FEDERAL UNIVERSITY OF OURO PRETO
INSTITUTE OF EXACT AND BIOLOGICAL SCIENCES
COMPUTING DEPARTMENT

JOÃO GABRIEL FERNANDES ZENÓBIO
Supervisor: Prof. Ph.D. Jadson Castro Gertrudes
Co-supervisor: M.Sc. Sakif Hossain

SHORT-TERM EARLY ACTION PREDICTION FOR HUMAN-ROBOT COLLABORATION: A DEEP LEARNING ATTENTION-BASED APPROACH

Ouro Preto, MG
2025

FEDERAL UNIVERSITY OF OURO PRETO
INSTITUTE OF EXACT AND BIOLOGICAL SCIENCES
COMPUTING DEPARTMENT

JOÃO GABRIEL FERNANDES ZENÓBIO

SHORT-TERM EARLY ACTION PREDICTION FOR HUMAN-ROBOT COLLABORATION: A DEEP LEARNING ATTENTION-BASED APPROACH

Monograph presented in the Computer Science course of the Federal University of Ouro Preto in partial fulfillment of the requirements necessary to obtain Computer Science Bachelor degree.

Supervisor: Prof. Ph.D. Jadson Castro Gertrudes
Co-supervisor: M.Sc. Sakif Hossain

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

João Gabriel Fernandes Zenóbio

Short-term Early Action Prediction for Human-Robot Collaboration: A Deep Learning Attention Based Approach

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 2 de Abril de 2025.

Membros da banca

Jadson Castro Gertrudes (Orientador) - Doutor - Universidade Federal de Ouro Preto
Sakif Houssain (Coorientador) - Mestre - Technische Universität Clausthal
Jörg Philipp Müller (Examinador) - Dr - Technische Universität Clausthal
Hugo Eduardo Ziviani (Examinador) - Mestre - Programa de Pós Graduação em Ciência da Computação (UFOP)

Jadson Castro Gertrudes, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 2/04/2025.

Documento assinado eletronicamente por **Jadson Castro Gertrudes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 04/04/2025, às 09:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0886263** e o código CRC **981ACA44**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

To all the people that come to mind when I ask myself what gives life meaning.
To all that taught me kindness.
To my 知音.

Acknowledgements

I thank the Federal University of Ouro Preto, which gave me the life experience and support I needed during college.

I thank my supervisors from Brazil, Jadson and Valéria, who guided and inspired me throughout this long journey. It is infeasible to measure how much I have learned from you and all these years at the university were only meaningful because of you.

I thank my supervisors from Germany, Jörg and Sakif, who inspired me to see in myself someone greater than I could imagine I could be. I am immensely grateful for the opportunity you gave me.

I thank my family for giving me warmth.

I thank my friends for teaching me about joy.

I thank my 知音 for listening to my music.

“I’m not gonna run away, and I never go back on my word! That’s my nindoo, my ninja way!” - Naruto Uzumaki (KISHIMOTO, 1999)

Abstract

The collaboration between humans and robots is no longer a movie scenario. Due to the increase in the number of work environments in which this type of interaction takes place, robots must be able, just like humans, to make predictions about their partners so that they can work together. This study systematically searched the literature for early action prediction models based on attention mechanisms, offering a broad overview of the state of the art. Only one suitable model with such characteristics was found, TemPr, which represents the state of the art in the area of early action prediction. It also proposed a methodology for analyzing the model found for the prior detection of actions in the context of human-robot collaboration in industrial environments to contribute to this area of research. The InHARD dataset is used to train the models, Optuna is used to automatically tune hyperparameters for the model, and MLflow is used to track hyperparameters and analyze the results of different models. The TemPr model is extensively analyzed for different video observation ratios. The experiments are run on a machine with an available GPU within a Docker container with a base image available on the Nvidia GPU Container. The final model achieved an accuracy of 59% and 55%, and an OvR macro-average ROC AUC score of 93% and 92%, for an observation ratio of 1.0 and 0.3, respectively. Thus, the network training was successful and had great results even when only a small part of the action video was available.

Keywords: Human-robot collaboration. Early action prediction. Short-term action. Transformers. Attention. Industrial environments.

Resumo

PREVISÃO ANTECIPADA DE AÇÕES CURTAS PARA A COLABORAÇÃO ENTRE HUMANOS E ROBÔS: UMA ABORDAGEM BASEADA EM ATENÇÃO E APRENDIZADO PROFUNDO

A colaboração entre humanos e robôs não é mais um cenário de filme. Devido ao aumento do número de ambientes de trabalho em que esse tipo de interação ocorre, os robôs devem ser capazes, assim como os humanos, de fazer previsões sobre seus parceiros para que possam trabalhar juntos. Este estudo pesquisou sistematicamente na literatura os modelos de previsão de ação antecipada baseados em mecanismos de atenção, oferecendo uma ampla visão geral do estado da arte. Foi encontrado apenas um modelo adequado com essas características, o TemPr, que representa o estado da arte na área de previsão de ação antecipada. Também foi proposta uma metodologia para analisar o modelo encontrado para a detecção prévia de ações no contexto da colaboração entre humanos e robôs em ambientes industriais para contribuir com essa área de pesquisa. O conjunto de dados InHARD é usado para treinar os modelos, o Optuna é usado para ajustar automaticamente os hiperparâmetros do modelo e o MLflow é usado para rastrear os hiperparâmetros e analisar os resultados de diferentes modelos. O modelo TemPr é amplamente analisado para diferentes proporções de observação de vídeo. Os experimentos são executados em uma máquina com uma GPU disponível em um contêiner Docker com uma imagem de base disponível no Nvidia GPU Container. O modelo final alcançou uma precisão de 59% e 55%, e uma pontuação ROC AUC macro-média OvR de 93% e 92%, para uma taxa de observação de 1,0 e 0,3, respectivamente. Assim, o treinamento da rede foi bem-sucedido e obteve ótimos resultados mesmo quando apenas uma pequena parte da ação.

Palavras-chave: Colaboração humano-robô. Previsão antecipada de ações. Ação de curta duração. Transformadores. Atenção. Ambientes industriais.

List of Figures

List of Tables

List of Abbreviations and Acronyms

RGB	Red Green Blue
EAP	Early Action Prediction
IP	Instantly Predictable
LP	Late Predictable
ANN	Artificial Neural Network
FC	Fully Convolutional
MLP	Multi-layer Perceptron
CNN	Convolutional Neural Network
NLP	Natural Language Processing
RNN	Recurrent Neural Network
GPU	Graphic Processing Unit
ViT	Vision Transformer
ViViT	Video Vision Transformer
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
MAS	Multi-Agent System
BDI	Beliefs, Desires, Intentions
NGC	Nvidia GPU Container
InHARD	Industrial Human Action Recognition Dataset
FN	False negative
FP	False Positive
TP	True positive
ROC	Receiver Operating Characteristic

AUC	Area Under the Curve
OvR	One over Rest
SGD	Stochastic Gradient Descent

List of Symbols

ρ	Greek letter rho
σ	Greek letter sigma
δ	Greek letter delta
ϕ	Greek letter phi
α	Greek letter alpha
$\mathcal{D}$	Database
$\mathcal{T}$	Attention tower
$\mathcal{E}$	Greek letter epsilon
β	Greek letter beta

Contents

1 Introduction

Although science fiction and cyberpunk stories have not yet become reality, humans and robots are not staying so far apart from each other. The number of places where they share space is increasing, and Human-Robot Interaction (HRI) has become an important field of robotics (ROBINSON et al., 2023). HRI comes from the field of Human-Computer Interaction, but it extends beyond simple teleoperation of a remote platform, enabling the robot to perform a range of autonomous behaviors (SCHOLTZ, 2003).

Interactions can be categorized into three types: instruction, cooperation, and collaboration (a visualization can be observed in Figure 1.1 (BÜTEPAGE; KRAGIC, 2017)). Instruction-based interaction represents a master-slave dynamic, where robots solely respond to human directives. Cooperation involves agents acting independently toward a shared goal, with only the outcomes of their actions being shared in the environment. Collaboration, however, necessitates that agents engage in interdependent actions (Figure 1.2).

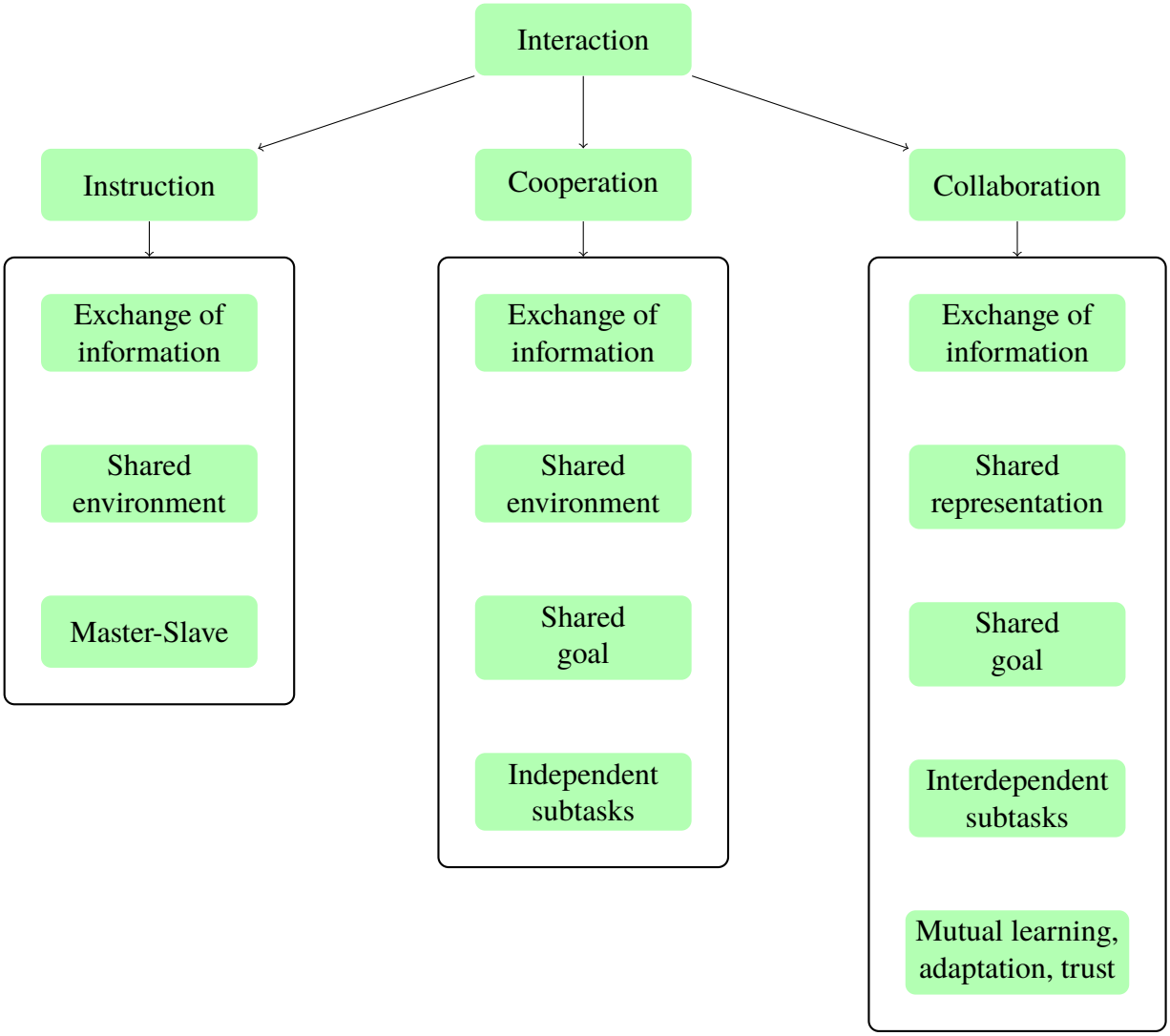
Scholtz (2003), when defining the theory of HRI, assumes that "the robot is already programmed to carry out basic functions and any 'reprogramming' happens during the intervention." This means that the robot or the user's learning ability is not considered. However, "collaboration is the basis for mutual learning and mutual adaptation and requires mutual trust" (BÜTEPAGE; KRAGIC, 2017). Thus, when this kind of action is considered, the context is defined as a Human-Robot Collaboration (HRC).

Human-robot collaboration (HRC) is challenging due to the uncertainty and constraints humans introduce, which are not foreseeable beforehand. Moreover, it requires that the robot act naturally. This behavior results in more efficient interactions between humans and robots and reduces the workload for humans. Although some tasks do not require sharing in all dimensions, humans commonly solve tasks collaboratively, even when unnecessary. Such instinct in humans shows the importance of giving robots the same skill (BÜTEPAGE; KRAGIC, 2017). Nevertheless, HRC is a relatively new field, and little work has been done in this study area.

In these multi-agent systems (MAS), beliefs, desires, and intentions (BDI) of the agents (BRATMAN, 1987) are modeled toward common goals, and collaborative agents share them. To act together with humans, robots must construct intelligent beliefs about their partner BDI sets to build appropriate shared intentions and act towards fulfilling a shared desire. Therefore, thoughtful approaches to dealing with other agents' possible state information are necessary.

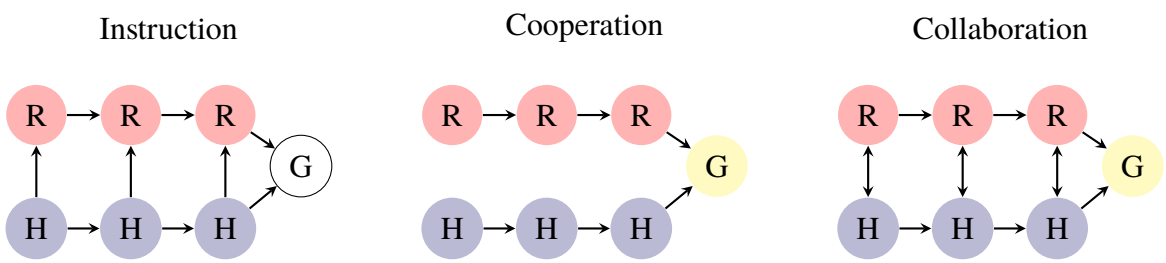
One possible approach is to use computer vision. This area of research deals with how to make machines leverage visual information from sensors that capture spatial information from

Figure 1.1 – Three types of interaction between humans and robots, increasing in the amount of interdependency



Source: Adapted from Bütepage e Kragic (2017).

Figure 1.2 – The interaction between the robot (R, red) and human (H, blue) to reach the human's (G, white) or shared (G, yellow) goal over time. Arrows indicate dependencies.



Source: Adapted from Bütepage e Kragic (2017)

visual sensors, such as cameras, LiDARs, etc. More specifically, robotic vision studies how computer vision is used in contexts involving robots.

Several works proposed solutions to recognize human activity in videos that have achieved high accuracy in constructing such beliefs from the human partner. However, "as soon as predictions can be made confidently, the robot can change from reactive to active behavior, allowing for collaborative actions" (BÜTEPAGE; KRAGIC, 2017). Hence, giving robots predictive power instead is ultimately crucial in HRC contexts.

One possible way to make early predictions from humans is to observe their actions. Early Action Prediction (EAP) studies how to perform such a task. Its objective is to make action predictions as soon as possible. Thus, it becomes a complex analysis due to the lack of information available about the target class to be predicted.

To contribute and incentivize more research in HRC, this monograph explores state-of-the-art Deep Learning models to accomplish EAP for robotic vision. EAP is a relatively new field, so to tackle this problem, attention mechanisms are used as the most recent approaches for video analysis.

Following the work of Vaswani et al. (2017) on attention mechanisms, these mechanisms gained widespread popularity among deep learning researchers globally. Their work introduced a novel model known as the Transformer, which is constructed using distinct artificial network layers—specifically, multi-head self-attention layers. This architecture has been successfully applied across nearly all subfields of Deep Learning. Consequently, we leverage this contemporary approach to contribute to the field of Human-Robot Collaboration (HRC).

1.1 Justification

Robots and humans are increasingly collaborating in real-world environments, especially in industrial settings. The interactions that happen inside the industries shape a hazardous environment where most of the machines move independently, even when humans are near them. New modern approaches to dealing with such interactions are necessary to provide safer and more efficient collaboration.

Therefore, this monograph aims to better HRC using modern robotic vision for protecting humans from harm and improving work productivity. Furthermore, this project contributes to developing the relatively new EAP field by analyzing attention mechanisms and building a new solution inside the HRC context.

1.2 Objectives

1.2.1 Main objectives

The core objectives of this research are multilayered:

1. Identification of various early action prediction models.
2. Analysis of prevalent early action prediction models.

1.2.2 Specific objectives

The specific objectives of this research are multilayered:

1. Systematically uncover a range of early action prediction models from existing literature.
2. Analyse the uncovered models, offering a broad perspective.
3. Select the state-of-the-art attention-based models from the analysis.
4. Evaluate the selected models regarding suitability for a human-robot collaboration setting.
5. Analyze the results of the evaluated models.

