.33%	85.64%	90.68%	89.16%	84.91%	80.64%
	<b>9</b>	94.46%	95.12%	92.00%	90.19%	93.21%	94.07%	91.61%	89.63%
	<b>10</b>	64.10%	73.64%	58.16%	54.83%	75.38%	67.56%	74.04%	72.53%

As with the *ResNet 3D* architecture, the variations of transfer learning and clip overlap were the only ones that lead to an increase in the model’s performance. As such, only these variations will be employed in the final iteration of this architecture. Moreover, it should be highlighted that, once again, the introduction of transfer learning lead to the best results during the validation of this architecture.

The remaining variations (data augmentation, clip duration, skeleton data, stacked LSTM, and class weights) all lead to networks with slightly worse performance. This supports the idea that some of the variations that lead to marginal improvements of the *ResNet18 + LSTM* architecture do not correspond to sustained changes with consistent improvements to the network. Regardless, since these alterations resulted in inferior results, they will not be used in the final version of the *ResNet34 + LSTM* architecture.

### Architecture selection

Table 7.4 displays the average evaluation (still using the validation set) of the final models for each of the proposed architectures. The color of the cells is related to the comparison between the final model and the defined baseline for each architecture (green and red cells indicate that the final model performed better or worse than the baseline, respectively). The absence of red cells indicates that every final model outperformed the corresponding baseline in all presented metrics. Moreover, considering the general metrics (accuracy and mIoU) the final networks outperformed every single model variation presented in Tables 7.1, 7.2, and 7.3. The full training and validation results of the final iterations of each architecture may be consulted in Appendix C.

The final models of each proposed architecture had somewhat similar results to each other, with accuracy and mIoU of approximately 95% and 91%, respectively. Nevertheless, the differences between the architectures, even if slight, are not negligible, especially considering that the presented results are the average of 5 different evaluations (thus decreasing the impact of the final state of the models at the end of the training process). With this consideration in mind, the

Table 7.4: Validation of the final models for each proposed architecture.

		<b>ResNet 3D 18</b>	<b>ResNet-18 + LSTM</b>	<b>ResNet-34 + LSTM</b>
<b>Accuracy</b>		95.15%	94.83%	95.45%
<b>mIoU</b>		91.13%	90.26%	91.68%
<b>IoU</b>	<b>0</b>	79.76%	78.23%	81.40%
	<b>1</b>	93.51%	94.78%	93.09%
	<b>2</b>	97.53%	97.35%	97.80%
	<b>3</b>	94.17%	92.57%	95.08%
	<b>4</b>	94.65%	94.45%	94.54%
	<b>5</b>	92.63%	90.75%	90.99%
	<b>6</b>	97.05%	97.78%	97.71%
	<b>7</b>	85.25%	83.73%	80.92%
	<b>8</b>	95.39%	95.99%	96.28%
	<b>9</b>	95.87%	96.42%	96.59%
	<b>10</b>	76.63%	70.81%	84.15%

*ResNet 3D 18* model outperformed the *ResNet-18 + LSTM* network. However, even though the CNNs of both these models have a similar depth, the comparison may be considered unfair due to the greater computational resource consumption of the 3D CNN. The results of the *ResNet-34 + LSTM* model support this idea as it outperformed the *ResNet 3D 18* model using a deeper CNN but more comparable resource consumption. Thus the architecture that achieved the best results is the deeper *ResNet-34 + LSTM*, reaching a validation accuracy of 95.45% and a mIoU of 91.68%. Additionally, only 3 classes (class 0 - ‘No relevant action’, class 7 - ‘Change of robot screwdriver head’, and class 10 - ‘Assembly of oil filter’) had an IoU of less than 90% using the final model of this architecture (the minimum registered value was 80.92% for class 7). This is the network that will be used for testing.

## 7.2 Model testing

The results obtained with the performed validation procedure revealed that the best model architecture for the considered application was the *ResNet-34 + LSTM*. The final iteration of this model employed a clip overlap of 0.5, with a duration of 1s, and without data augmentation. Only data from the recorded videos was used, discarding the estimated 3D skeleton of the operator. Regarding the model itself, transfer learning was utilized and a single LSTM layer was used. The network was trained without using class weights. This model configuration was evaluated with the test partition of the dataset using the final model state after the full 20 epochs of training. The obtained results will be presented in the current section.

### Evaluation

Table 7.5 shows all the performance metrics of the model on the test set. As it may be seen, the network had a great performance, achieving an accuracy of 96.65% and a mIoU of 94.11%. Additionally, it should be highlighted that the performance of the model on the test set was even better than the one registered with the validation set. This supports the idea that the architecture decisions carried out during the validation procedure did not introduce any additional bias to the network. Additionally, using the test set, only 2 classes (class 0 - ‘No relevant action’, and class 7 - ‘Change of robot screwdriver head’) registered an IoU below 93%, which is also an improvement relative to the results of the validation set.

In turn, Figure 7.1 displays the confusion matrix obtained for the performed evaluation (the values are normalized with the total number of frames for each class). These results indicate, once again, a great model performance, as it may be seen by the prominent diagonal of the confusion matrix. This metric is also extremely useful to identify the errors of the model. For this case, two main issues may be spotted. The first is the misidentification of several frames as class 0 (‘No relevant action’), as it may be seen by the slight presence of color in the first column of the confusion matrix. However, since the class of ‘No relevant action’ is extremely common throughout the assembly process, these false predictions likely correspond to small errors in the transitions between classes and, thus, the impact of these misidentifications on the final application

Table 7.5: Performance of the final model (ResNet-34 + LSTM) on the test set.

		ResNet-34 + LSTM
<b>Accuracy</b>		96.65%
<b>mIoU</b>		94.11%
<b>IoU</b>	<b>0</b>	82.53%
	<b>1</b>	94.64%
	<b>2</b>	96.76%
	<b>3</b>	97.15%
	<b>4</b>	95.95%
	<b>5</b>	93.73%
	<b>6</b>	96.68%
	<b>7</b>	85.98%
	<b>8</b>	97.76%
	<b>9</b>	98.09%
	<b>10</b>	95.92%

should be negligible. The second problem is the appearance of a significant model prediction of class 5 ('Placement of crankcase and screws') when the ground truth is class 7 ('Change of robot screwdriver head'), indicating the model may not be able to perfectly differentiate between these classes. In fact, this seems to be a systematic problem with the proposed models, as it may be seen by the confusion matrices displayed in the full validation results presented in Appendix C. An analysis of the recorded dataset revealed visual similarities between frames of both these classes, as the operator tends to reach forwards to interact with either the back of the crankcase or the tip of the robot screwdriver, which are close to each other. This is believed to be the reason why the networks systematically tend to have trouble differentiation between these classes. Nevertheless, even though this problem was much more prevalent in previously proposed model variations, the final network used for testing was able to minimize this problem to a great extent. In fact, as it was shown in Table 7.5, this model achieved an IoU score of 93.73% and 85.98% for classes 5 and 7, respectively, which may be considered good results, especially since these actions are not used as triggers for the proposed collaborative assembly plan.

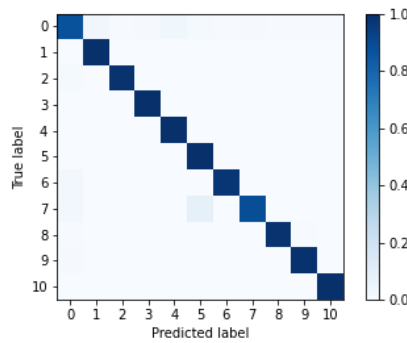


Figure 7.1: Confusion matrix of the final model (ResNet-34 + LSTM) on the test set.