1.3 Structure of the Monograph

Chapter 1 provides the Introduction. Chapter 2 covers the Literature Review, including the main theoretical concepts, related work, and final considerations. Chapter 3 focuses on the Development of the monograph, detailing the methodology and experiments. Chapter 4 presents the Results of the experiments, showcasing the outcomes obtained. Finally, Chapter 5 offers the Final Considerations, summarizing the conclusion and suggesting future work.

2 Literature Review

This chapter aims to contextualize the reader about the current state of the art in early action prediction using deep learning approaches, as well as related work in the context of human-robot collaboration environments utilizing RGB second- and third-view video data.

2.1 Theoretical Foundations

2.1.1 Action

In computer vision research, an action refers to any dynamic process composed of observed body movements, typically conveyed in a video lasting a few seconds. Defining which movements are performed by specific limbs and how they compose a particular type of action is a complex task. Due to the challenges in representing and classifying this concept, formulating a formal definition for it remains difficult (KON; FU, 2022).

To simplify this task, the computer vision community typically limits the scope of their work by observing specific contexts in which the environment is set, and the agents' goals are known. Examples of contexts are sports (KARPATHY et al., 2014), human-object daily interactions (GOYAL et al., 2017), human-human daily interactions (RYOO; AGGARWAL, 2010), kitchen tasks (DAMEN et al., 2018), etc. Elected the scope, several actions now become feasible to be formally defined. However, they cannot be typified outside these scenarios.

2.1.1.1 Action recognition

Intelligent systems that perform action recognition classify actions based on complete executions. Inside the computer vision area of research, the task consists of generating an action representation by converting the video into feature vectors and inferring a class from these features (KON; FU, 2022). The state of the art of this topic mainly embraces video foundation models (WANG et al., 2023b; WANG et al., 2024).

2.1.1.2 Action prediction

An intelligent system capable of prompt reactions to ongoing situations performs better when dealing with time-sensitive tasks, such as autonomous driving (GIRASE et al., 2021), human-robot interactions (RYOO et al., 2015), etc. This system must recognize agents' actions as soon as possible to perform prompt reactions, turning prediction into a necessary skill. Unlike recognition, prediction relies on incomplete data to reason about the future and make educated

guesses about upcoming events. In computer vision, predicting an action concerns inferring a label to an action captured in a video before its execution ends (KONG; TAO; FU, 2017; STERGIOU; DAMEN, 2023; KON; FU, 2022).

2.1.1.3 Long-term action prediction

Long-term action prediction infers a future set of actions based on the current observed set of actions. Formally, given one or more independent, semantically meaningful, and temporally correlated sets of actions $\mathbf{X} = \{x_1, x_2, ..., x_n\}$, the goal is to predict the next set of actions $\mathbf{Y} = \{y_1, y_2, ..., y_n\}$. Long-duration videos are more suitable for this purpose, generally lasting several minutes (KON; FU, 2022).

2.1.1.4 Short-term action prediction

Short-term prediction focuses on inferring labels for a single action in a temporally incomplete video. Formally, the goal is to infer the label $y : \mathrm{x} \mapsto y$ given an incomplete video x and the complete action execution $\mathrm{X} = \{f_1, f_2, ..., f_T\}$, which only contains a single action and is composed of the frames f_i. The goal is constrained by the fact that $\mathrm{x} \cup \mathrm{X}$ and $|\mathrm{x}| < T$. Generally, the action videos used in this case are short, lasting several seconds (KON; FU, 2022).

2.1.1.5 Early action prediction

Early action prediction (EAP) is a special case of short-term action prediction, which aims to classify an action at the very beginning of its full execution (e.g., 10% (KONG; TAO; FU, 2017), or 20% (WANG et al., 2023c)). The amount of observed frames is defined as $T_\rho = \lceil \rho T \rceil$ para $0 < \rho < 1$, where ρ, the observation ratio, dictates the percentage of the total amount of frames in the video, T, taken in account to make the final labeling decision (STERGIOU; DAMEN, 2023).

The difficulty of this task is dictated not only by the low percentage of information the system is allowed to observe but also by each action prediction characteristic. Kong, Tao e Fu (2017) grouped and sorted actions according to the portion of a video needed to be observed before separating the action correctly, and three different categories were defined: *instantly predictable* (IP), *early predictable* (EP) and *late predictable* (LP) (Table 2.1). The observation ratio ρ necessary for the classification of the action grows, respectively, within these categories. Thus, the distribution of their discriminative patterns over time varies for different actions, independently of the environment and the agent. Among the EAP cases, late predictable actions are more challenging to classify.

Table 2.1 – Top 10 instantly, early, and late predictable actions in UCF101 dataset. Action names are sorted according to the percentage of testing samples falling in the IP, EP, or LP category.

Instantly Predictable	**Early Predictable**	**Late Predictable**
Billiards	Fencing	Javelin Throw
Ice Dancing	Frisbee Catch	High Jump
Rock Climbing Indoor	Soccer Penalty	Front Crawl
Playing Piano	Volleyball Spiking	Head Massage
Pommel Horse	Hula Hoop	Haircut
Rowing	Field Hockey Penalty	Playing Violin
Ski jet	Basketball Dunk	Handstand Walking
Juggling Balls	Cliff Diving	Pole Vault
Soccer Juggling	Bowling	Cricket Bowling
Tai Chi	Tennis Swing	Throw Discus

Source: Adapted from Kong, Tao e Fu (2017).

2.1.2 Deep Learning

Deep Learning is a branch of Machine Learning that develops techniques designed with many interconnected layers of algebraic circuits with tunable connection strengths, called Artificial Neural Networks. This area of research originates in the work of McCulloch e Pitts (1943) that tried to model the brain's neurons with computational nets. These structures have long computation paths, allowing the input variables to interact in complex ways and capture the intricacy of real-world data (RUSSELL; NORVIG, 2022). The works of Cybenko (1989) and Micchelli (1986) even suggest that with a single hidden layer network, modeled with enough neurons and the proper weights, can approximate any function.

2.1.2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are the main structure of Deep Learning algorithms. They comprise a series of connected layers that perform computations over input data X and carry the results, called activations (H), to the neighbor layer until it reaches the output O. When the network has several layers apart from the input, these are called hidden layers (ZHANG et al., 2023; RUSSELL; NORVIG, 2022).

These layers have units, called neurons, that map the input values to a function composed of a linear and a non-linear part. The linear part (f) holds tunable parameters, called weights and bias (W and b), that control its behavior. However, a sequence of computation over linear functions would have the same effect as a single linear computation, independent of the number of hidden layers. The non-linear part (σ) permits neural networks to work with non-linear data but also allows them to leverage deep architectures with several hidden layers and learn complex patterns. It maps the result of the latter to a non-linear function called the activation function,

which returns the activations. Several activation functions have been proposed in the literature, such as Sigmoid, ReLU (NAIR; HINTON, 2010), Hyperbolic Tangent, etc (ZHANG et al., 2023; RUSSELL; NORVIG, 2022).

ANNs intend to optimize their tunable parameters for a given objective function called the loss function (l). The loss calculates the error distance (L) between the current value returned from the network and a desired reference (y) given input data. Therefore, the network is optimized to minimize the loss function while learning from examples (ZHANG et al., 2023; RUSSELL; NORVIG, 2022).

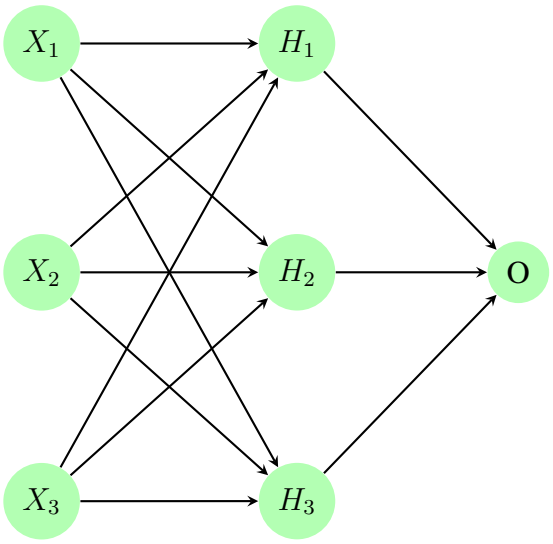
After the forward propagation - when all layers have performed their calculations sequentially - a backward propagation algorithm is applied to the network to adapt it. The backward process uses the negative gradients of the loss ∇L regarding each tunable parameter given input data to minimize the objective function. The gradients are determined using the chain rule from differential calculus and dynamic programming to avoid recalculating the chain parts that repeat over the calculations of each layer's gradients (ZHANG et al., 2023).

This adaptation follows an optimization algorithm, such as SGD (ZHANG et al., 2023), Adagrad (DUCHI; HAZAN; SINGER, 2011), RMSprop (ZHANG et al., 2023), Adam (KINGMA; BA, 2017), among others (ZHANG et al., 2023; RUSSELL; NORVIG, 2022). Moreover, the optimization functions use the learning rate η to weigh the amount of change this adaptation should generate in the network.

A common ANN architecture solution to connect layers is to send all the activations to all neurons of the neighboring layer. This type of layer is called fully connected (FC). An ANN composed of several fully connected layers is called a Multilayer Perceptron (MLP) (Figure 2.1). The equations 2.1 compose the forward and backward propagation of this type of architecture with two layers, where the optimization function is the Gradient Descent. The superscript t regards the values of the variables throughout time, whilst the subscript i regards the layer associated with them.

$$
\begin{aligned}
f_i &= \mathrm{X}W_i^t + \mathrm{b}_i^t \\
\mathrm{H}_1 &= (\sigma_1 \circ f_1)(\mathrm{X}) \\
\mathrm{H}_2 &= (\sigma_2 \circ f_2)(\mathrm{H}_1) \\
\mathrm{O} &= f(\mathrm{H}_2) \\
\mathrm{L} &= l(\mathrm{O}, \mathrm{y}) \\
\mathrm{W}_\mathrm{i}^{\mathrm{t}+1} &= \mathrm{W}_\mathrm{i}^{\mathrm{t}} \times \eta \times \nabla L \\
\mathrm{b}_\mathrm{i}^{\mathrm{t}+1} &= \mathrm{b}_\mathrm{i}^{\mathrm{t}} \times \eta \times \nabla L
\end{aligned}
\tag{2.1}
$$

Figure 2.1 – Multilayer perceptron with hidden layers. This example contains a hidden layer with five hidden neurons.



Source: Adapted from Zhang et al. (2023).

2.1.2.2 Convolutional Neural Networks

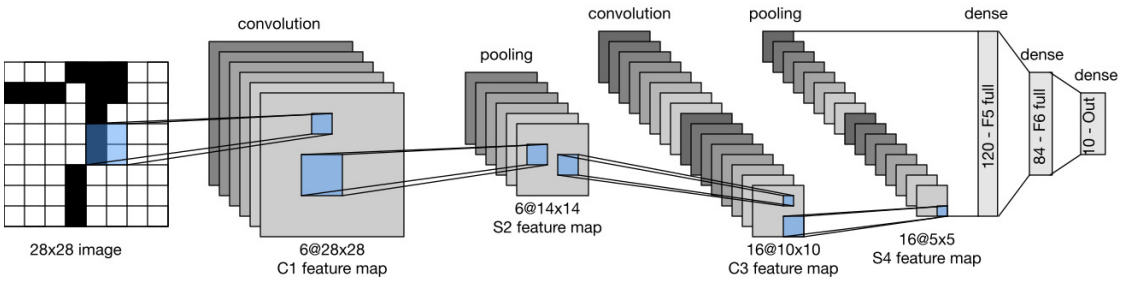
However, working with a flattened one-dimensional layer architecture makes the neural network invariant to the input data features' order. When dealing with images, this invariance discards information relative to the relation between nearby pixels. Also, it makes the computation of these algorithms infeasible due to the large number of parameters needed to train the network for the high dimensionality of this type of data.

Convolutional Neural Networks (CNNs) overcome this problem by using convolutional layers (Equation 2.2). By defining a kernel V that is applied to local regions of the image channels X (i.g. RGB), the number of trainable parameters is reduced without losing the internal relationships in the image. This type of layer adds translation invariance (ZHANG et al., 1988) and permits learning from regions rather than single pixels. Deeper layers also capture longer-range features in deep convolutional neural networks (ZHANG et al., 2023; RUSSELL; NORVIG, 2022; LECUN et al., 1989).

Figure 2.2 shows a Deep CNN, an ANN composed of multiple convolutional, pooling, and FC layers. The first has parameters that produce a unique activation value to learn patterns within its defined kernel. The second applies a function within the kernel to reduce the values inside it to a lower rank matrix, producing a smaller image that contains summarized information of the original. Lastly, the FC layers flatten and learn from the embeddings, produced after a series of convolution and pooling blocks, to reach the desired output (LECUN et al., 1989).

$$[\mathrm{H}]_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c} [\mathrm{V}]_{a,b,c,d} [\mathrm{X}]_{i+a,j+b,c} \tag{2.2}$$

Figure 2.2 – Data flow in LeNet. The input is a handwritten digit, and the output is a probability of over ten possible outcomes.



Source: LeCun et al. (1989)

2.1.2.3 Recurrent Neural Networks

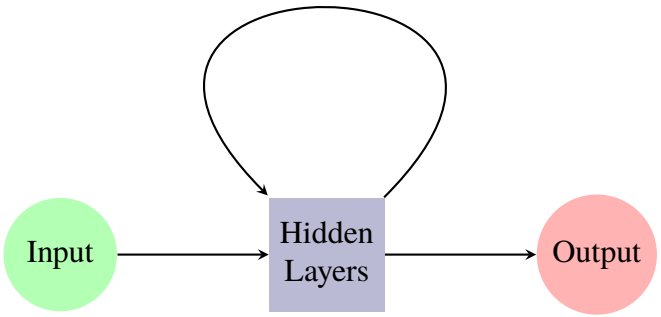
Various tasks also require dealing with sequentially structured data, i.g. natural language processing (NLP), controlling robots, video processing, etc. Recurrent Neural Networks (RNNs) were created to extend the potential of neural networks and tackle sequentially structured information. While standard connections propagate to a subsequent layer, RNNs use recurrent connections between their neurons to capture the dynamics of sequences carrying information across time steps (ZHANG et al., 2023; RUSSELL; NORVIG, 2022).

The recurrence may be thought of as cycles in the network, as shown in Figure 2.3, and permits the network to learn not only from the examples themselves but from the information derived by the sequence in which they appear. Such information could be related to the order of words in a phrase or the position of an object in a video through time. RNNs grow their recurrent hidden layers (Figure 2.4) according to the number of dimensions of the sequential data and introduce the notion of a hidden state (ZHANG et al., 2023; RUSSELL; NORVIG, 2022).

"Compared with [the MLP layers], [RNN layers] add one more term $\boldsymbol{H_{t-1}W_{hh}}$ and thus instantiates [(Equation 2.3)]. From the relationship between hidden layer outputs $\boldsymbol{H_t}$ and $\boldsymbol{H_{t-1}}$ of adjacent time steps, we know that these variables captured and retained the sequence's historical information up to their current time step, just like the state or memory of the neural network's current time step" (ZHANG et al., 2023). The superscript text in Equation 2.3 describes the layers' numbers, and the weights subscript text describes the dimensions of the matrices.

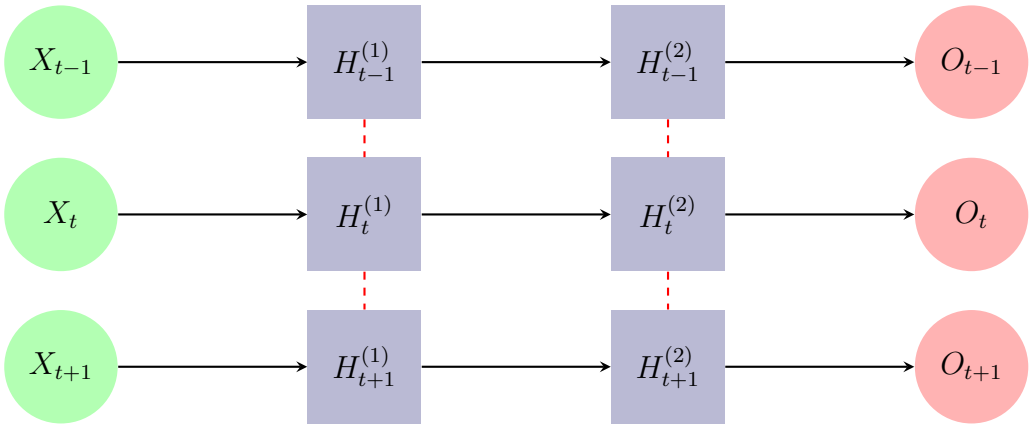
$$\mathrm{H}_t^{(l)} = \phi_l(\mathrm{H}_t^{(l-1)}\mathrm{W}_{\mathrm{xh}}^{(l)} + \mathrm{H}_{t-1}^{(l)}\mathrm{W}_{\mathrm{hh}}^{(l)} + \mathrm{b}_{\mathrm{h}}^{(l)}) \tag{2.3}$$

Figure 2.3 – RNN with recurrent connections depicted via cyclic edges.



Source: Adapted from Zhang et al. (2023).

Figure 2.4 – Unfolded RNN over time steps. Recurrent edges span adjacent time steps, while conventional connections are computed synchronously.



Source: Adapted from Zhang et al. (2023).

2.1.2.4 3D Convolutional Networks

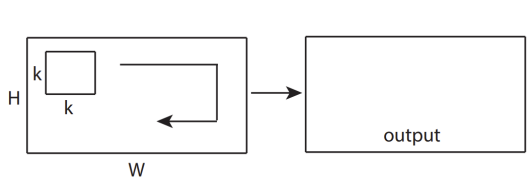
3D convolutional networks are an extension of CNNs. 3D kernels can capture temporal and spatial information along a video volume of data by including a depth dimension k (Equation 2.4), while 2D ones only capture each frame's spatial information (Figure 2.5). Therefore, it is well-suited for spatiotemporal correlation feature learning.

$$[\mathsf{H}]_{k,i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c=-\Delta}^{\Delta} \sum_{d} [\mathsf{V}]_{a,b,c,d,e} [\mathsf{X}]_{k+a,i+b,j+c,d} \tag{2.4}$$

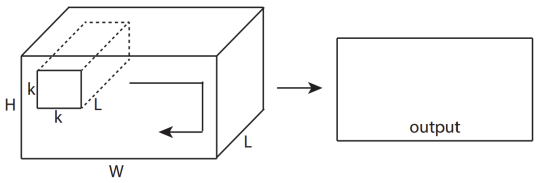
Tran et al. (2015) analyze several variations of 3D convolution architectures. The results show that a homogeneous setting with convolution kernels of 3 Œ 3 Œ 3 is the best option for 3D CNNs, consistent with the results shown in Simonyan e Zisserman (2015) for 2D CNNs. Then, it proposes the C3D model (Figure 2.6), a deep 3D convolutional neural network that uses a sequence of 3d convolution layers.

Using the deconvolution method, Tran et al. (2015) observes that C3D weights, at first,

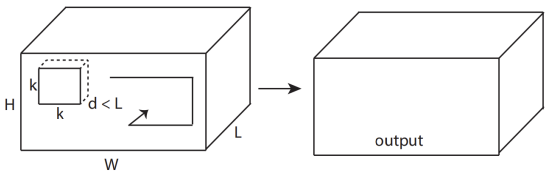
Figure 2.5 – 2D and 3D convolution operations. a) Applying 2D Convolution on an image results in an image. b) Applying 2D Convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D Convolution on a video volume results in another volume, preserving temporal information of the input signal.



(a) 2D Convolution.



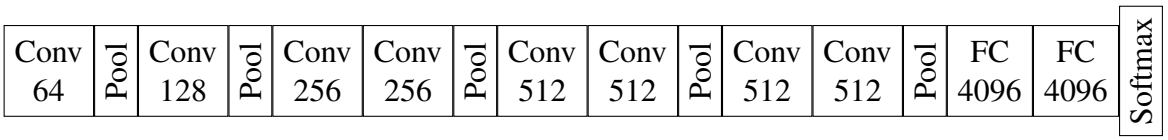
(b) 2D Convolution on multiple frames.



(c) 3D Convolution.

Source: Tran et al. (2015)

Figure 2.6 – C3D architecture. C3D net has eight convolutions, five max-pooling layers, and two fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with a stride of 1 in both spatial and temporal dimensions. The number of filters is denoted in each box. The 3D pooling layers are denoted from *pool1* to *pool5*. All pooling kernels are $2 \times 2 \times 2$, except for pool1 is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.



Source: Adapted from Tran et al. (2015).

focus on learning spatial information and, then, track the motion of the objects on the video. In the example shown in Figure 2.7, the network "focuses on the whole person and then tracks the motion of the pole vault performance over the rest of the frames".

2.1.3 Attention Mechanisms and Transformers

Until the last decade, Deep Learning algorithms had remarkably little compared to their first breakthroughs, and most of the progress was focused on enhancing and scaling up current architectures. However, this trend was recently broken after the adoption of the Attention mechanism (BAHDANAU; CHO; BENGIO, 2016) and, following, the Transformer architecture (VASWANI et al., 2017). Transformer-based models have emerged as state-of-the-art, or at least competitive, methods for tasks such as NLP, image recognition, object detection, semantic segmentation, su-

Figure 2.7 – Visualization of C3D model using deconvolution method. Interestingly, C3D captures appearance for the first few frames but only attends to salient motion afterward. It is best viewed on a color screen.

Source: Tran et al. (2015).

perresolution, speech recognition, reinforcement learning, and graph neural networks (ZHANG et al., 2023).

2.1.3.1 Attention

Consider the key-value database $\mathcal{D} \stackrel{\text{def}}{=} \{(\mathrm{k}_1, \mathrm{v}_1), \ldots (\mathrm{k}_m, \mathrm{v}_m)\}$ and a query q. Invoking this query on the database means returning all the values that associate the query with its key. This association can be measured in significance levels, and multiple values may be returned. The attention mechanism proposed by (BAHDANAU; CHO; BENGIO, 2016) performs this process using scalar attention weights $\alpha(\mathrm{q}, \mathrm{k}_i) \in \mathbb{R}$ $(i = 1, \ldots, m)$ that give importance to keys with more significance. The scaled values are then combined by attention pooling (Equation 2.5), which returns a linear combination of the data in $\mathcal{D}$, a new value defined from the existing ones and their relation to the query.

$$\text{Attention}(\mathrm{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathrm{q}, \mathrm{k}_i)\mathrm{v}_i \tag{2.5}$$

2.1.3.2 Attention Scoring Functions

Attention scoring functions (a) are mostly used as attention functions since distance-based ones are slightly more expensive to compute. A common attention scoring function is the dot-product attention (Equation 2.6), where d is a regularization term (VASWANI et al., 2017). To accept queries and keys of different dimensions, the dot-product is adapted through an MLP-like approach called additive attention (Equation 2.7). The query and the key are fed into an MLP with a single hidden layer, where $\mathrm{W}_q \in \mathbb{R}^{h \times q}$, $\mathrm{W}_k \in \mathbb{R}^{h \times k}$ and $\mathrm{w}_v \in \mathbb{R}^h$. The bias term was turned off for simplicity (ZHANG et al., 2023). Optimizing the layers of these MLPs is one of the key areas of advance in recent years (SHOEYBI et al., 2020).

$$a(\mathrm{q}, \mathrm{k}_i) = \mathrm{q}^\top \mathrm{k}_i / \sqrt{d} \tag{2.6}$$

$$a(\mathrm{q}, \mathrm{k}) = \mathrm{w}_v^\top \sigma(\mathrm{W}_q \mathrm{q} + \mathrm{W}_k \mathrm{k}) \in \mathbb{R} \tag{2.7}$$

2.1.3.3 Masked Attention

Similar to ANN outputs when dealing with multiple class results, attention weights still need to be normalized into a differentiable probability density function. Hence, the results are intelligible, and the network can be trained efficiently. To this end, softmax (Equation 2.8) is attention mechanisms' most popular normalization function.

$$\alpha(\mathrm{q}, \mathrm{k}_i) = \mathrm{softmax}(a(\mathrm{q}, \mathrm{k}_i)) = \frac{\exp(a(\mathrm{q}, \mathrm{k}_i))}{\sum_{j=1} \exp(a(\mathrm{q}, \mathrm{k}_j))}. \tag{2.8}$$

Softmax also efficiently permits attention mechanisms to evaluate queries with different dimensions. To achieve that, the attention weights relative to dimensions unused by the query are set to large negative numbers. When the softmax operation is applied, these weights are set to zero. This operation is heavily optimized for GPUs and much more efficient than conditional statements. "Since it is such a common problem, it has a name: the masked softmax operation" (ZHANG et al., 2023).

2.1.3.4 Bahdanau Attention Mechanism

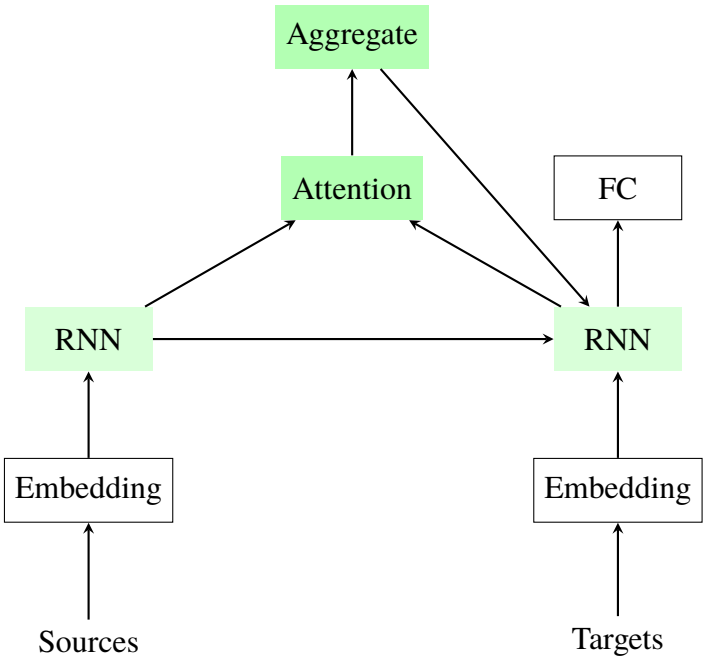
Bahdanau, Cho e Bengio (2016) used an attention mechanism in a language model. It uses the hidden states of a sequence-to-sequence model with an encoder-decoder RNN architecture to build the next output, selectively aggregating different parts of the input sequence (Figure 2.8). Data embeddings in the NLP context used as input and returned as output of language models, such as the Bahdanau Attention Mechanism, are called tokens. Therefore, i.g. for a translation task, the input data is the tokens representing the words in a phrase.

According to Zhang et al. (2023), the Bahdanau attention mechanism has become one of the most impactful innovations in deep learning over the last decade, serving as the foundation for the development of Transformers and inspiring numerous related architectures.

2.1.3.5 Multi-head Attention

Multi-head attention combines FC layers with linear activation and attention mechanisms to capture different knowledge subspaces. (Equation 2.9) Queries $\mathrm{q} \in \mathbb{R}^{d_q}$, keys $\mathrm{k} \in \mathbb{R}^{d_k}$ and values

Figure 2.8 – Layers in an RNN encoder-decoder model with the Bahdanau attention mechanism.



Source: Adapted from Zhang et al. (2023).

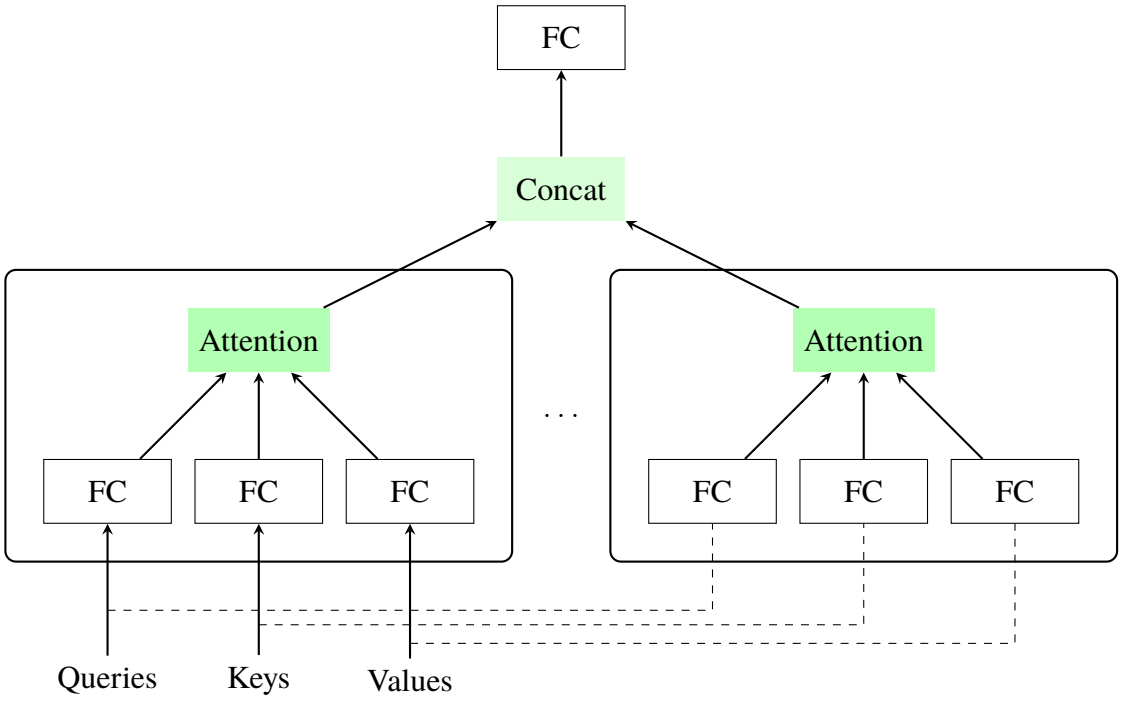
$\mathrm{v} \in \mathbb{R}^{d_v}$ are used as input for an FC layer each with weights $\mathrm{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q}$, $\mathrm{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k}$ and $\mathrm{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$, respectively. The embedding from these FC layers is then used as input to an attention-scoring function, which formalizes a *head*.

$$\mathrm{h}_i = f(\mathrm{W}_i^{(q)}\mathrm{q}, \mathrm{W}_i^{(k)}\mathrm{k}, \mathrm{W}_i^{(v)}\mathrm{v}) \in \mathbb{R}^{p_v} \tag{2.9}$$

This process increases the number of independent attention weights learned from the same query, key, and values tuple, which leads to the discovery of different independent features from various ranges of the same data. Multi-head attention concatenates the *heads* outputs and maps them to a function (i.e. another linear transformation via an FC layer with weights $\mathrm{W}_o \in \mathbb{R}^{p_o \times hp_v}$ - Equation 2.10). The output merges the several subspaces into a generalized and more representative embedding based on a query, keys, and values given (ZHANG et al., 2023; VASWANI et al., 2017).

$$\mathrm{W}_o \begin{bmatrix} \mathrm{h}_1 \\ \vdots \\ \mathrm{h}_h \end{bmatrix} \in \mathbb{R}^{p_o} \tag{2.10}$$

Figure 2.9 – Multi-head attention, where multiple heads are concatenated and then linearly transformed.



Source: Adapted from Zhang et al. (2023).

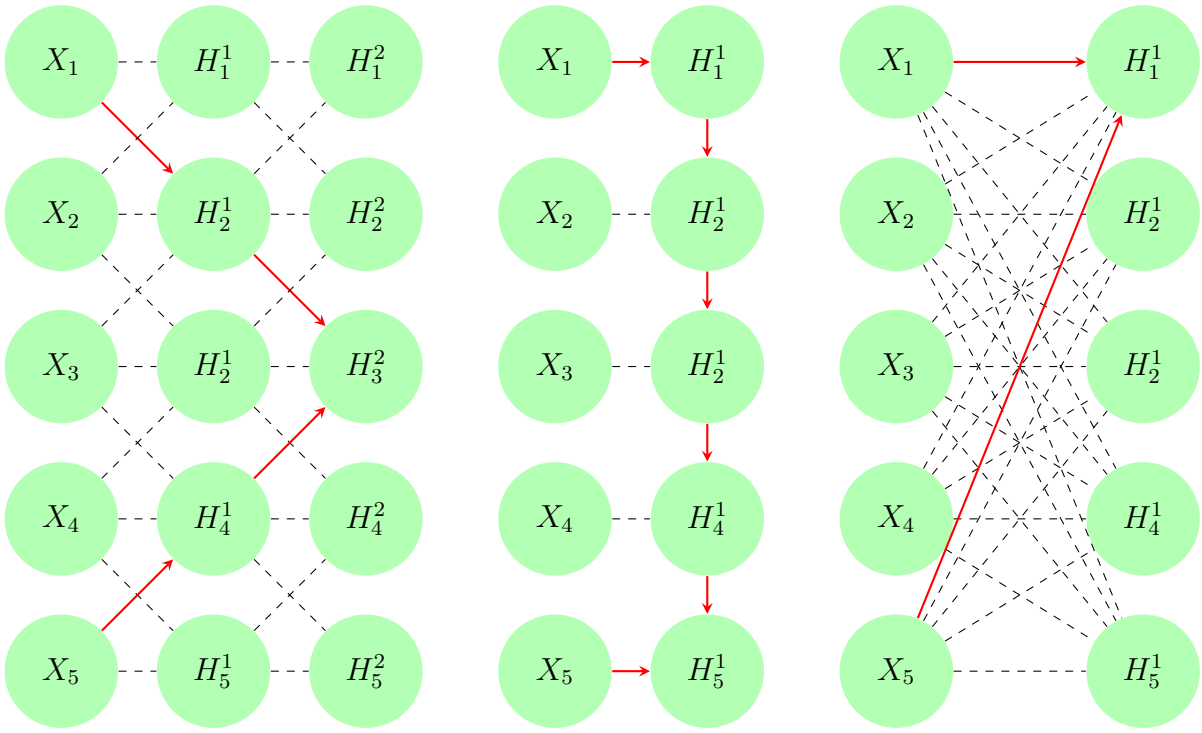
2.1.3.6 Self-Attention

Consider a series of input data $\mathrm{x}_1, \ldots, \mathrm{x}_n | \mathrm{x}_i \in \mathbb{R}^d$. Self-attention computes the embedding from the attention mechanism using only this source for all queries, keys, and values. Vaswani et al. (2017) also adds the output to the query through residual connections (HE et al., 2015). This builds a more representative embedding based on how the original information fits inside the context of all the data (ZHANG et al., 2023; VASWANI et al., 2017).