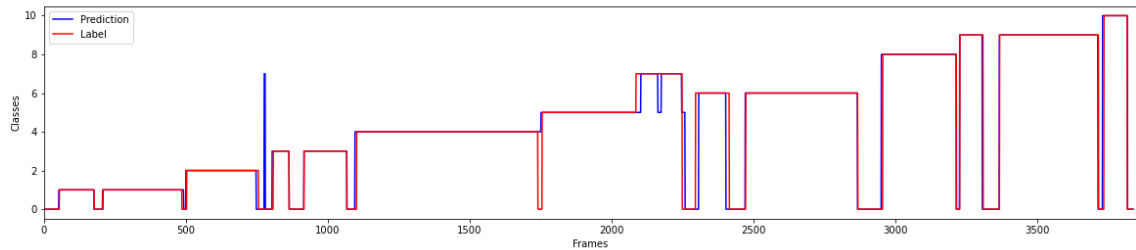


### Task-oriented evaluation

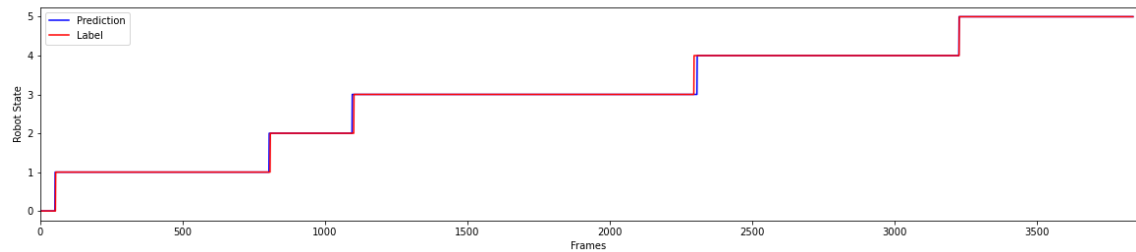
In addition to the previously presented model evaluation, a task-oriented evaluation was also performed, using the videos of the test partition of the dataset to simulate an instance of the collaborative assembly of the engine. The only difference between this test and the intended application is that the predictions are computed offline and, thus, the model does not send commands to the robot. Nevertheless, with this approach, it is still possible to assess the predictions of the model and infer the corresponding robot triggers.

The plot in Figure 7.2a shows a visual representation of the described evaluation, where the evolution of the ground truth and model predictions for the human action of each frame are represented. As it may be seen, the performed human action recognition was extremely accurate, with just some occasional and brief errors. In fact, most of the time, the plotted lines are almost indistinguishable, highlighting the accurateness of the implemented network.

Additionally, the robot state machine represented in Figure 4.2 was implemented, allowing the conversion of the human action recognition data into robot states that would have been commanded by the developed cognitive system. The plot in Figure 7.2b shows the comparison between the ideal robot state and its state when controlled by the model predictions. As it may be seen, the plotted lines are almost indistinguishable, meaning that the proposed human action recognition system achieved an almost perfect transition between robot states. These results support the idea that slight inaccuracies of the deep learning network may not have an impact on the state of the robot, which is the end goal of the proposed system.



(a) Comparison between ground truth and model predictions for the operator actions.

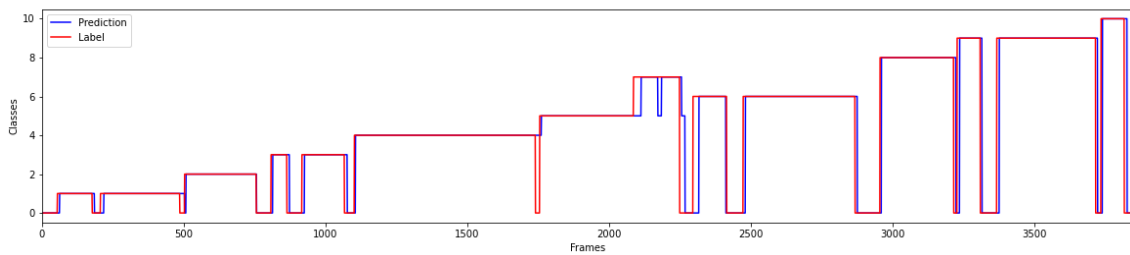


(b) Comparison between the ideal robot state and the one controlled by the human action recognition system.

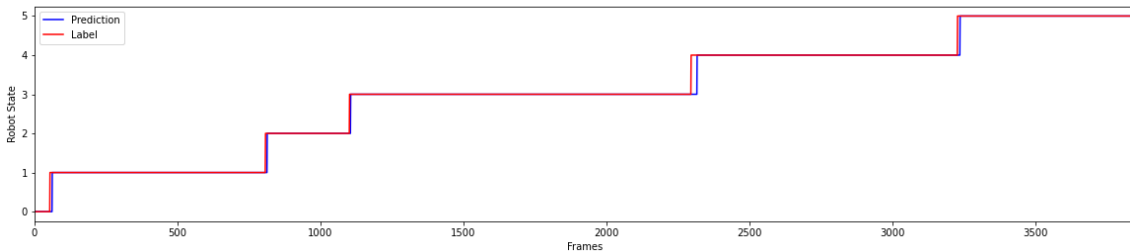
Figure 7.2: Task-oriented evaluation of the final model (ResNet-34 + LSTM).

Nevertheless, even though the small inaccuracies of the model's predictions did not have any impact on the final state of the robot, it does not mean that they could not have led to false robot state transitions. As an example, the brief misdetection around frame 750 of Figure 7.2a could have triggered a false robot transition if the model had wrongfully identified a triggering action. However, since the application at hand pertains to the detection of human actions, it is possible to construct the assumption that extremely short actions probably correspond to prediction errors, as it is unlikely that a human action can occur in less than, for example, 1s (at least considering the task-oriented actions of the use case at hand). With this assumption, a temporal consistency filter can be constructed to remove brief predictions, considering them as misdetections. The drawbacks of this filter are twofold. Not only does it decrease the temporal resolution of the performed classifications but it also delays the detection of the transitions, causing the robot to have a slower response. As an example, Figure 7.3a represents the predictions of the same video using a temporal filter of 2.5 seconds. As it may be seen, this filter removed the misdetection around frame 750. Moreover, upon close inspection, it is also possible to observe an overall delay in the predictions when compared to the original unfiltered results (Figure 7.2a). This is a side effect of the introduction of the temporal consistency filter and the delay magnitude is directly proportional to the size of the used filter window.

Additionally, the filtered predictions were, once again, used to compute the evolution of the robot state, which is represented in Figure 7.3b. Apart from the introduction of the delay associated with the filter, these results were in every way similar to the previously obtained robot states. Nevertheless, the introduction of this filter may be a good compromise for potentially avoiding mistriggers of the robot state at the cost of a slower system response.



(a) Comparison between ground truth and model predictions of the operator actions.



(b) Comparison between the ideal robot state and the one controlled by the human action recognition system.

Figure 7.3: Task-oriented evaluation of the final model (ResNet-34 + LSTM) using a temporal consistency filter of 2.5 seconds.

Finally, it should be underlined that the presented predictions are almost coincident with the ground truth because this classification was performed offline and the data is being presented at the frame level. In the real scenario, where the human action recognition system is being executed online, this evaluation should be performed in time units due to the delays introduced by the data processing and model inference times. Nevertheless, it is expected that these delays will not have a significant impact on the performance of the system as a whole.

### 7.3 Generalization assessment

As was described in Chapter 5, the limited human resources available for the recording of the dataset resulted in a compromise where only one operator participated in the collection of the data. Naturally, this simplification may have caused the trained models to be overfitted to the operator present in the dataset and, thus, impacted the ability of the network to be used to perform action recognition with any person. To test this hypothesis, data of the collaborative engine assembly process was collected with two additional operators. The developed model was used to perform the classification of the new data, allowing for a comparison to be drawn between the performance of the model for the operator of the original dataset and for other operators.

Table 7.6: Performance of the final model (ResNet-34 + LSTM) with data collected with other operators.

		Operator A	Operator B
<b>Accuracy</b>		77.66%	82.32%
<b>mIoU</b>		63.82%	75.91%
<b>IoU</b>	<b>0</b>	66.00%	66.83%
	<b>1</b>	90.41%	87.55%
	<b>2</b>	64.63%	97.5%
	<b>3</b>	98.23%	68.4%
	<b>4</b>	65.37%	83.03%
	<b>5</b>	13.13%	28.78%
	<b>6</b>	59.13%	34.98%
	<b>7</b>	76.60%	96.15%
	<b>8</b>	14.65%	97.18%
	<b>9</b>	60.07%	87.15%
	<b>10</b>	93.83%	87.5%

Table 7.6 shows the performance metrics of the model using the data collected with the two new operators. The color of the cells is a direct comparison with the final metrics displayed in Table 7.5. In turn, Figure 7.4 displays the corresponding confusion matrices of the model. The presented results show that the model was able to leverage the knowledge learned with the original dataset to perform a somewhat accurate classification of the videos collected with the two new operators. This idea is also supported by the displayed confusion matrices, where the main diagonal is still quite prominent. Nevertheless, in comparison to the results obtained with the test partition of the original dataset, these results are quite poor. In fact, the presented indicators show a significant performance drop for almost all metrics - the accuracy dropped from 96.65% to

77.66% (operator A) and 82.32% (operator B), and the mIoU decreased from 94.11% to 63.82% (operator A) and 75.91% (operator B). It should also be highlighted that the performance of the model for the Operator B was significantly better than for the Operator A. This discrepancy was probably caused by the way each person performed the assembly tasks. Since the performance of the action recognition system was superior with the data of operator B, the tasks were likely performed in a more similar way to the operator of the original dataset.

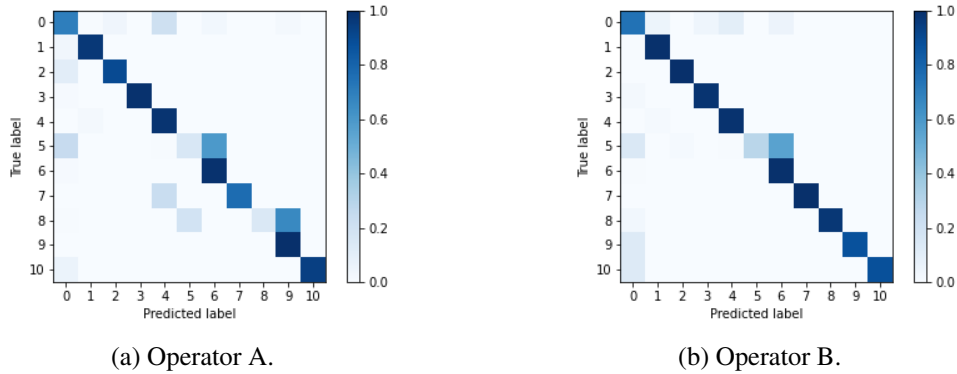


Figure 7.4: Confusion matrices of the final model (ResNet-34 + LSTM) with data collected the other operators.

This inability to be generalized to any operator is the main limitation of the proposed deep learning methodology. As such, intending to overcome this shortcoming, two different solutions may be proposed. The first is the creation of a larger dataset with the contribution of several operators. This is considered to be the optimal solution as it would allow the model to be trained with a variable dataset representative of the domain in which it will be implemented. However, this approach would require a considerable initial cost in human resources for the recording of the dataset. Moreover, this approach would also require an evaluation of its ability to work with other operators (that did not participate in the recording of the dataset) to assess if the problem was surpassed. The other solution is to fine-tune the model by training it with a few instances of data collected with the operator that will be using the system. This is a direct application of transfer learning as the knowledge obtained with the larger (yet less variable) dataset collected by only one operator is being leveraged to facilitate the training process with data collected with the other operator. The idea behind this approach is that it will allow the model to reach a high-performance level using just a few examples for training and, thus, avoid long and labor-intensive dataset collection operations. The main disadvantage of this approach is that it requires a given operator to record a few instances of the engine assembly process before being able to use the action recognition system to its full capabilities. Nevertheless, since the model will be fine-tuned specifically for the considered operator, it is possible that higher levels of performance may be reached.

## Chapter 8

# Conclusions and Future Work

The present dissertation aimed to develop computer vision techniques empowered by deep learning to enhance the collaboration between human and robot in the use case of an internal combustion engine assembly. A literary review on the topic was performed with a clear focus on the interpretation of implicit communication cues, as they do not put any additional burden on the operator. The main contribution of this study was the development and implementation of a deep learning pipeline to perform human action recognition and, thus, infer the context of the assembly process.

A collaborative assembly plan was defined to leverage the information obtained by the proposed cognitive system with the intention of commanding the robot's actions efficiently, ensuring a close and effortless collaboration with the operator. This was achieved by using the detection of certain operator actions as triggers for the robot to initiate its task.

A dedicated dataset was recorded for the use case of the considered collaborative assembly station composed not only of video recordings of the operation but also high-level three-dimensional skeleton information of the operator. Due to human-resource limitations, only one operator was used for the recording process. A dataloader was developed and optimized, ensuring that the data loading operation was as efficient as possible and, thus, was not a bottleneck for the training of the models. Additionally, a video labeling application was developed to expedite the dataset labeling process. This was considered an important contribution since data labeling is often an expensive and labor-intensive operation. With the proposed approach, the 53 264 frames of the recorded dataset were labeled using only 401 annotations.

Three different model architectures using deep learning methodologies were proposed to perform the human action recognition directly with video clips, leveraging not only the spatial information of the frames but also the temporal relationships between them. The suggested models were *ResNet 3D 18*, *Resnet-18 + LSTM*, and *ResNet-34 + LSTM*. Different model and data variations were proposed for each architecture and a validation procedure was carried out to select the best variations for each architecture. Additionally, a comparison between the model architectures was also performed, allowing the selection of the *ResNet-34 + LSTM* network as the best model for the task at hand.

The final model was evaluated with an adequate data partition, having achieved an accuracy of 96.65% and a mIoU of 94.11%. Moreover, a task-oriented evaluation was also performed, using videos of the test set to perform an offline simulation of the proposed collaborative assembly system. This evaluation showed that the action recognition system had a great performance with only a few brief misidentifications. Furthermore, the action recognition data was used to simulate the commanded robot actions, showing close to perfect accuracy. Nevertheless, with the intention of minimizing the probability of false robot triggers caused by potential action misdetections, a temporal consistency filter was implemented. This filter showed potential to correct brief errors of the model at the cost of slowing down the robot response time.

Finally, an assessment on the impact of using different operators was made to evaluate its effect on the performance of the human action recognition system. The obtained results showed that even though the model was able to use some of the previously obtained knowledge, there was a significant performance drop when compared to the data of the original operator. To solve this shortcoming two solutions were proposed - the recording of a larger and more variable dataset with several operators, and the employment of transfer learning to fine-tune the base model for the considered operators. The former requires higher initial costs but is the only solution to develop a model that may be used by any person without previous adjustments. In turn, the latter solution may lead to better performance (as the model will always be fine-tuned for the considered operator) at the cost of a new operator not being able to use the system without any prior adjustments.

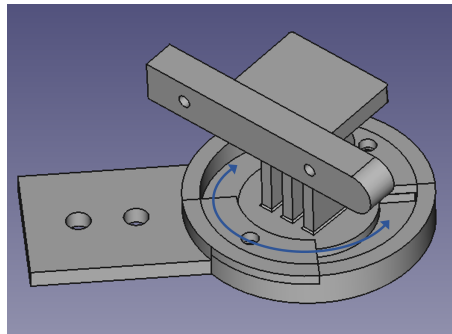
Regarding future work, the proposed solutions to overcome the highlighted model limitation should be implemented and evaluated. The model fine-tuning approach should be evaluated before investing in the recording of the larger and more variable dataset as it requires fewer costs. Additionally, the proposed system should be integrated into the presented collaborative assembly station so that it may be evaluated online while actively participating in the assembly process. To achieve this, the model may be integrated with a Robot Operating System (ROS) node so that it may effortlessly communicate with the remaining system.

Moreover, the development of further cognitive systems to monitor the operator is also planned as future work for the progressive development of the collaborative cell. As an example, eye gaze data may be used to anticipate the actions of the operator. This paradigm shift from reactive to anticipatory control would allow the system to have faster response times. Additionally, safety features should be implemented on the cell allowing closer collaboration between the operator and the robot. One simple safety feature would be the use of eye gaze data to ensure that the operator is paying attention to robot movements in its vicinity. A more complex solution would be the use of the 3D skeleton data of the operator to implement a dynamic Speed and Separation Monitoring safety strategy, allowing for the robot to slow down or adapt its path to ensure that the safety of the operator is never at risk.