Deep learning layers should reduce their complexity, the number of sequential operations performed, and the path between any combination of sequence positions from the input. This leads to efficiency and permits leveraging parallel computation and learning long-range dependencies in the data. Like recurrent and convolutional layers, self-attention layers encode a sequence of symbol representations into another sequence of the same length (Figure 2.10). Compared to convolutional and recurrent layers (Table 2.2), self-attention layers enjoy parallelized processing, leveraging GPU computation power, and have the shortest maximum path length between the latter (VASWANI et al., 2017).

Furthermore, attention automatically and dynamically models the weights and induces bias according to the input data relation to itself and the context, different from MLPs, CNNs, and RNNs that have fixed weights and predefined inductive bias approaches (i.e. kernels, hidden states). This leads to more generic architectures that need more data and computation resources and have more representation power (VASWANI et al., 2017; SANFORD; HSU; TELGARSKY,

Figure 2.10 – Comparing CNN (padding tokens are omitted), RNN, and self-attention architectures.



a) CNN architecture. b) RNN architecture. b) Self-attention architecture.

Source: Adapted from Vaswani et al. (2017).

Table 2.2 – Maximum path lengths, per-layer complexity, and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions, and r is the size of the neighborhood in restricted self-attention.

Layer Type	**Complexity**	**Sequential Operations**	**Maximum Path Length**
Self-Attention	$O(n^2.d)$	$O(1)$	$O(1)$
Recurrent	$O(n.d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k.n.d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r.n.d)$	$O(1)$	$O(n/r)$

Source: Adapted from Vaswani et al. (2017).

2023; BAHDANAU; CHO; BENGIO, 2016).

2.1.3.7 Positional Encoding

Self-attention does not preserve the order of the sequence arrived, previously mentioned as use cases of RNNs. To tackle this task, positional encoding avoids processing sequences sequentially as the latter detours the quadratic computational complexity concerning the sequence length and leverages parallel computing. Given the sequence of d-dimensional embed-

dings $\mathrm{X} \in \mathbb{R}^{n \times d}$, the positional encoding outputs $\mathrm{X} + \mathrm{PE}$. Hence, the original embedding inherits its positional information inside its values. For example, Vaswani et al. (2017) uses $\mathrm{PE} \in \mathbb{R}^{n \times d}$ with positions pos, columns i and values

$$\begin{aligned} \mathrm{PE}_{pos,2i} &= \sin\left(\frac{pos}{10000^{2j/d}}\right), \\ \mathrm{PE}_{pos,2i+1} &= \cos\left(\frac{pos}{10000^{2j/d}}\right). \end{aligned} \tag{2.11}$$

2.1.3.8 Transformers

Unlike previous self-attention models that relied on RNNs or CNNs to generate inputs, Vaswani et al. (2017) transformers (Figure 2.11) depended solely on attention mechanisms. The model is composed of an encoder-decoder architecture. In contrast to Bahdanau, Cho e Bengio (2016), transformers perform sequence-to-sequence learning using positional encoding in both the encoder and decoder inputs.

The encoder is a stack of identical blocks composed of a multi-head self-attention layer with normalization and a position-wise MLP with residual connection and normalization. These are important for training a deep model. The inputs from a previous block feed the next one.

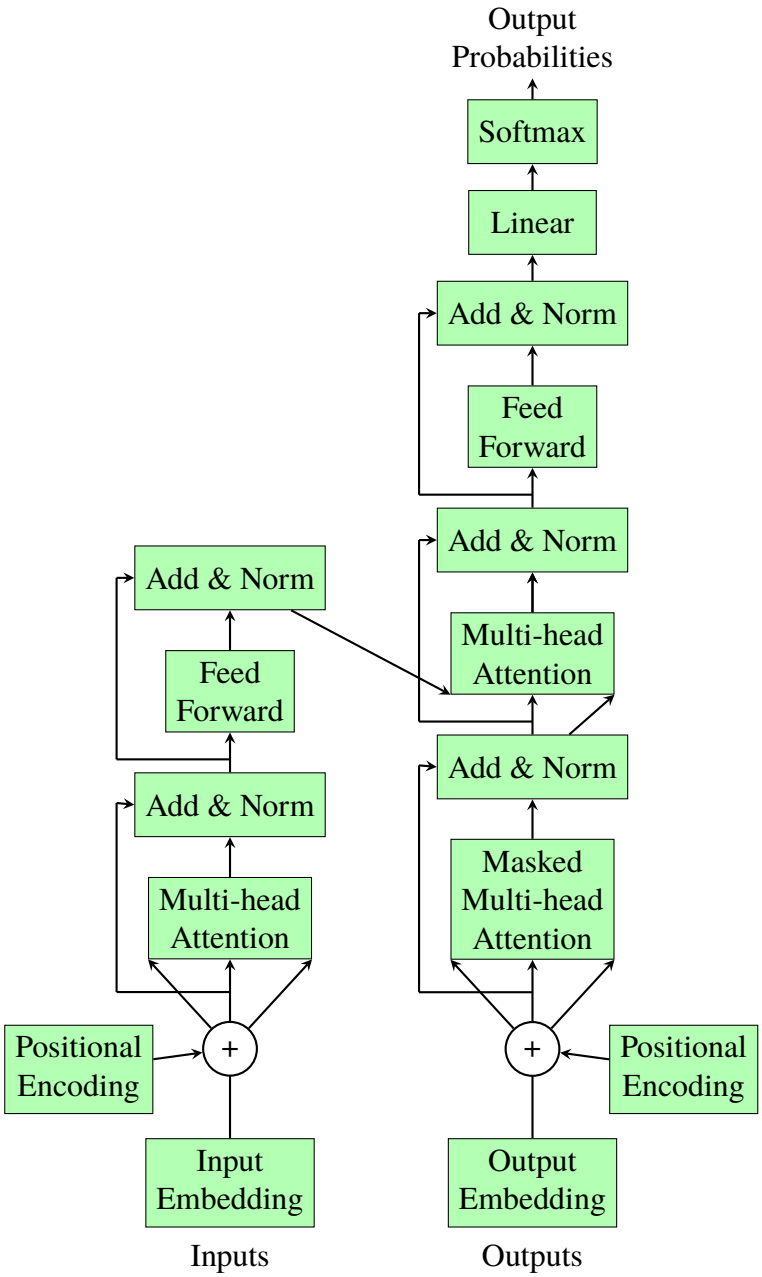
The positionswise MLP is two linear layer MLP, also known as expand-and-contract network, with weights $W_1 \in \mathbb{R}^{d \times h}$ and $W_1 \in \mathbb{R}^{h \times d}$. The weights expand the output embedding of the dimension of the self-attention layer to h and then contract it back to its original size (Equation 2.12). This network operates independently and identically on each input sequence position, allowing the model to learn positional information effectively while processing calculations in parallel with parameter efficiency. After passing through the multi-head self-attention layer, this small MLP is necessary to learn from the embeddings.

$$FFN(\mathrm{x}) = \sigma(\mathrm{xW}_1 + \mathrm{b1})\mathrm{W}_2 + \mathrm{b}_2 \tag{2.12}$$

The decoder is also built using the same strategy but inserts a third layer, encoder-decoder cross-attention, between the previous two. The cross-attention layer is another normalized multi-head self-attention layer that uses the keys and values from the encoder output, while queries still come from the last layer. Cross-attention contextualizes the decoder with the encoder embeddings. The decoder also uses causal masked attention on the first multi-head self-attention layer to preserve the model's auto-regressive property, which ensures a sequential dependency between data input (ZHANG et al., 2023; VASWANI et al., 2017).

Just like Bahdanau, Cho e Bengio (2016) model, transformers are suitable for NLP tasks and can be fed with and learn new tokens. Note that although the transformer architecture uses

Figure 2.11 – The Transformer - model architecture.



Source: Adapted from Vaswani et al. (2017).

both an encoder and decoder, this is not always true for every application (DEVLIN et al., 2019; BROWN et al., 2020).

2.2 Related Work

EAP can entail multimodal data (RGB, depth, skeleton, etc.), and the typical methods in the literature work on egocentric/first-person videos. Our research focuses on RGB video-only data and third-person or second-person views for the human-robot collaboration (HRC) context. We extensively research the literature to understand the state of robotic vision within this scope and search for EAP models.

Vision Transformer-based approaches are highly prominent and significantly influential in action recognition and EAP. Therefore, we begin by discussing the HRC context research field. Then we introduce an older model that tackles the HRC context, and explore the fundamentals of Vision Transformers (ViT) and Video Vision Transformers (ViViT). Finally, we perform a deeper investigation into specific EAP models.

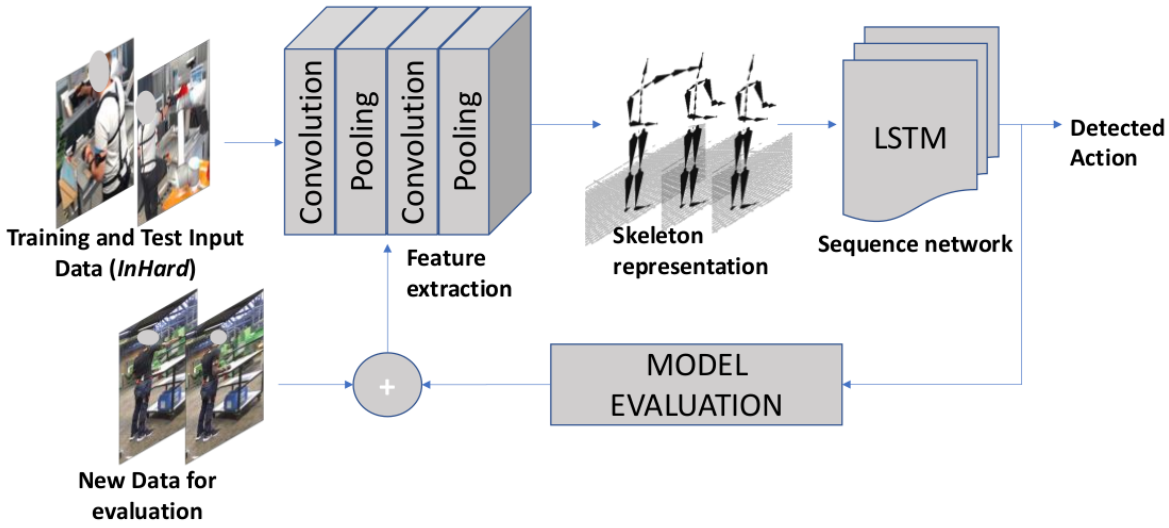
2.2.1 Robotic Vision for Human-Robot Collaboration (HRC)

Human-robot collaboration (HRC) is an area of research that studies how to work out teamwork between humans and robots to achieve shared goals with a common purpose and directed outcome. "Collaboration with a robot can help to improve task speed and work productivity, reduce the number of errors, and improve human safety to minimize repetition fatigue and injuries" (ROBINSON et al., 2023). This monograph focuses on the impact of robotic vision in the HRC context.

Robotic vision is a crucial and mainstream computer vision process for robots to analyze human actions. By integrating machines with visual sensors, such as cameras and LiDARs, they can better understand and interact with the natural world and make relevant decisions and actions. For action prediction and collaborative tasks, Robinson et al. (2023) shows that these two contexts have been poorly investigated.

Robinson et al. (2023) also shows that little work has been done on robotic vision for handover and collaborative manipulation between humans and robots. Furthermore, only three transformer-based approaches were cited, none for EAP, and no dataset with second- and third-view visual data was used for training in the presence of both the robot and the human. Also, no “studies that have attempted to combine a shared representation, prediction, and signaling to achieve true collaboration” (BÜTEPAGE; KRAGIC, 2017) were found.

Figure 2.12 – Fusion of CNN and LSTM architecture for action recognition and model evaluation using InHard and a new dataset.



Source: Tuli, Patel e Manns (2022).

2.2.2 Harnets

Tuli, Patel e Manns (2022) work shows a common deep-learning approach to tackle action recognition in videos: a fusion between a CNN and an LSTM. Since videos are composed of spatial and temporal information, this type of model uses CNNs to encode spatial data and feed an LSTM that uncovers the temporal relationship embedded within the generated values. The model used in the paper is composed of an Inception V3 model (SZEGEDY et al., 2015) as the backbone and an LSTM (ZHANG et al., 2023) adapted to the length of the video data as the head. Figure 2.12 shows the model architecture.

The model created by Tuli, Patel e Manns (2022) was trained on human activity recognition datasets and showed the importance of robots performing intelligent behavior inside working environments. It showed the lack of research published in the field and also generated a custom dataset within the context. Some of the outcomes obtained from this paper are then used as a comparison to our work in the results chapter.

2.2.3 Vision Transformers (ViT)

"Similar to the landscape of network architecture design in natural language processing, Transformers have also become a game-changer in computer vision " (ZHANG et al., 2023).

Although Vaswani et al. (2017) transformer architecture emerged as the state of the art in various NLP tasks, substituting RNNs, it could not be trained on image data. Cordonnier, Loukas e Jaggi (2020) proved that a multi-head self-attention layer with sufficient heads is at least as expressive as any convolutional layer. Then, Dosovitskiy et al. (2021) created a transformer-

based model for image classification called Vision Transformer (ViT - Figure 2.13).

This model extracts non-overlapping $m = hw/p^2$ patches from the input image with height h, width w, and c channels, where p is the height and width of the patch. These are then flattened to a vector of length cp^2. The flattened image patches are mapped to D dimensions with a trainable linear projection $\mathrm{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$ and summed with learnable positional embeddings $\mathrm{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$ (Equation 2.13). The patches are fed together with a special token <*cls*> to the Transformer that outputs the same number of vectors. Finally, the Transformer encodes the m + 1 vectors, and the <*cls*> token is used to find the class using an MLP (ZHANG et al., 2023).

$$\mathrm{z}_0 = [\mathrm{x}_{class};\ \mathrm{x}_p^1\mathrm{E}\ ;\ \mathrm{x}_p^2\mathrm{E}\ ;\ \cdots\ ;\ \mathrm{x}_p^N\mathrm{E}] + \mathrm{E}_{pos} \tag{2.13}$$

Different from the original Transformer architecture, ViT only uses the encoder block. Also, layers are normalized before the multi-head attention and the MLP, leading to more effective or efficient training for Transformers (XIONG et al., 2020).

Three variations of the ViT architecture were implemented in Dosovitskiy et al. (2021) as shown in Table 2.3.

Table 2.3 – Details of Vision Transformer model variants.

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Source: Adapted from Dosovitskiy et al. (2021).

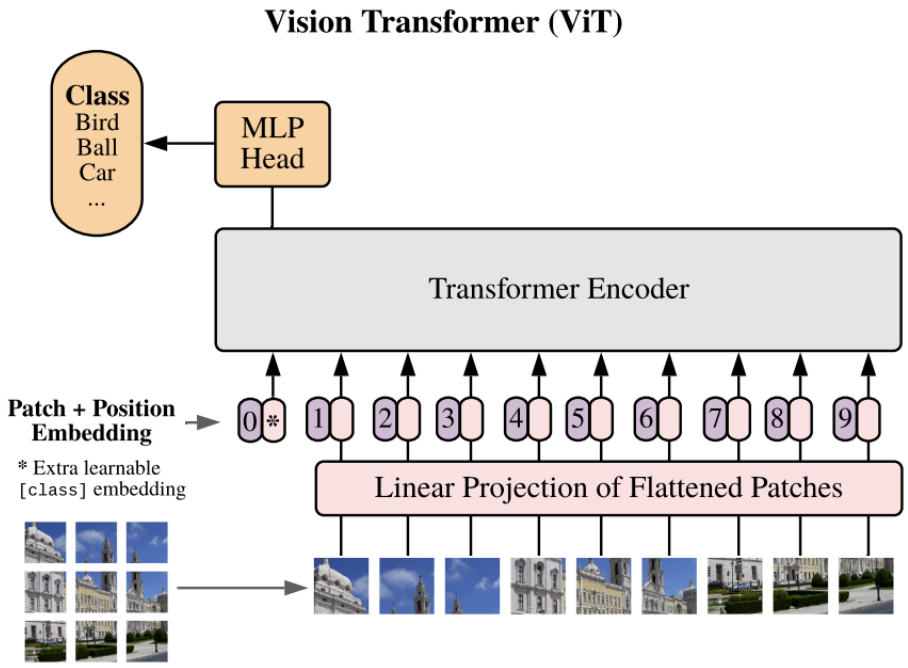
When trained with large datasets, ViT outperforms the state-of-the-art CNN-based models, demonstrating Transformers' scalability superiority. A lot of effort has been put into how to effectively train this model (TOUVRON et al., 2021) and use high-quality images (LIU et al., 2021b).