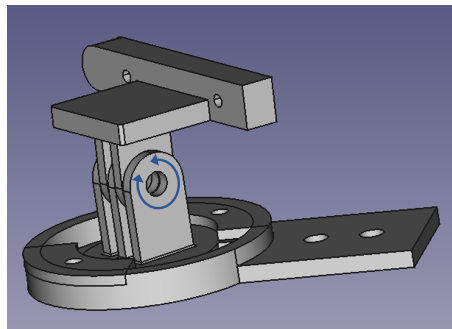
## Appendix A

### Camera support prototype

A support prototype for the *Intel Realse D435* camera was modeled and 3D printed to be integrated in the cell. With the intention of allowing for quick adjustments, the support was design with 2 degrees of freedom. Figure A.1 displays the designed support. The depicted blue arrows represent the two degrees of freedom. Both angles can be easily adjusted manually and locked in place using bolts. The camera is secured to the top of the support with two M3 bolts. In turn, Figure A.2 shows the position where the RGB-D camera was mounted within the collaborative assembly station.



(a)



(b)

Figure A.1: Three-dimensional model of the adjustable camera support. The blue arrows represent the two degrees of freedom of the support.



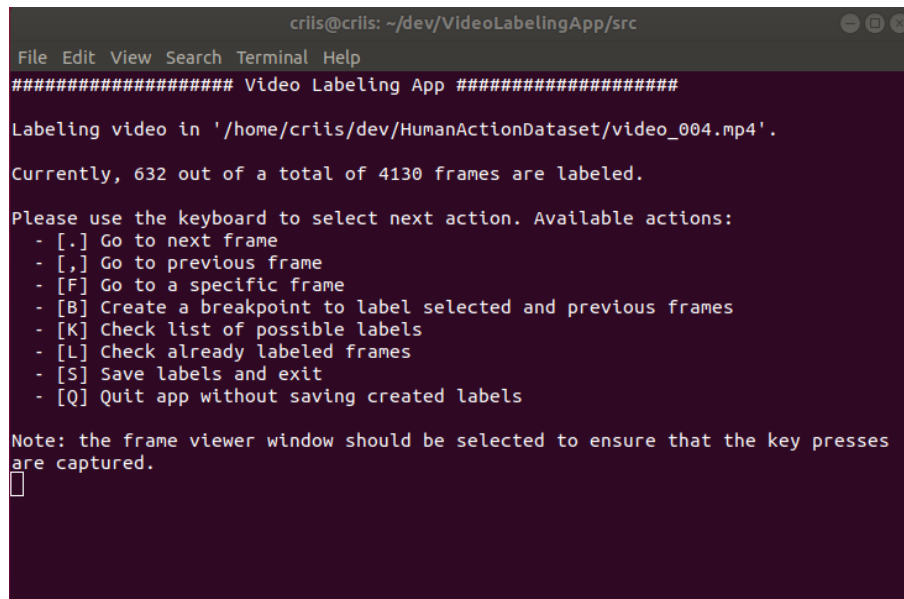
Figure A.2: Camera location in the assembly station.



## Appendix B

# Video Labeling Application

A video labeling *Python* application was developed to facilitate the labeling of all the video frames of the captured dataset. Figure B.1, which corresponds to the console window, showcases the main interface with the user. In turn, Figure B.2 corresponds to the frame viewer with overlaid real-time information about the frames and the selected labels.

A screenshot of a terminal window titled 'criis@criis: ~/dev/VideoLabelingApp/src'. The window displays the 'Video Labeling App' interface. It shows a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu bar, there is a separator line followed by '##### Video Labeling App #####'. The main text in the terminal reads: 'Labeling video in '/home/criis/dev/HumanActionDataset/video\_004.mp4''. Below this, it says 'Currently, 632 out of a total of 4130 frames are labeled.' Then, it prompts the user: 'Please use the keyboard to select next action. Available actions:'. A list of actions follows: '- [.] Go to next frame', '- [,] Go to previous frame', '- [F] Go to a specific frame', '- [B] Create a breakpoint to label selected and previous frames', '- [K] Check list of possible labels', '- [L] Check already labeled frames', '- [S] Save labels and exit', and '- [Q] Quit app without saving created labels'. At the bottom, a note states: 'Note: the frame viewer window should be selected to ensure that the key presses are captured.' followed by a small square cursor icon.

```
criis@criis: ~/dev/VideoLabelingApp/src
File Edit View Search Terminal Help
##### Video Labeling App #####
Labeling video in '/home/criis/dev/HumanActionDataset/video_004.mp4'.
Currently, 632 out of a total of 4130 frames are labeled.
Please use the keyboard to select next action. Available actions:
- [.] Go to next frame
- [,] Go to previous frame
- [F] Go to a specific frame
- [B] Create a breakpoint to label selected and previous frames
- [K] Check list of possible labels
- [L] Check already labeled frames
- [S] Save labels and exit
- [Q] Quit app without saving created labels
Note: the frame viewer window should be selected to ensure that the key presses
are captured.
█
```

Figure B.1: Console window. Corresponds to the main interface for the user interaction.

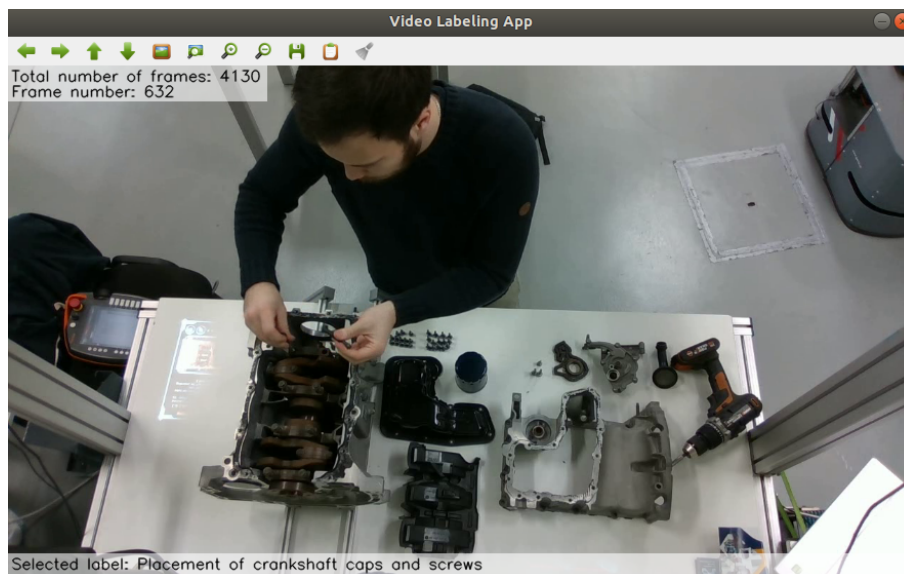


Figure B.2: Frame viewer window with real-time overlaid information about the performed annotations.

## Appendix C

# Training and validation details

This appendix displays the full training and validation results for the proposed model architectures. For each architecture, the results of both the baseline and the final model will be presented, showcasing a plot with the evolution of the training procedure alongside a table with the obtained validation results (including the results of the evaluation at the end of the training of each epoch from 16 to 20 and their average). Additionally, the confusion matrices for the evaluation of each epoch will be displayed.