2.2.4 Video Vision Transformer (ViViT)

Videos are a type of data that merges spatial and temporal information. Each unit of the data sequence that composes a video is called a frame and contains an image. Since they are inside a sequence, frames are related to each other just like tokens in NLP.

To deal with both types of information simultaneously, CNNs were initially adapted to deal with 3D information in models defined as deep 3D convolutional architectures. After the emergence of transformers, this model was augmented by introducing attention mechanisms into their layers, following the same trend as what happened to RNNs in Bahdanau, Cho e Bengio

Figure 2.13 – The vision Transformer architecture. In this example, an image is split into nine patches. A special "<cls>" token and the nine flattened image patches are transformed via patch embedding and n Transformer encoder blocks into ten representations, respectively. The "<cls>" representation is further transformed into the output label.



Source: Dosovitskiy et al. (2021).

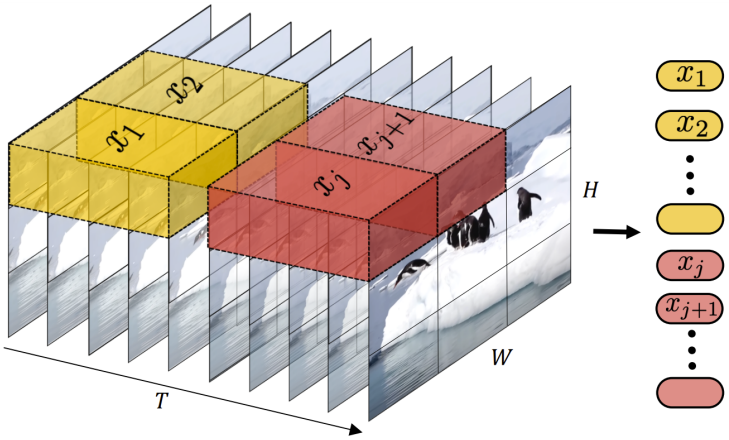
(2016). Inspired by Dosovitskiy et al. (2021), Arnab et al. (2021) introduces pure-transformer models for video classification, called Video Vision Transformers.

Arnab et al. (2021) first defines 2 ways of tokenizing a video $\mathrm{V} \in \mathbb{R}^{T \times H \times W \times C}$ into $\hat{\mathrm{z}} \in \mathbb{R}^{n_t \times n_h \times n_w \times d}$. Uniform Frame Sampling extracts n_t frames, embeds each independently using the same method as in Dosovitskiy et al. (2021), and concatenates them, generating $n_t \cdot n_h \cdot n_w$. Tubelet Embedding (Figure 2.14) extracts "tubes" $\in \mathbb{R}^{t \times h \times w}$, where $n_t = \lfloor \frac{T}{t} \rfloor$, $n_h = \lfloor \frac{H}{h} \rfloor$ and $n_w = \lfloor \frac{W}{w} \rfloor$ from the video volume and linearly project them into $\mathbb{R}^d$. The latter extends Dosovitskiy et al. (2021) embedding and is analogous to a 3D convolution. Arnab et al. (2021) shows that Tubelet Embedding performs better and uses it to train its models.

According to Arnab et al. (2021), this approach integrates spatiotemporal information during the tokenization process, unlike "Uniform frame sampling", where temporal information from various frames is combined by the Transformer.

Next, four models are defined and tested based on the ViT-Base as the backbone. The first model, Spatio-Temporal Attention, feeds a ViT using uniform frame sampling. Its downside is that self-attention layers do not perform well on long videos due to their quadratic complexity

Figure 2.14 – Tubelet embedding: extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.



Source: Arnab et al. (2021).

concerning the number of tokens. The Factorised Encoder model (Figure 2.15) uses a two-stage encoding process and has the best performance and floating-point operations (FLOPs) tradeoff. First, it uses a spatial encoder to process tokens extracted from the same temporal index. It transforms the representations outputted and feeds them into a temporal encoder that predicts a class. The Factorised self-attention and Factorised dot-product attention models proposed did not perform as well.

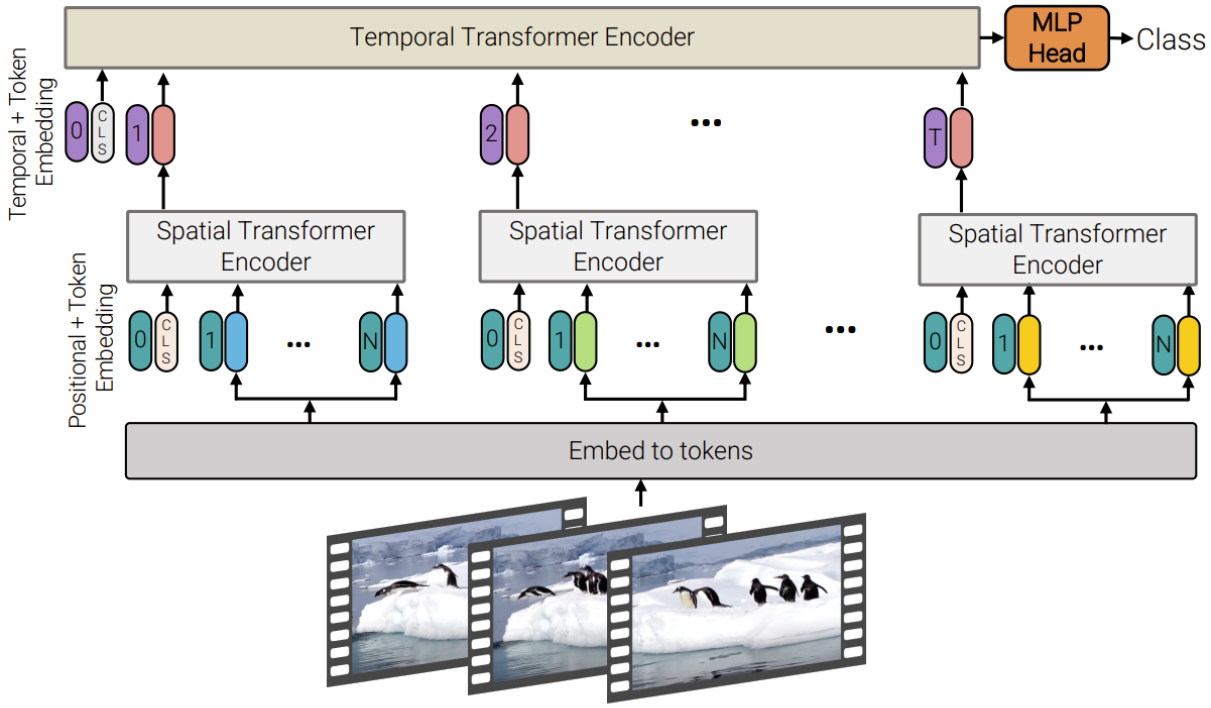
2.2.5 EAP attention models

Although the works of Dosovitskiy et al. (2021) and Arnab et al. (2021) showed some advantages of using self-attention for image encoding, 3D convolution networks still held the best results as backbones for feature extraction in EAP models among most of the benchmarks. Several of these networks have been inspired by Howard et al. (2017) and He et al. (2015) (LIU et al., 2022; KONDRATYUK et al., 2021). Instead of focusing on only one architecture choice, Stergiou e Damen (2023) leverages both attention and 3D convolutional architecture advantages at once in one model, called TemPr (Figure 2.16).

Stergiou e Damen (2023) tackles the EAP task by working on multiple video scales using progressive sampling. Sampling videos uniformly, generating segments of equal size might separate discriminative action patterns between segments. Given the partially observed video T_ρ, progressive sampling separates the segments into scales $s_i = \{1, \ldots, T_{s_i}\}$, where $T_{s_i} = \lceil \frac{i}{n} \cdot T_\rho \rceil \; \forall i \in \mathbb{N} = 1, \ldots, n$, and selects F frames randomly from each.

For each scale s_i, an volume x_i of size $3 \times F \times H \times W$, where F is ordered, H is height and W is width. The volumes are input to a shared encoder Φ that outputs the per-scale, multi-dimensional spatio-temporal encoded features volumes z_i, of size $C \times t \times h \times w$. Then, z_i is reshaped to $C \times (thw)$ and concatenated to Fourier Positional Embeddings of size $n \times (thw)$.

Figure 2.15 – Factorized encoder (Model 2). This model consists of two transformer encoders in series: the first model interacts between tokens extracted from the same temporal index to produce a latent representation per time index. The second transformer models interactions between time steps. It thus corresponds to a "late fusion" of spatial and temporal information.



Source: Arnab et al. (2021).

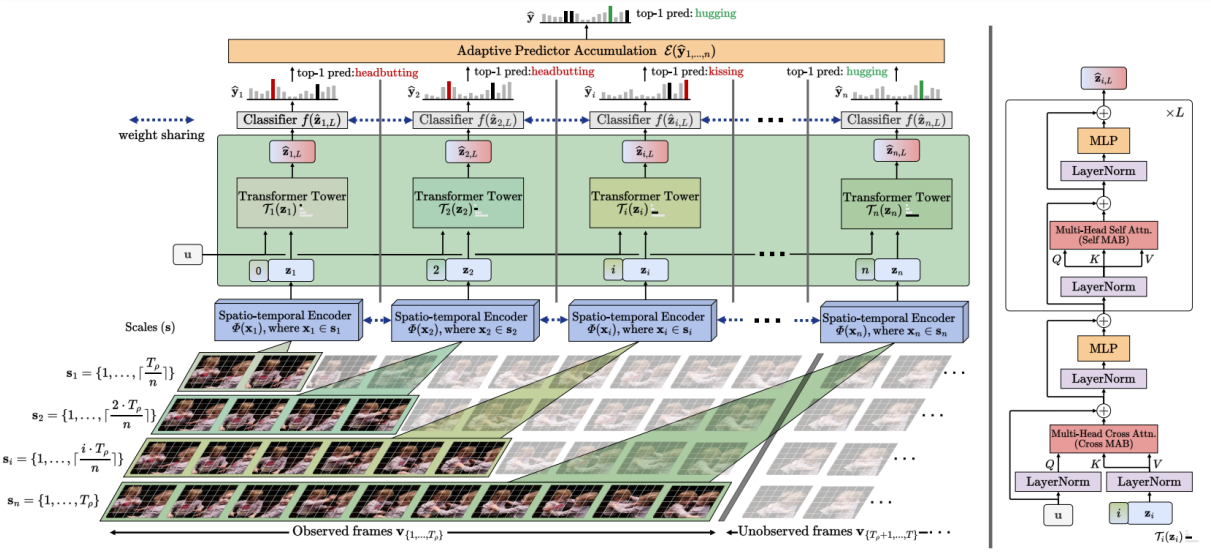
The features z_i are them fed to an attention tower $\mathcal{T}_i$ that outputs $\hat{\mathrm{z}}_i$.

Each tower is composed of two components. A cross multi-head attention block (Equation 2.14) uses a trainable latent array u to produce of size $C \times d$ ($d \ll thw$) to create the asymmetric attention dot-product between u and z_i. This bottleneck approach with a latent vector turns attention tower models more efficient than using only a stack of L self-attention models, which is the second component. Additionally, the attention tower shares a linear classifier $\hat{\mathrm{y}} = f(\hat{\mathrm{z}}_{i,L})$ that establishes a joint feature space across scales.

$$\begin{aligned} h_{i,0} &= MCA(LN(\mathrm{u}), LN(z_i)) + \mathrm{u} \; \forall \; i \in \mathbb{N} \\ \hat{z}_{i,0} &= MLP(LN(h_{i,0})) + h_{i,0} \end{aligned} \tag{2.14}$$

Finally, the $\hat{\mathrm{y}}$ labels generated are fed to the adaptive aggregation function, which calculates the individual attention tower's predictions and confidences. The aggregation function chosen for the model in the paper was the Adaptive Pooling (STERGIOU; POPPE, 2023), shown in Equation 2.15. The class agreement $\mathcal{E}_{eICW}$ reduces the uncertainty of individual predictions using the Exponential Inverse Coefficient Weighting, in which $DSC(\cdot)$ is the Dice-Sørensen

Figure 2.16 – (Left) TemPr architecture. Features are extracted over each input x_i sampled from video scale s_i and combined with the scale and spatiotemporal positional encodings. The encoded features z_i are passed to attention towers $\mathcal{T}_i$ which output tensors $\hat{\mathrm{z}}_{i,L}$ in the latent space. The shared-weight classifier $f(\cdot)$ is applied to every tower output to make per-scale predictions. These predictions are aggregated by the aggregation function $\mathcal{E}(\cdot)$ for early action prediction over the observed frames. (Right) Attention Tower. Each utilizes pre-norm and a shared latent array u for the cross-attention block (Cross MAB). This is followed by a stack of L self-attention blocks (Self MAB).



Source: Stergiou e Damen (2023).

Coefficient (Equation 2.16). The confidence agreement $\mathcal{E}_{eM}$ gives higher weights to predictions with higher class probability using exponential maximum (i.e. softmax) across all predictions. The hyperparameter β weights the importance of each agreement.

$$\mathcal{E}(\hat{\mathrm{y}}_{1,\dots,n}) = \sum_{i \in \mathbb{N}} \beta \cdot \mathcal{E}_{eICW}\left(\hat{\mathrm{y}}_i, \bar{\hat{\mathrm{y}}}\right) + (1-\beta) \cdot \mathcal{E}_{eM}\left(\hat{\mathrm{y}}_i\right) \tag{2.15}$$

$$\mathcal{E}_{eICW}\left(\hat{\mathrm{y}}_i, \bar{\hat{\mathrm{y}}}\right) = \frac{e^{DSC(\hat{\mathrm{y}}_i, \bar{\hat{\mathrm{y}}})^{-1}}}{\sum_{k \in \mathbb{N}} e^{DSC(\hat{\mathrm{y}}_k, \bar{\hat{\mathrm{y}}})^{-1}}} \cdot \hat{\mathrm{y}}_i \tag{2.16}$$

Stergiou e Damen (2023) model is state of the art across all major benchmarks for EAP.

2.3 Final remarks

Several works create approaches to Action Recognition and EAP within the HRI context (RYOO et al., 2015; TULI; PATEL; MANNS, 2022). However, to our knowledge, we are the first to tackle the EAP task using attention mechanisms within the HRC context. Moreover, we advance research to achieve true collaboration between humans and robots using such tools.

3 Development

In this chapter, we describe how the EAP models were evaluated in terms of suitability for the HRC setting using the InHARD dataset (DALLEL VINCENT HAVARD, 2020). The only attention-based model proposed for EAP found in the literature that uses RGB second and third-view video data was the TemPr (STERGIOU; DAMEN, 2023). Therefore, we focus our work on evaluating this model broadly.

3.1 Methodology

This section describes the instruments and methodological procedures chosen as the base decisions repeated in every experiment.

3.1.1 Standardized Experiment environment

3.1.1.1 Virtual environment

We used a Docker Container (MERKEL, 2014) to keep experiment runs consistent and comparable. The Docker file inherits a base image from the Nvidia GPU Cloud (NGC) Catalog with CUDA version 12.4.1 (NVIDIA, 2024). Also, the Docker file is adapted to support the Stergiou e Damen (2023) model code [†] dependencies.

3.1.1.2 Physical environment

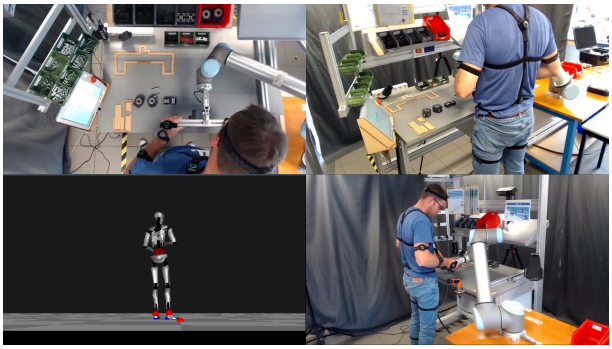
The virtual environment is executed on a machine with an Intelő Core™i7-8700K CPU, an NVIDIA GEFORCE GTX 1080 GPU, and 32GB DDR4 RAM. This project only runs on machines with GPU support.

3.1.2 Dataset

The Industrial Human Action Recognition Dataset (InHARD) is an RGB+S (RGB + Skeleton) collection of videos from a real-world setting for industrial human action recognition (DALLEL VINCENT HAVARD, 2020). The goal of the dataset is to foster the development of learning techniques for analyzing human actions in settings involving human-robot collaboration. It contains over 2 million frames collected from 16 distinct subjects, 13 industrial action classes, and over 4,800 action samples. For the sake of this project's scope, only the RGB video data is used for training.

[†]Code available at: <https://github.com/alexandrosstergiou/progressive-action-prediction>

Figure 3.1 – InHARD frame example.



Source: DALLEL Vincent HAVARD (2020).

Three C920 cameras were used to capture RGB video data from different perspectives of the same action. To capture the left and right views, two cameras were positioned at the same height but at different horizontal angles, -45ř and +45ř. The third camera was mounted above the subjects to capture a top-down view.

The dataset saves the recordings of every camera in a mosaic style. Camera 1 records top views and is displayed in the top left quarter. Camera 2 records left-side views and is shown in the top right quarter. Camera 3 records right-side views and is displayed in the bottom right quarter (Figure 3.1).

The data is labeled using 13 meta-action labels and 74 action labels - full description in Annex A. Meta-action labels generally describe the action being executed. In contrast, action labels describe the specifics of the action using more details of the environment, objects used, place, and distance measures, etc. Furthermore, the latter includes the goal operation ID.

DALLEL Vincent HAVARD (2020) suggests that the data is divided into two categories of subjects: experts and beginners, according to the subject's expertise with the manipulation. Subjects who complete the entire manipulation in an average duration of less than 6 minutes are classified as experts, whereas the remaining subjects are categorized as beginners. The training, validation, and test splits are defined based on these categories (Table 3.1).

Table 3.1 – Number of beginner and expert subject video files on the training, validation, and testing sets defined in DALLEL Vincent HAVARD (2020).

	Begginer	**Expert**
Training	17	9
Validation	4	2
Testing	4	2

Source: DALLEL Vincent HAVARD (2020).

3.1.3 Model

The complete model architecture consists of a backbone, a head, and a fusion layer, and several strategies were chosen to train it in different experiments.

3.1.3.1 Backbone

Following the same strategy as in Stergiou e Damen (2023), the backbone extracts features from the videos and feeds the head. Two models were selected as backbones for the experiments: X3D-M (FEICHTENHOFER, 2020) and VideoMAEv2-Base (WANG et al., 2023a). They were chosen because the X3D network represents the best backbone from the results of Stergiou e Damen (2023), and the VideoMAEv2 network is the state-of-the-art model for Action Recognition. All parameters of the backbone are pre-trained and set as not trainable.

X3D uses convolutional layers, whilst VideoMAEv2 uses attention. By comparing the results from these models, it is possible to determine which kind of approach, convolution or attention, works best within this work's architectural organization as backbones.

3.1.3.2 Head

The model's head is a TemPr block with three attention towers. The data feed to the head is first pre-processed by sampling progressive video scales from the RGB video data. The sampling strategy followed is withdrawing three groups of 16 frames of the video randomly and sequentially for the training and validation/testing phases, respectively. All parameters from the head are trainable.

The TemPr module has 256 latent arrays with dimension 512, that compose the trainable latent array u used to create the asymmetric attention dot-product with the features extracted from the backbone. Moreover, owers with one cross-attention layer and one self-attention layer. The cross-attention and self-attention layers have one and eight heads, respectively, with a dimension of 64.

3.1.3.3 Fusion

The fusion block used is the Adaptive Pooling (STERGIOU; POPPE, 2023) shown in 2.15, which had the best results at Stergiou e Damen (2023) across several other fusion options. The parameters from the Fusion block are trainable.

3.1.3.4 Precision

The whole model is trained using half-precision (16 bits). A PyTorch scaler and auto-casting adapt the backbone and ensure all three parts of the model work with the same precision.

3.1.3.5 Optmization

The model is optimized using the Weighted Cross Entropy Loss in which the weights are calculated using the *balanced* heuristics from the Pedregosa et al. (2011) library, inspired by King e Zeng (2001). This strategy is chosen to deal with the InHARD dataset class imbalance.

Furthermore, the optimization strategy is built upon a sequence of optimizers and learning rate schedulers. First, the AdamW optimizer with a weight decay of 0.01 is used together with two learning rate schedulers: a RAdam warmup (LIU et al., 2021a) and a cosine annealing with warm restarts (LOSHCHILOV; HUTTER, 2017). After a specific number of epochs, defined as a hyperparameter, the optimizer is changed to a Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a cosine annealing learning rate scheduler. Moreover, an early stopping procedure with patience 7 that observes the mean validation loss was used.

3.1.4 Metrics

In this section, we described the metrics used to evaluate the model. MLflow (MLFLOW, 2024) is used to track and compare different experiment-run models' prediction metrics.

3.1.4.1 Accuracy

The accuracy score is the probability that the model's prediction is correct. It gives different importance to classes based on their frequency in the dataset (Equation 3.1) (GRANDINI; BAGLI; VISANI, 2020).

$$Accuracy = \frac{Correctly\ classified\ predictions}{Total\ of\ predictions} \tag{3.1}$$

3.1.4.2 OvR macro-average ROC AUC score

Precision (Equation 3.2) and recall (Equation 3.3) are the number of False Positives (FP) and False Negatives (FN), respectively, compared to the number of True Positives (TP) predicted. The Receiver Operating Characteristic (ROC) curve illustrates the performance of a classifier by plotting the precision against the recall over various classification thresholds of confidence, which determines the minimum probability required for a positive prediction. These are called false positive rates and true positive rates, respectively.

The ROC Area Under the Curve (AUC) score, representing the area under the ROC curve, summarizes a model's ability to distinguish between positive and negative instances across all thresholds. One over Rest (OvR) macro-average ROC AUC score calculates the average of ROC for each class using the macro-average precision (Equation 3.4) and recall (Equation 3.5), and then calculates the AUC. This strategy considers under-represented classes as important as

highly populated classes. High OvR macro-average ROC AUC values indicate good performance in all the K classes; the opposite is true for low performance (PEDREGOSA et al., 2011).

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{3.2}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{3.3}$$

$$Macro\ Average\ Precision = \sum_{k=1}^{K} Precision_k \tag{3.4}$$

$$Macro\ Average\ Recall = \sum_{k=1}^{K} Recall_k \tag{3.5}$$

3.2 Experiments

This section describes the experiment variables and the procedures they are related to.

3.2.1 Hyperparameters tuning

Many hyperparameters are tunable in the TemPr network and the model optimization. The hyperparameters tuned in the experiments are found with a description and their possible values in Table 3.2. Optuna (AKIBA et al., 2019) is used to tune all the chosen hyperparameters automatically using the Tree-structured Parzen Estimator (OZAKI; NOMURA; ONISHI, 2020) with the validation loss as the objective function. Furthermore, the fANOVA algorithm (HUTTER; HOOS; LEYTON-BROWN, 2014) is used to calculate the hyperparameters' importance to analyze the impact each of them has on the results. MLflow (MLFLOW, 2024) is used to track all the variations of the experiment and compare each.

3.2.2 Training, validation and testing

Each model with its unique set of hyperparameters is trained using batches of 8 from the training set with an observation ratio of ρ = 1.0 (complete video). For every epoch, the model loss is calculated by inferring all data from the validation set. Every time the mean loss from the inference of the validation set batches surpasses the loss from the last epoch the inference was made, the current model is saved using MLflow's PyTorch functionalities.

After training all model variations, the best trials, as decided by the Optuna algorithm, are tested based on the harmonic mean of the chosen metrics for this project. The model with the best testing results is recorded as the best model. Then, the best model hyperparameters set is used to train two models using the observation ratio ρ = 0.3 and 0.5, respectively.

Table 3.2 – Hyperparameters with descriptions and possible values used for the experiments.

Hyperparameter	Description	Possible values
Training		
`epochs`	Number of epochs used to train the model	Integer from 30 to 40
`lr`	Base learning rate	Float from 0.01 to 0.1
`lr_change_optimizer`	Epoch to trigger change in model optimizer	Integer from 20 to 30
Architecture		
`attn_dropout`	Dropout rate for dropout layer at the output of each attention block	Float from 0.01 to 0.15
`ff_dropout`	Dropout rate for dropout layer at the output of each positionwise MLP	Float from 0.01 to 0.15
`backbone`	Backbone used in the model	X3D or VideoMAEv2

Source: Elaborated by the author.

4 Results

In this chapter, the results are shown and discussed.

4.1 Hyperparameters tuning results

First, five experiments were executed using variations of the selected hyperparameters, which were chosen using the Optuna algorithm for each experiment. These experiments used full short action videos (observation ratio $\rho = 1.0$). Table 4.1 shows the hyperparameters selected for each run.

The fANOVA algorithm results are shown in Figure 4.1. The high importance given to the attention dropout shows once more how this kind of layer is critical to the learning process. Related to this hyperparameter is the Pointwise FC layer dropout, which is affected directly by the latter. The Optuna algorithm could not find an equilibrium between how much each of these two related hyperparameters should affect the network.

Following in terms of importance is the learning rate scheduler hyperparameter. This shows that the learning strategy adopted has a great impact on the results and the combination. The other hyperparameters from the fANOVA results still lack new experiments to point out results that show any other suitable approach related to them.

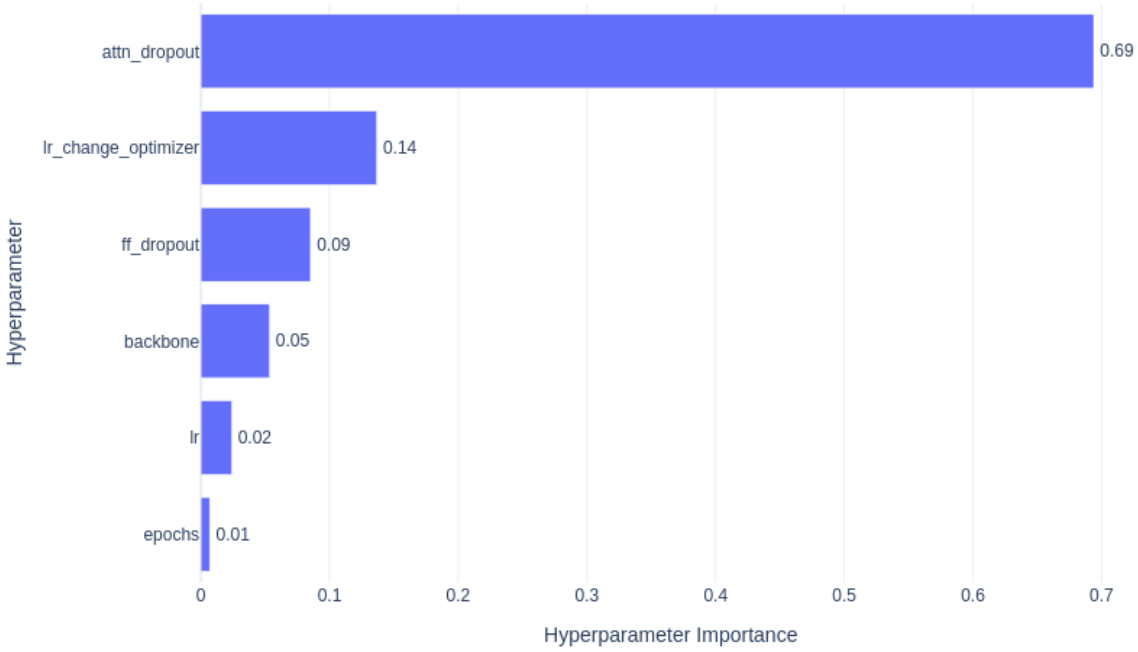
Figure 4.2 shows the final results from the Optuna optimization problem for each trial. Trial 3 did not improve the optimization results, therefore, it was disconsidered by Optuna, and that is the reason for the graph to only show that the optimization objective result remained the same. The graph shows a tendency of the validation loss to decrease with noise. This indicates that the Optuna optimization algorithm is going in the right direction to find the trial closest to the optimum.

4.2 Training and validation results

The training loss, validation loss, accuracy, and OvR macro-average ROC AUC score were collected during the training. The results per epoch for the collected data are found in Figures 4.3 to 4.6. The values in the graphs were pre-processed using Univariate Spline Interpolation (VIRTANEN et al., 2020) smoothing function for enhanced visualization. The model from Trial 4 performed best and was chosen by Optuna as the only trial for testing.

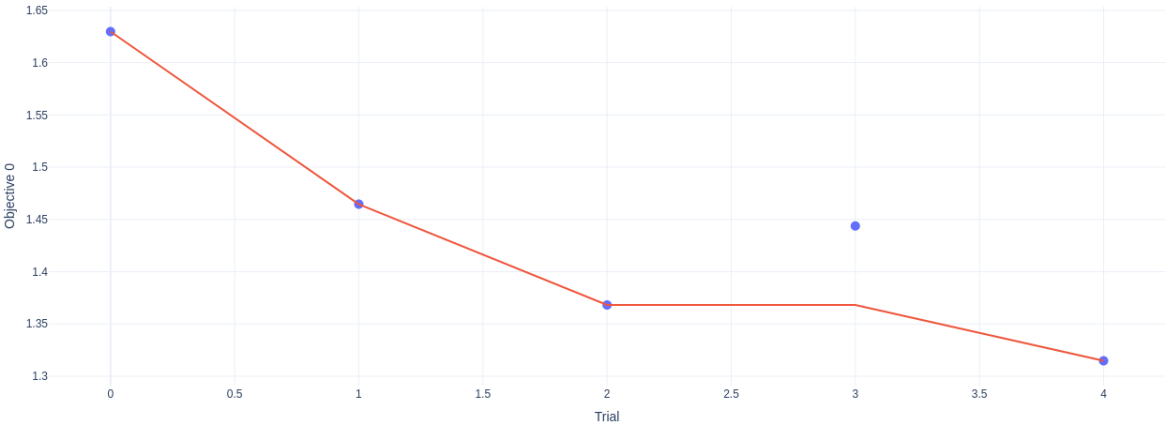
In all of the metrics collected, it is clear that the models have learned successfully. However,

Figure 4.1 – fANOVA hyperparameters' importance results.



Source: Elaborated by the author.

Figure 4.2 – Optuna optimization results based on the validation loss.



Source: Elaborated by the author.

Table 4.1 – Hyperparameters selected in the experiments.

Hyperparameter	**Trial 0**	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**
`epochs`	34	39	32	31	36
`lr`	0.064	0.012	0.057	0.051	0.025
`lr_change_optimizer`	21	30	24	28	20
`attn_dropout`	0.14	0.094	0.036	0.051	0.017
`ff_dropout`	0.112	0.109	0.053	0.061	0.095
`backbone`	X3D		VideoMAEv2		

Source: Elaborated by the author.

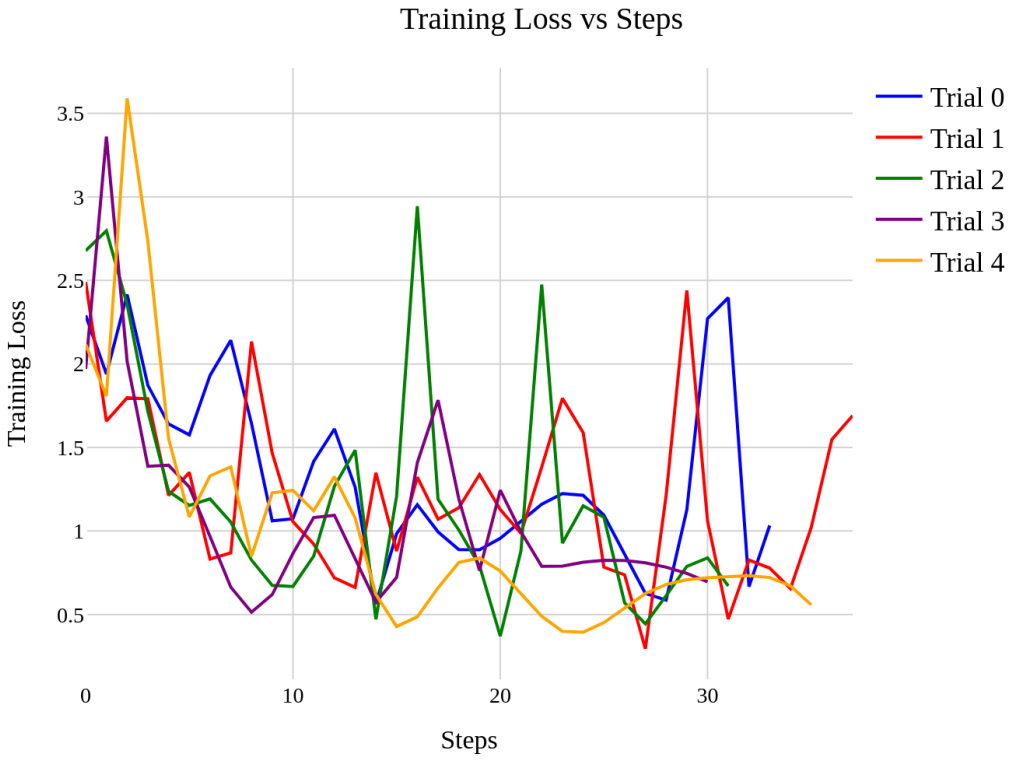
the low complexity of the TemPr block at the head of the model led to an underfitting issue, and it seems that only Trial 1 was still evolving at a higher rate. This can be explained by the fact that Trial 1 had the lowest base learning rate and high dropout rates.

Overall, on all trials, all trial models have overcome the InHARD class imbalance, as shown in Figure 4.6. The high OvR macro-average ROC AUC score demonstrates that the optimization and learning strategies have led to success in training the models without giving importance to more representative classes. Furthermore, the models still could achieve a relevant accuracy result, shown in Figure 4.5.