### C.1 ResNet 3D 18

#### Baseline

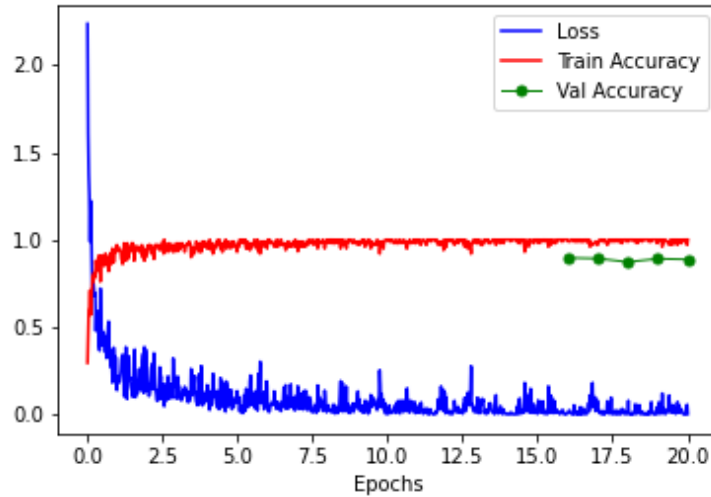
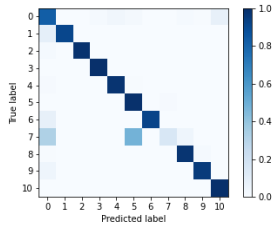


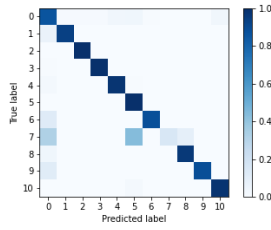
Figure C.1: Training evolution for the baseline model of the *ResNet 3D 18* architecture.

Table C.1: Results of the baseline model of the *ResNet 3D 18* architecture.

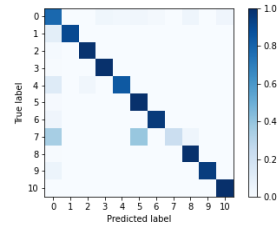
		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		89.64%	89.26%	87.27%	89.11%	88.75%	88.81%
<b>mIoU</b>		79.86%	80.18%	77.88%	79.14%	78.68%	79.15%
<b>IoU</b>	<b>0</b>	68.48%	68.07%	60.80%	67.00%	66.09%	66.09%
	<b>1</b>	90.61%	91.08%	89.36%	86.85%	87.41%	89.06%
	<b>2</b>	97.87%	96.99%	89.39%	97.87%	97.87%	96.00%
	<b>3</b>	94.68%	95.68%	86.34%	80.90%	94.68%	90.46%
	<b>4</b>	94.46%	93.81%	81.30%	95.11%	94.16%	91.77%
	<b>5</b>	74.44%	74.07%	76.60%	72.03%	67.48%	72.92%
	<b>6</b>	91.30%	87.11%	92.22%	95.57%	90.78%	91.40%
	<b>7</b>	15.14%	15.47%	22.10%	13.26%	15.14%	16.22%
	<b>8</b>	92.01%	90.59%	85.78%	91.42%	96.10%	91.18%
	<b>9</b>	93.52%	88.37%	94.01%	91.67%	92.77%	92.07%
	<b>10</b>	65.99%	80.80%	78.82%	78.82%	63.06%	73.50%



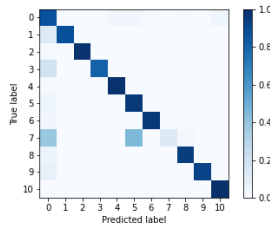
(a) Epoch 16



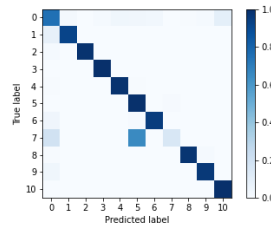
(b) Epoch 17



(c) Epoch 18



(d) Epoch 19



(e) Epoch 20

Figure C.2: Confusion matrices for the baseline model of the *ResNet 3D 18* architecture.

## Final model

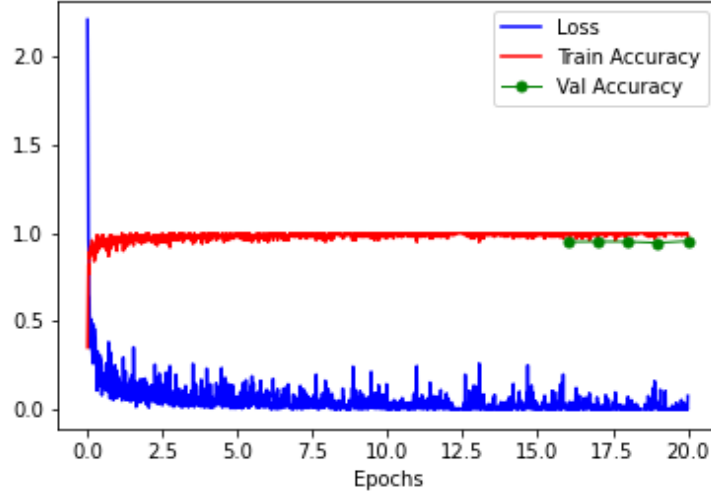


Figure C.3: Training evolution for the final model of the *ResNet 3D 18* architecture.

Table C.2: Results of the final model of the *ResNet 3D 18* architecture.

		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		95.11%	95.28%	95.22%	94.59%	95.55%	95.15%
<b>mIoU</b>		91.13%	91.71%	91.24%	89.90%	91.68%	91.13%
<b>IoU</b>	<b>0</b>	79.48%	79.35%	80.52%	77.64%	81.80%	79.76%
	<b>1</b>	93.28%	92.72%	93.16%	95.11%	93.26%	93.51%
	<b>2</b>	97.44%	98.82%	95.61%	97.88%	97.89%	97.53%
	<b>3</b>	92.71%	94.68%	95.17%	92.71%	95.58%	94.17%
	<b>4</b>	94.20%	93.02%	95.90%	94.78%	95.37%	94.65%
	<b>5</b>	90.91%	95.42%	94.08%	91.38%	91.38%	92.63%
	<b>6</b>	96.66%	98.66%	96.19%	96.19%	97.54%	97.05%
	<b>7</b>	86.41%	83.33%	86.41%	83.56%	86.56%	85.25%
	<b>8</b>	95.80%	95.11%	93.75%	95.82%	96.49%	95.39%
	<b>9</b>	96.69%	96.63%	93.67%	95.45%	96.92%	95.87%
	<b>10</b>	78.82%	81.07%	79.20%	68.37%	75.67%	76.63%

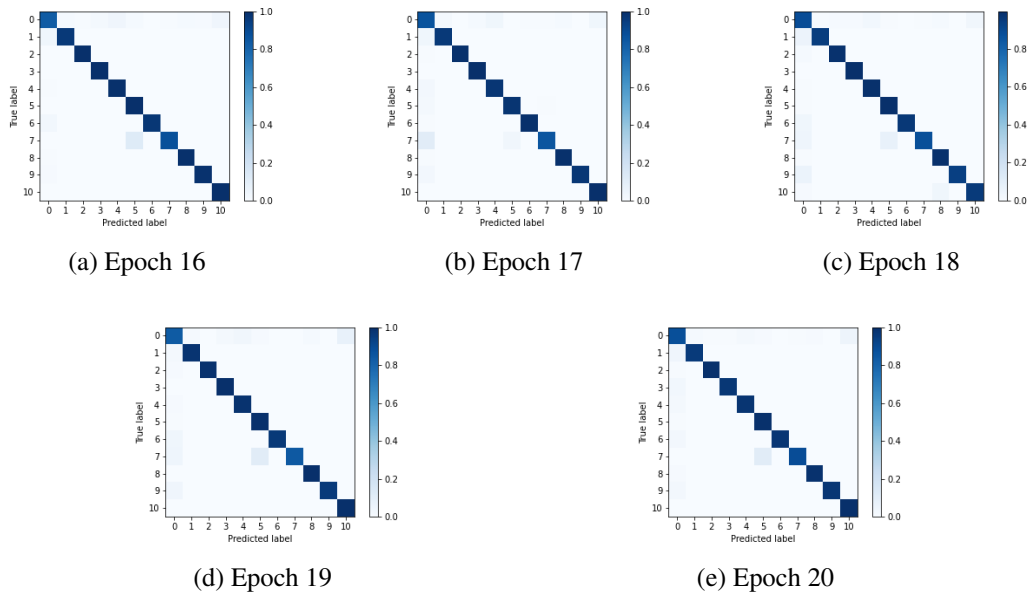


Figure C.4: Confusion matrices for the final model of the *ResNet 3D 18* architecture.