4.3 Testing metrics results

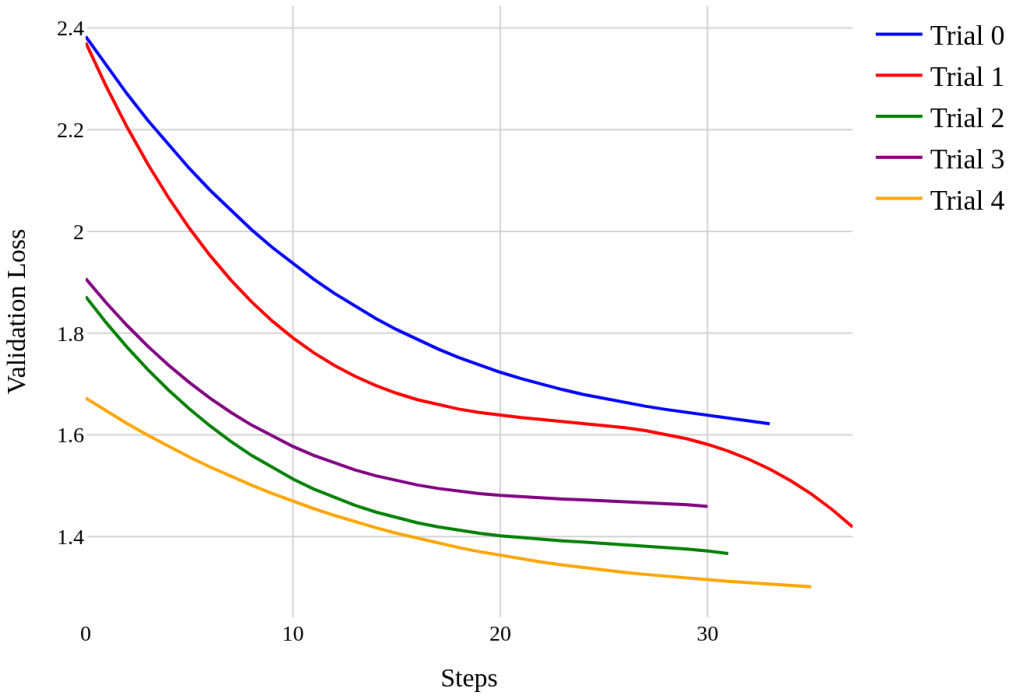
The model with the hyperparameters set of Trial 4 was tested using the testing set, and the accuracy and OvR macro-average ROC AUC score were collected. The results from the inference using this model are found in Table 4.2. The high accuracy results, markedly near the validation and training metrics results, show that the trained model achieved high generalization power.

Figure 4.3 – Training loss per epoch.



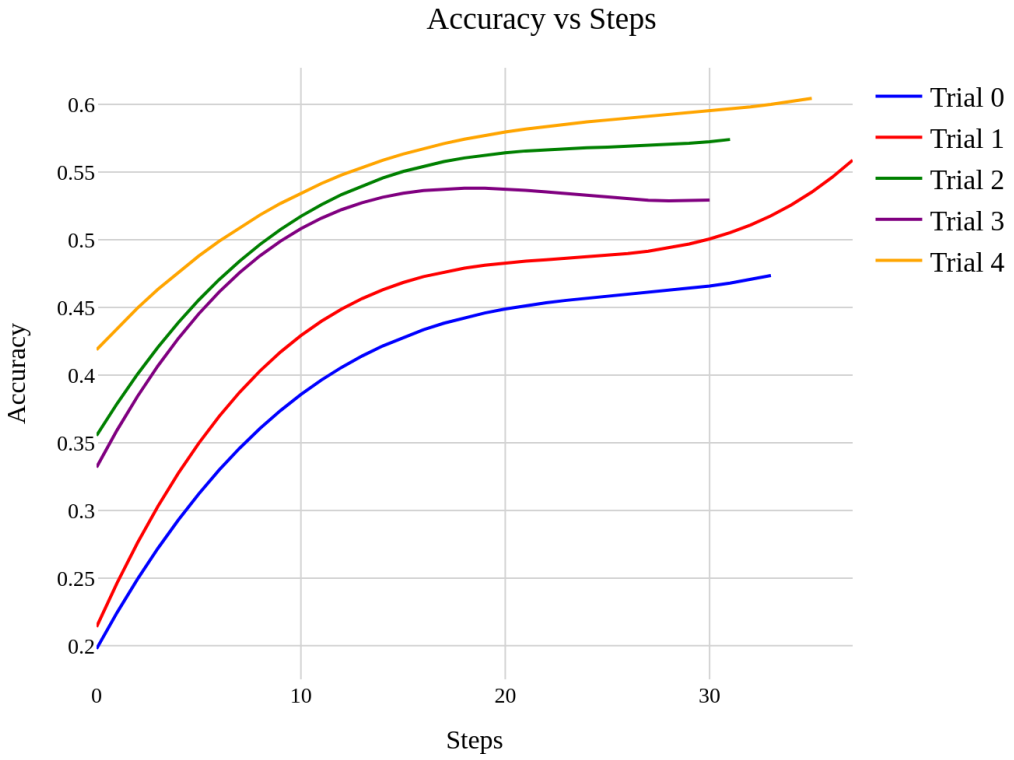
Source: Elaborated by the author.

Figure 4.4 – Validation loss per epoch.



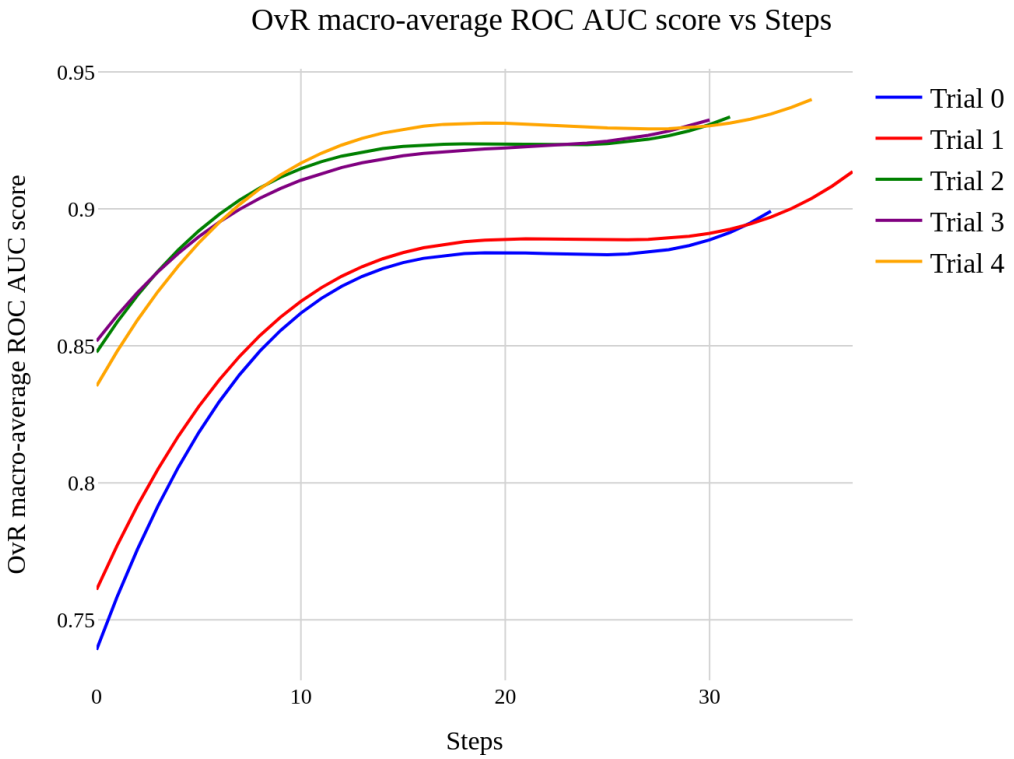
Source: Elaborated by the author.

Figure 4.5 – Accuracy per epoch.



Source: Elaborated by the author.

Figure 4.6 – OvR macro-average ROC AUC score per epoch.



Source: Elaborated by the author.

Table 4.2 – Result of the metrics for experiments with $\rho = 1.0$.

Accuracy	OvR macro-average ROC AUC score
0.598	0.933

Source: Elaborated by the author.

4.4 Training, validating, and testing with lower observation metrics

The hyperparameters set from Trial 4 were used to train two models with observation ratio $\rho = 0.3$ and 0.5, respectively. The accuracy, OvR macro-average ROC AUC score, training loss, and validation loss per epoch collected during the training are found in Figures 4.7 to 4.10. The testing accuracy and OvR macro-average ROC AUC score from the inference using these models are found in Table 4.3.

Figures 4.7 to 4.10 show very similar results to the ones when the observation ratio ρ was equal to 1.0. The same conclusions from the metrics are visible and the success in training too. Thus, the models trained with lower ρ manage to learn exceptionally considering the lack of information given.

The fact that with $\rho = 0.3$ the model surpassed the accuracy compared to when trained with $\rho = 0.5$ emphasizes the underfitting issue with the model, and increases the possibility of even greater results with a more complex model. Nevertheless, the model still presents great generalization even in this scenario.

Table 4.3 – Result of the metrics for experiments with $\rho = 0.3$ and 0.5.

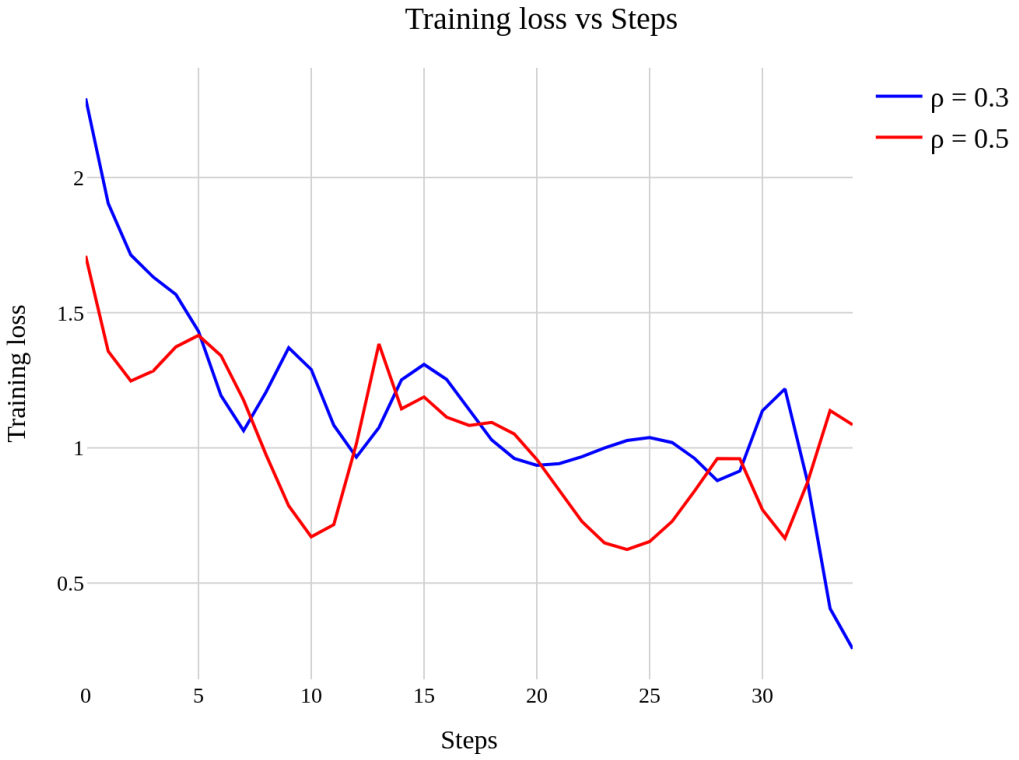
	Accuracy	OvR macro-average ROC AUC score
$\rho = 0.3$	0.558	0.921
$\rho = 0.5$	0.548	0.928

Source: Elaborated by the author.

4.5 Results comparison and discussion

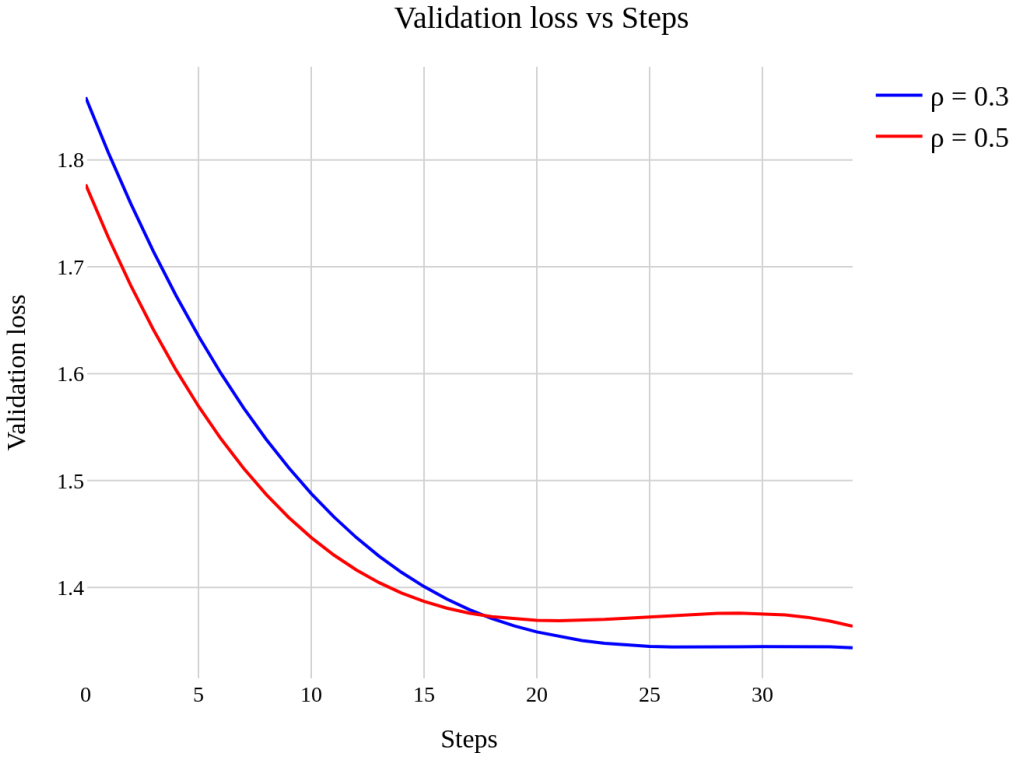
To evaluate the results of our model, we compare it to the work of Tuli, Patel e Manns (2022), which has also dealt with the InHARD dataset. Their model has achieved a total of 74% accuracy while our model achieved 59%. The advantage found in our model relies on the fact that even when reducing the observation ratio to $\rho = 0.3$, our model still performs with 55% accuracy. This slight reduction shows its suitability for EAP. Furthermore, our model also excelled on the OvR macro-average ROC AUC score, 93%, which is a more reliable performance metric when dealing with imbalanced datasets such as the InHARD.

Figure 4.7 – Training loss per epoch when using lower observation ratios ρ = 0.3 and 0.5.



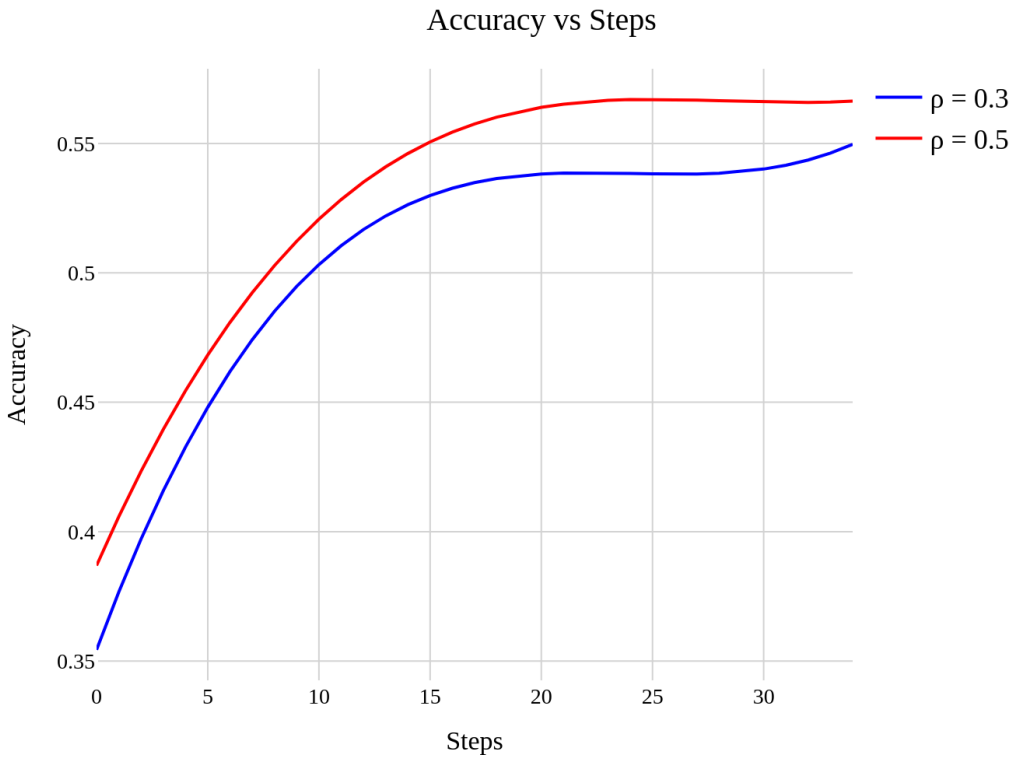
Source: Elaborated by the author.