## C.2 ResNet-18 + LSTM

### Baseline

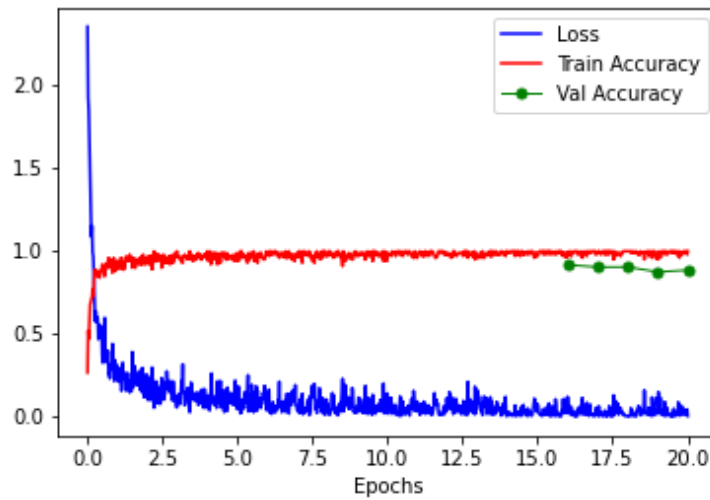
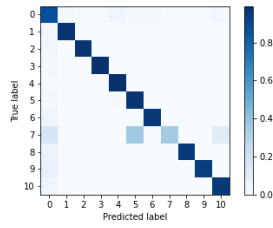


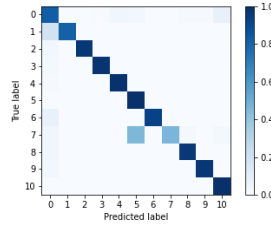
Figure C.5: Training evolution for the baseline model of the *ResNet-18 + LSTM* architecture.

Table C.3: Results of the baseline model of the *ResNet-18 + LSTM* architecture.

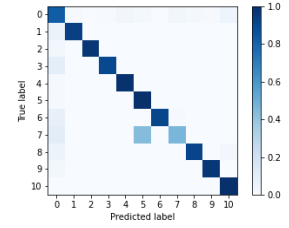
		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		91.78%	90.45%	90.51%	87.49%	88.62%	89.77%
<b>mIoU</b>		83.40%	82.21%	82.27%	78.72%	78.49%	81.02%
<b>IoU</b>	<b>0</b>	75.04%	69.75%	69.94%	59.07%	65.29%	67.82%
	<b>1</b>	93.62%	80.59%	91.08%	80.47%	88.69%	86.89%
	<b>2</b>	94.64%	96.44%	96.44%	97.87%	96.44%	96.37%
	<b>3</b>	95.10%	95.10%	88.95%	93.65%	88.95%	92.35%
	<b>4</b>	94.04%	94.65%	95.11%	94.65%	92.13%	94.12%
	<b>5</b>	78.91%	75.47%	77.34%	62.69%	70.18%	72.92%
	<b>6</b>	94.06%	92.41%	90.62%	92.91%	92.58%	92.52%
	<b>7</b>	35.36%	46.41%	40.19%	31.75%	30.46%	36.83%
	<b>8</b>	93.14%	92.59%	87.57%	84.68%	90.45%	89.69%
	<b>9</b>	92.55%	94.95%	94.95%	86.37%	91.82%	92.13%
	<b>10</b>	70.92%	65.99%	72.76%	81.85%	56.47%	69.60%



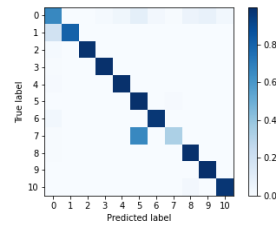
(a) Epoch 16



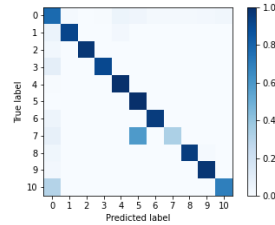
(b) Epoch 17



(c) Epoch 18



(d) Epoch 19



(e) Epoch 20

Figure C.6: Confusion matrices for the baseline model of the *ResNet-18 + LSTM* architecture.

## Final model

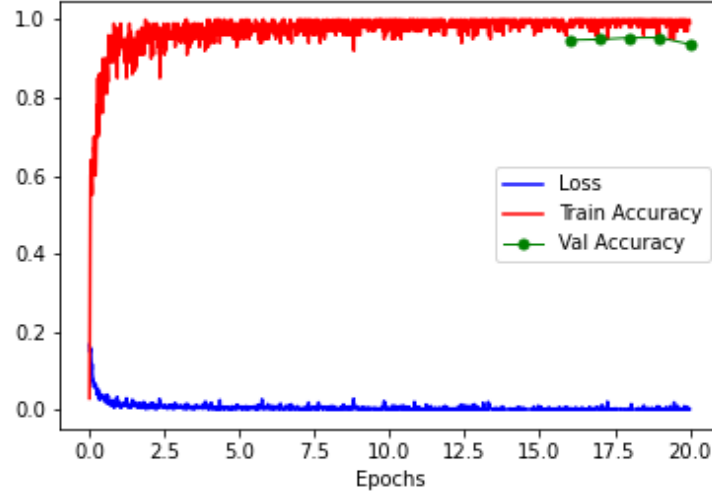


Figure C.7: Training evolution for the final model of the *ResNet-18 + LSTM* architecture.

Table C.4: Results of the final model of the *ResNet-18 + LSTM* architecture.

		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		94.78%	94.94%	95.36%	95.40%	93.68%	94.83%
<b>mIoU</b>		89.98%	90.43%	91.09%	91.30%	88.51%	90.26%
<b>IoU</b>	<b>0</b>	79.42%	79.26%	80.27%	80.05%	72.15%	78.23%
	<b>1</b>	95.02%	95.02%	95.67%	95.05%	93.15%	94.78%
	<b>2</b>	97.87%	98.35%	97.41%	97.87%	95.23%	97.35%
	<b>3</b>	94.65%	88.12%	94.65%	93.68%	91.75%	92.57%
	<b>4</b>	93.94%	94.34%	94.95%	95.12%	93.92%	94.45%
	<b>5</b>	88.63%	88.89%	93.83%	93.83%	88.56%	90.75%
	<b>6</b>	98.22%	97.37%	98.67%	97.33%	97.33%	97.78%
	<b>7</b>	80.27%	81.27%	86.34%	86.34%	84.42%	83.73%
	<b>8</b>	96.16%	97.59%	96.13%	97.59%	92.48%	95.99%
	<b>9</b>	97.23%	96.43%	94.02%	97.21%	97.22%	96.42%
	<b>10</b>	68.37%	78.05%	70.03%	70.22%	67.37%	70.81%



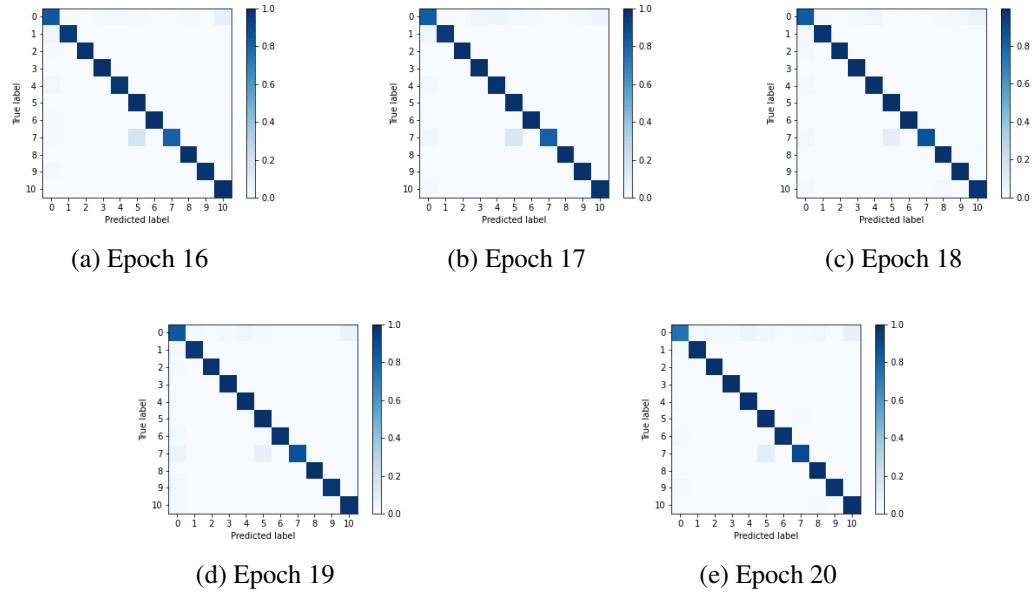


Figure C.8: Confusion matrices for the final model of the *ResNet-18 + LSTM* architecture.

### C.3 ResNet-34 + LSTM

#### Baseline

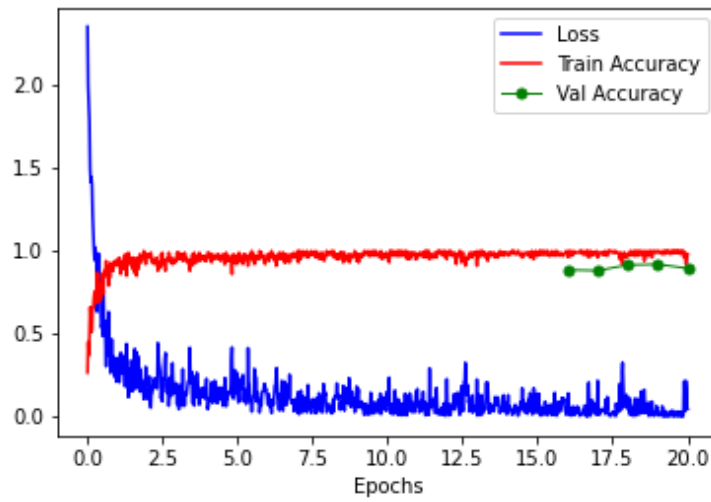
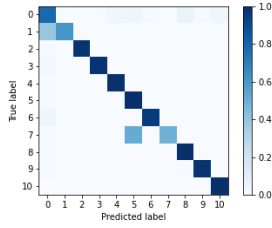


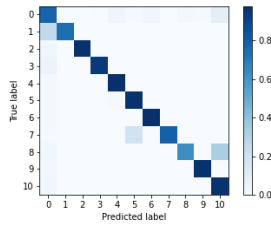
Figure C.9: Training evolution for the baseline model of the *ResNet-34 + LSTM* architecture.

Table C.5: Results of the baseline model of the *ResNet-34 + LSTM* architecture.

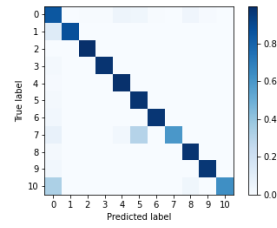
		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		88.58%	88.16%	91.58%	91.94%	89.54%	89.96%
<b>mIoU</b>		80.48%	79.38%	84.01%	86.23%	80.04%	82.03%
<b>IoU</b>	<b>0</b>	63.60%	64.92%	71.56%	69.83%	66.06%	67.19%
	<b>1</b>	60.57%	74.17%	86.85%	88.11%	90.73%	80.09%
	<b>2</b>	97.87%	96.44%	98.35%	98.35%	97.87%	97.78%
	<b>3</b>	95.10%	91.78%	95.10%	92.92%	93.65%	93.71%
	<b>4</b>	94.65%	93.06%	92.85%	92.85%	94.65%	93.61%
	<b>5</b>	71.75%	85.89%	77.89%	82.83%	70.48%	77.77%
	<b>6</b>	93.21%	92.77%	96.44%	93.12%	96.45%	94.40%
	<b>7</b>	48.62%	79.56%	59.67%	75.14%	31.75%	58.95%
	<b>8</b>	83.60%	60.18%	87.06%	84.68%	85.78%	80.26%
	<b>9</b>	95.29%	94.14%	94.95%	94.04%	93.87%	94.46%
	<b>10</b>	81.07%	40.26%	63.44%	76.62%	59.11%	64.10%



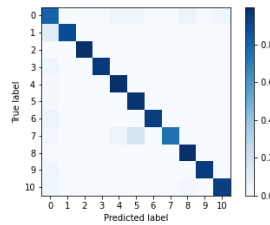
(a) Epoch 16



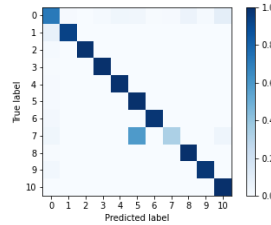
(b) Epoch 17



(c) Epoch 18



(d) Epoch 19



(e) Epoch 20

Figure C.10: Confusion matrices for the baseline model of the *ResNet-34 + LSTM* architecture.

### Final model

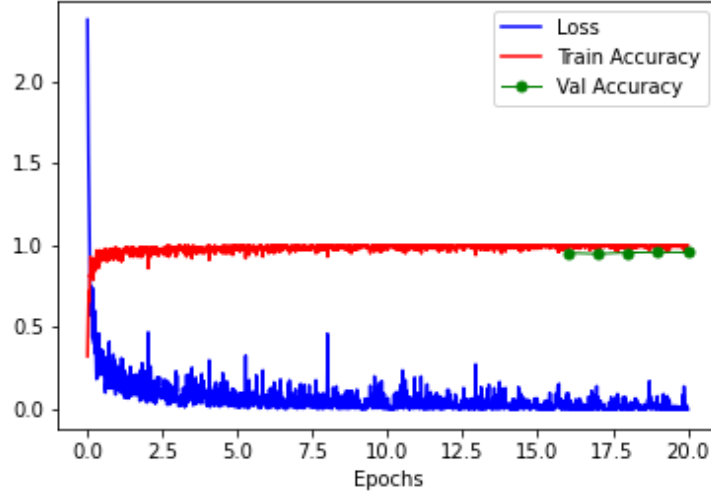


Figure C.11: Training evolution for the final model of the *ResNet-34 + LSTM* architecture.

Table C.6: Results of the final model of the *ResNet-18 + LSTM* architecture.

		Epoch number					Average
		16	17	18	19	20	
<b>Accuracy</b>		95.36%	94.82%	95.55%	95.82%	95.68%	95.45%
<b>mIoU</b>		91.49%	90.35%	91.87%	92.61%	92.10%	91.68%
<b>IoU</b>	<b>0</b>	81.63%	81.35%	80.97%	81.26%	81.79%	81.40%
	<b>1</b>	92.67%	93.18%	91.63%	93.46%	94.50%	93.09%
	<b>2</b>	97.90%	98.35%	97.88%	98.35%	96.50%	97.80%
	<b>3</b>	96.19%	96.16%	94.68%	92.71%	95.65%	95.08%
	<b>4</b>	94.64%	93.15%	95.58%	94.34%	94.97%	94.54%
	<b>5</b>	88.87%	86.96%	92.49%	95.20%	91.43%	90.99%
	<b>6</b>	97.80%	97.78%	97.80%	98.23%	96.92%	97.71%
	<b>7</b>	79.50%	65.84%	84.24%	88.59%	86.41%	80.92%
	<b>8</b>	95.43%	96.03%	95.46%	97.59%	96.87%	96.28%
	<b>9</b>	97.21%	96.62%	96.38%	95.83%	96.93%	96.59%
	<b>10</b>	84.59%	88.40%	83.46%	83.21%	81.07%	84.15%

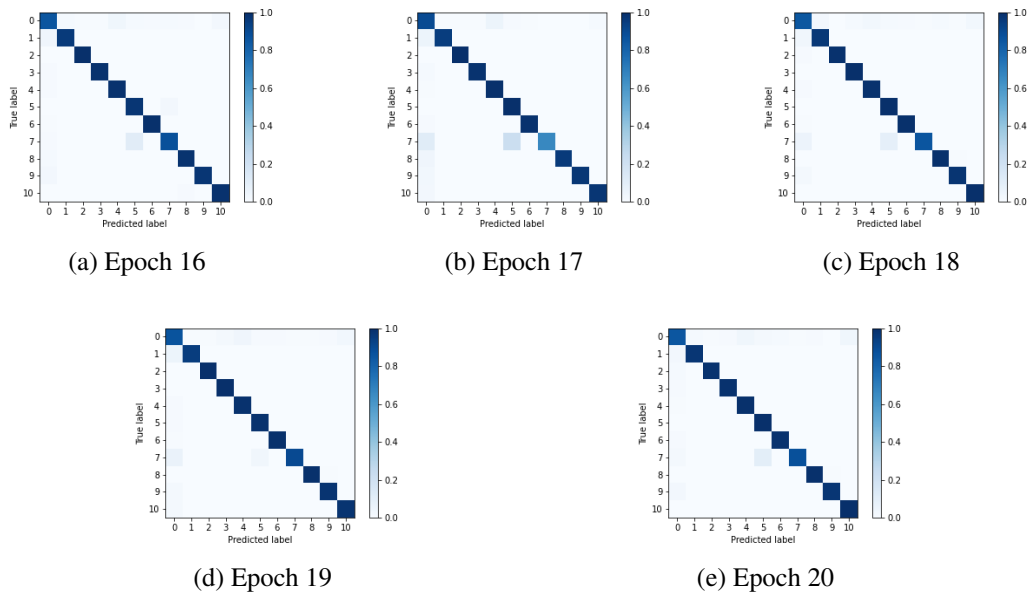


Figure C.12: Confusion matrices for the final model of the *ResNet-34 + LSTM* architecture.

# References

- [1] Henny Admoni and Brian Scassellati. Social Eye Gaze in Human-Robot Interaction: A Review. *Journal of Human-Robot Interaction*, 6(1):25, 2017.
- [2] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kotsuge, and Oussama Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, 42(5):957–975, 2018.
- [3] Fatemeh Mohammadi Amin, Maryam Rezayati, Hans Wernher van de Venn, and Hossein Karimpour. A mixed-perception approach for safe human–robot collaboration in industrial automation. *Sensors (Switzerland)*, 20(21):1–20, 2020.
- [4] Tadas Baltrusaitis, Peter Robinson, and Louis Philippe Morency. OpenFace: An open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016.
- [5] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis Philippe Morency. OpenFace 2.0: Facial behavior analysis toolkit. In *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, pages 59–66, 2018.
- [6] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human-robot collaboration: A survey. *International Journal of Humanoid Robotics*, 5(1):47–66, 2008.
- [7] Maija Breque, Lars De Nul, and Athanosios Petrides. Industry 5.0 - Towards a sustainable, human- centric and resilient European industry. *European Commission*, page 48, 2021.
- [8] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.
- [9] Zhe Cao, Tomas Simon, Shih En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 1302–1310, 2017.
- [10] Ravi Teja Chadalavada, Henrik Andreasson, M. Schindler, Rainer Palm, and Achim J. Lilienthal. Bi-directional navigation intent communication using spatial augmented reality and eye-tracking glasses for improved safety in human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 61, 2020.
- [11] Balasubramaniyan Chandrasekaran and James M. Conrad. Human-robot collaboration: A survey. In *Conference Proceedings - IEEE SOUTHEASTCON*, 2015.

- [12] Carlos M. Costal, Germano Veiga, Armando Sousa, Luís Rocha, A. Augusto Sousa, Rui Rodrigues, and Ulrike Thomas. Modeling of video projectors in OpenGL for implementing a spatial augmented reality teaching system for assembly operations. In *19th IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2019*, 2019.
- [13] Jaime Duque-Domingo, Jaime Gómez-García-Bermejo, and Eduardo Zalama. Gaze Control of a Robotic Head for Realistic Interaction With Humans. *Frontiers in Neurorobotics*, 14, 2020.
- [14] Tobias Fischer, Hyung Jin Chang, and Yiannis Demiris. RT-GENE: Real-time eye gaze estimation in natural environments. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11214 LNCS, pages 339–357, 2018.
- [15] Jing Guo, Yi Liu, Qing Qiu, Jie Huang, Chao Liu, Zhiguang Cao, and Yue Chen. A Novel Robotic Guidance System with Eye-Gaze Tracking Control for Needle-Based Interventions. *IEEE Transactions on Cognitive and Developmental Systems*, 13(1):179–188, 2021.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, 2016.
- [17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.
- [18] Chien-Ming Huang, Sean Andrist, Allison Sauppé, and Bilge Mutlu. Using gaze patterns to predict task intent in collaboration. *Frontiers in Psychology*, 6, 2015.
- [19] Chien Ming Huang and Bilge Mutlu. Anticipatory robot control for efficient human-robot collaboration. In *ACM/IEEE International Conference on Human-Robot Interaction*, volume 2016-April, pages 83–90, 2016.
- [20] ISO/TS 15066:2016. Robots and robotic devices — Collaborative robots. Standard/technical specification, International Organization for Standardization, 2016.
- [21] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [22] Maryam Koohzadi and Nasrollah Moghadam Charkari. Survey on deep learning methods in human action recognition. *IET Computer Vision*, 11(8):623–632, 2017.
- [23] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. PifPaf: Composite fields for human pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 11969–11978, 2019.
- [24] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2021.
- [25] Junwoo Lee and Bummo Ahn. Real-time human action recognition with a low-cost RGB camera and mobile robot platform. *Sensors (Switzerland)*, 20(10), 2020.

- [26] Peng Li, Xuebin Hou, Le Wei, Guoli Song, and Xingguang Duan. Efficient and low-cost deep-learning based gaze estimator for surgical robot control. In *2018 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2018*, pages 58–63, 2019.
- [27] Songpo Li, Xiaoli Zhang, Fernando J. Kim, Rodrigo Donalisio da Silva, Diedra Gustafson, and Wilson R. Molina. Attention-aware robotic laparoscope based on fuzzy interpretation of eye-gaze patterns. *Journal of Medical Devices, Transactions of the ASME*, 9(4), 2015.
- [28] Zitong Liu, Quan Liu, Wenjun Xu, Zhihao Liu, Zude Zhou, and Jie Chen. Deep learning-based human motion prediction considering context awareness for human-robot collaboration in manufacturing. In *Procedia CIRP*, volume 83, pages 272–278, 2019.
- [29] Oliver Lorenz and Ulrike Thomas. Real Time Eye Gaze Tracking System using CNN-based Facial Features for Human Attention Measurement. In *VISIGRAPP 2019 - Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 598–606, 2019.
- [30] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A Simple Yet Effective Baseline for 3d Human Pose Estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2659–2668, 2017.
- [31] Oskar Palinko, Francesco Rea, Giulio Sandini, and Alessandra Sciutti. A Robot reading human gaze: Why eye tracking is better than head tracking for human-robot collaboration. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2016-Novem, pages 5048–5054, 2016.
- [32] Matteo Parigi Polverini, Andrea Maria Zanchettin, and Paolo Rocco. A constraint-based programming approach for robotic assembly skills implementation. *Robotics and Computer-Integrated Manufacturing*, 59:69–81, oct 2019.
- [33] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 7745–7754, 2019.
- [34] Akanksha Saran, Srinjoy Majumdar, Elaine Schaertl Shor, Andrea Thomaz, and Scott Niekum. Human Gaze Following for Human-Robot Interaction. In *IEEE International Conference on Intelligent Robots and Systems*, pages 8615–8621, 2018.
- [35] Lei Shi, Cosmin Copot, and Steve Vanlanduit. What are you looking at? detecting human intention in gaze based human-robot interaction. *arXiv preprint arXiv:1909.07953*, 2019.
- [36] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset. *arXiv preprint arXiv:2010.10864*, 2020.
- [37] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 4263–4270, 2017.
- [38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann Lecun, and Manohar Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

- [39] Ales Vysocky and Petr Novak. Human - Robot collaboration in industry. *MM Science Journal*, 2016-June:903–906, 2016.
- [40] Peng Wang, Hongyi Liu, Lihui Wang, and Robert X. Gao. Deep learning-based human motion recognition for predictive context-aware human-robot collaboration. *CIRP Annals*, 67(1):17–20, 2018.
- [41] Jianjing Zhang, Hongyi Liu, Qing Chang, Lihui Wang, and Robert X. Gao. Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly. *CIRP Annals*, 69(1):9–12, 2020.