Figure 4.8 – Validation loss per epoch when using lower observation ratios ρ = 0.3 and 0.5.



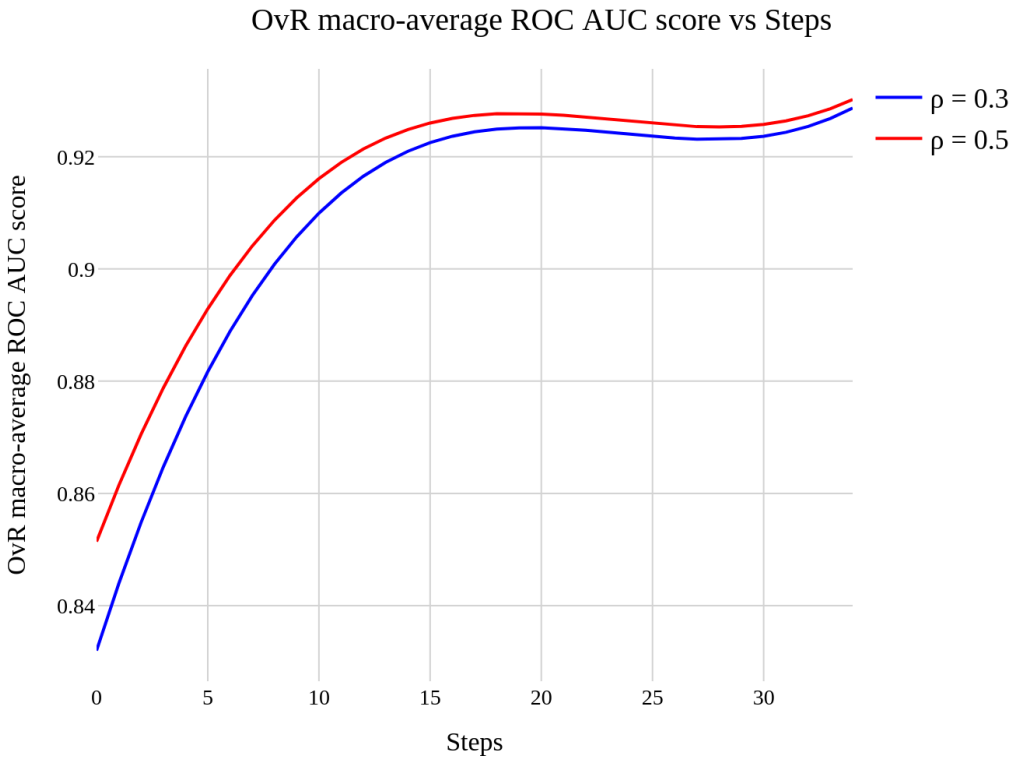
Source: Elaborated by the author.

Figure 4.9 – Accuracy per epoch when using lower observation ratios $\rho = 0.3$ and 0.5.



Source: Elaborated by the author.

Figure 4.10 – OvR macro-average ROC AUC score per epoch when using lower observation ratios $\rho = 0.3$ and 0.5.



Source: Elaborated by the author.

The small number of trainable parameters might explain the low accuracy $\rho = 1.0$ and the increase in the validation loss of the model when ρ is set as 0.5. An oversimplified model may not be able to learn all the different characteristics of each action when more information about it is given. Furthermore, the accuracy when dealing with lower observation ratios would also increase with a more complex structure. Therefore, it is noticeable that the standard architectural decisions for the TemPr model could be changed for higher performance, and there is a high chance this strategy will succeed.

In addition, the hyperparameters' importance results show that some of them were slightly unimportant. Since they are all crucial for the training of the model, the range in which they were constrained did not give the Optuna algorithm enough freedom to choose a better value. Therefore, the choice of the hyperparameters' possible range could be improved to accept more options.

5 Final Considerations

This chapter presents the final considerations of the monograph, describing the objectives reached and future work proposals.

5.1 Conclusion

The present work developed an approach for early action prediction using attention mechanisms in the human-robot collaboration context. First, the only model known to tackle the early action prediction task, TemPr, proposed by Vaswani et al. (2017), is identified by systematically uncovering and extracting a range of early action prediction models from existing literature. To our knowledge, we are the first to tackle the EAP task using attention mechanisms within the HRC context.

The uncovered model was extensively tested using automatic hyperparameter tuning and modern metric comparison tools, Optuna and Mlflow. The results show that the training on the Industrial Human Action Recognition Dataset (InHARD) was successful. Moreover, the model performed exceptionally on lower observation ratios ρ, demonstrating the great capabilities of the strategy adopted for the model.

In conclusion, both of our main objectives were achieved. The EAP model found after systematic research on papers in the area was trained on a Human-Robot Collaboration dataset and succeeded in inferring short-term action videos. Furthermore, this study provided significantly useful information to evolve robotics vision for providing a safer and more collaborative working environment for humans and robots.

5.2 Future work

To achieve better results, the model architectural decisions, such as the number of attention towers, the number of layers and their dimensions, the number of latents, and attention layer heads, could be included as hyperparameters to be tuned automatically by Optuna. Furthermore, the range of values the hyperparameters can assume could be extended for further experimentation. The skeleton and depth data available in the InHARD dataset could also be used for training.

A bigger model would be necessary to learn more data patterns and this would require higher computational power. In addition, if multi-GPU training is possible the Fully Sharded Data Parallel (FSDP) would also be important for better distribution of the model parameters.

Bibliography

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. Disponível em: <https://arxiv.org/abs/1907.10902>.

ARNAB, A.; DEHGHANI, M.; HEIGOLD, G.; SUN, C.; LUI, M.; SCHMID, C. *ViViT: A Video Vision Transformer*. 2021. Disponível em: <https://arxiv.org/abs/2103.15691>.

BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. Disponível em: <https://arxiv.org/abs/1409.0473>.

BRATMAN, M. *Intention, Plans, and Practical Reason*. Cambridge: Cambridge, MA: Harvard University Press, 1987.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HERBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESS, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. *Language Models are Few-Shot Learners*. 2020. Disponível em: <https://arxiv.org/abs/2005.14165>.

BÜTEPAGE, J.; KRAGIC, D. *Human-Robot Collaboration: From Psychology to Social Robotics*. 2017. Disponível em: <https://arxiv.org/abs/1705.10146>.

CORDONNIER, J.-B.; LOUKAS, A.; JAGGI, M. *On the Relationship between Self-Attention and Convolutional Layers*. 2020. Disponível em: <https://arxiv.org/abs/1911.03584>.

CYBENKO, G. V. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, v. 2, p. 303–314, 1989. Disponível em: <https://api.semanticscholar.org/CorpusID:3958369>.

DALLEL VINCENT HAVARD, D. B. X. S. M. An industrial human action recogniton dataset in the context of industrial collaborative robotics. In: *IEEE International Conference on Human-Machine Systems ICHMS*. [s.n.], 2020. Disponível em: <https://github.com/vhavard/InHARD>.

DAMEN, D.; DOUGHTY, H.; FARINELLA, G. M.; FIDLER, S.; FURNARI, A.; KAZAKOS, E.; MOLTISANTI, D.; MUNRO, J.; PERRETT, T.; PRICE, W.; WRAY, M. Scaling egocentric vision: The epic-kitchens dataset. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 720–736.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Disponível em: <https://arxiv.org/abs/1810.04805>.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. Disponível em: <https://arxiv.org/abs/2010.11929>.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011.

FEICHTENHOFER, C. *X3D: Expanding Architectures for Efficient Video Recognition*. 2020. Disponível em: <https://arxiv.org/abs/2004.04730>.

GIRASE, H.; GANG, H.; MALLA, S.; LI, J.; KANEHARA, A.; MANGALAM, K.; CHOI, C. LOKI: long term and key intentions for trajectory prediction. *CoRR*, abs/2108.08236, 2021. Disponível em: <https://arxiv.org/abs/2108.08236>.

GOYAL, R.; KAHOU, S. E.; MICHALSKI, V.; MATERZYNSKA, J.; WESTPHAL, S.; KIM, H.; HAENEL, V.; FRUEND, I.; YIANILOS, P.; MUELLER-FREITAG, M. et al. The "something something" video database for learning and evaluating visual common sense. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. p. 5842–5850.

GRANDINI, M.; BAGLI, E.; VISANI, G. *Metrics for Multi-Class Classification: an Overview*. 2020. Disponível em: <https://arxiv.org/abs/2008.05756>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. *Deep Residual Learning for Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. Disponível em: <https://arxiv.org/abs/1704.04861>.

HUTTER, F.; HOOS, H.; LEYTON-BROWN, K. An efficient approach for assessing hyperparameter importance. In: XING, E. P.; JEBARA, T. (Ed.). *Proceedings of the 31st International Conference on Machine Learning*. Bejing, China: PMLR, 2014. (Proceedings of Machine Learning Research, 1), p. 754–762. Disponível em: <https://proceedings.mlr.press/v32/hutter14.html>.

KARPATHY, A.; TODERICI, G.; SHETTY, S.; LEUNG, T.; SUKTHANKAR, R.; FEI-FEI, L. Large-scale video classification with convolutional neural networks. In: *CVPR*. [S.l.: s.n.], 2014.

KING, G.; ZENG, L. Logistic regression in rare events data. *Political Analysis*, v. 9, p. 137–163, Spring 2001.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>.

KISHIMOTO, M. *Naruto*. Tokyo, Japan: Shueisha, 1999.

KON, Y.; FU, Y. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, v. 130, p. 1366–1401, 2022.

KONDRATYUK, D.; YUAN, L.; LI, Y.; ZHANG, L.; TAN, M.; BROWN, M.; GONG, B. *MoViNets: Mobile Video Networks for Efficient Video Recognition*. 2021. Disponível em: <https://arxiv.org/abs/2103.11511>.

KONG, Y.; TAO, Z.; FU, Y. Deep sequential context networks for action prediction. In: IEEE. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.], 2017. p. 1473–1481.

LECUN, Y.; JACKEL, L. D.; BOSER, B.; DENKER, J. S.; GRAF, H. P.; GUYON, I.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W. Comparison of learning algorithms for handwritten digit recognition. In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. [S.l.: s.n.], 1989.

LIU, L.; JIANG, H.; HE, P.; CHEN, W.; LIU, X.; GAO, J.; HAN, J. *On the Variance of the Adaptive Learning Rate and Beyond*. 2021. Disponível em: <https://arxiv.org/abs/1908.03265>.

LIU, Z.; LIN, Y.; CAO, Y.; HU, H.; WEI, Y.; ZHANG, Z.; LIN, S.; GUO, B. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. Disponível em: <https://arxiv.org/abs/2103.14030>.

LIU, Z.; MAO, H.; WU, C.-Y.; FEICHTENHOFER, C.; DARRELL, T.; XIE, S. *A ConvNet for the 2020s*. 2022. Disponível em: <https://arxiv.org/abs/2201.03545>.

LOSHCHILOV, I.; HUTTER, F. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. Disponível em: <https://arxiv.org/abs/1608.03983>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, Springer, v. 5, p. 115–133, 1943.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, v. 2014, n. 239, p. 2, 2014.

MICCHELLI, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, v. 2, p. 11–22, 1986. Disponível em: <https://api.semanticscholar.org/CorpusID:14461054>.

MLFLOW. *MLflow: A Tool for Managing the Machine Learning Lifecycle*. 2024. <https://mlflow.org/>. Accessed: 2024-09-30.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 9781605589077.

NVIDIA. *NVIDIA GPU Cloud (NGC) Containers*. 2024. <https://catalog.ngc.nvidia.com/>. Accessed: 2024-09-30.

OZAKI, Y.; NOMURA, M.; ONISHI, M. Hyperparameter optimization techniques in machine learning: Overview and features. *IEICE Journal D*, J103-D, n. 9, p. 615–631, 09 2020. ISSN 1881-0225. Early release date: 2020/05/14.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011.

ROBINSON, N.; TIDD, B.; CAMPBELL, D.; KULI, D.; CORKE, P. Robotic vision for human-robot interaction and collaboration: A survey and systematic review. *ACM Transactions on Human-Robot Interaction*, Association for Computing Machinery (ACM), v. 12, n. 1, p. 1–66, fev. 2023. ISSN 2573-9522. Disponível em: <http://dx.doi.org/10.1145/3570731>.

RUSSELL, S.; NORVIG, P. *Inteligência Artificial - Uma Abordagem Moderna*. GEN LTC, 2022. ISBN 9788595158870. Disponível em: <https://books.google.com.br/books?id=5Xf0zwEACAAJ>.

RYOO, M. S.; AGGARWAL, J. K. *UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)*. 2010. "http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html".

RYOO, M. S.; FUCHS, T. J.; XIA, L.; AGGARWAL, J. K.; MATTHIES, L. Robot-centric activity prediction from first-person videos: What will they do to me? In: *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. [S.l.: s.n.], 2015. p. 295–302.

SANFORD, C.; HSU, D.; TELGARSKY, M. *Representational Strengths and Limitations of Transformers*. 2023. Disponível em: <https://arxiv.org/abs/2306.02896>.

SCHOLTZ, J. Theory and evaluation of human robot interactions. In: *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. [S.l.: s.n.], 2003. p. 10 pp.–.

SHOEYBI, M.; PATWARY, M.; PURI, R.; LEGRESLEY, P.; CASPER, J.; CATANZARO, B. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2020. Disponível em: <https://arxiv.org/abs/1909.08053>.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1409.1556>.

STERGIOU, A.; DAMEN, D. The wisdom of crowds: Temporal progressive attention for early action prediction. In: *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2023.

STERGIOU, A.; POPPE, R. Adapool: Exponential adaptive pooling for information-retaining downsampling. *IEEE Transactions on Image Processing*, v. 32, p. 251–266, 2023.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. *Rethinking the Inception Architecture for Computer Vision*. 2015. Disponível em: <https://arxiv.org/abs/1512.00567>.

TOUVRON, H.; CORD, M.; DOUZE, M.; MASSA, F.; SABLAYROLLES, A.; JéGOU, H. *Training data-efficient image transformers distillation through attention*. 2021. Disponível em: <https://arxiv.org/abs/2012.12877>.

TRAN, D.; BOURDEV, L.; FERGUS, R.; TORRESANI, L.; PALURI, M. *Learning Spatiotemporal Features with 3D Convolutional Networks*. 2015. Disponível em: <https://arxiv.org/abs/1412.0767>.

TULI, T. B.; PATEL, V. M.; MANNS, M. *HARNets: Human Activity Recognition Networks Based on Python Programming Language*. Zenodo, 2022. Disponível em: <https://doi.org/10.5281/zenodo.6366665>.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; van der Walt, S. J.; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VanderPlas, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; van Mulbregt, P.; SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020.

WANG, L.; HUANG, B.; ZHAO, Z.; TONG, Z.; HE, Y.; WANG, Y.; WANG, Y.; QIAO, Y. *VideoMAE V2: Scaling Video Masked Autoencoders with Dual Masking*. 2023. Disponível em: <https://arxiv.org/abs/2303.16727>.

WANG, S.; XIE, Z.; LI, Y.; LIN, D.; LUO, P. Videomae v2: Scaling video masked autoencoders with dual masking. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2023.

WANG, W.; CHANG, F.; ZHANG, J.; YAN, R.; LIU, C.; WANG, B.; SHOU, M. Z. Magi-net: Meta negative network for early activity prediction. *IEEE Transactions on Image Processing*, v. 32, p. 3254–3265, 2023.

WANG, Y.; LI, K.; LI, X.; YU, J.; HE, Y.; WANG, C.; CHEN, G.; PEI, B.; YAN, Z.; ZHENG, R.; XU, J.; WANG, Z.; SHI, Y.; JIANG, T.; LI, S.; ZHANG, H.; HUANG, Y.; QIAO, Y.; WANG, Y.; WANG, L. *InternVideo2: Scaling Foundation Models for Multimodal Video Understanding*. 2024. Disponível em: <https://arxiv.org/abs/2403.15377>.

XIONG, R.; YANG, Y.; HE, D.; ZHENG, K.; ZHENG, S.; XING, C.; ZHANG, H.; LAN, Y.; WANG, L.; LIU, T.-Y. *On Layer Normalization in the Transformer Architecture*. 2020. Disponível em: <https://arxiv.org/abs/2002.04745>.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. *Dive into Deep Learning*. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>.

ZHANG, W.; TANIDA, J.; ITOH, K.; ICHIOKA, Y. Shift-invariant pattern recognition neural network and its optical architecture. In: *Proceedings of Annual Conference of the Japan Society of Applied Physics*. [S.l.: s.n.], 1988.

Annex

ANNEX A – Meta-actions and actions from the InHARD dataset

ID	Meta action label	Action label
0	No action	No action
1	Consult sheets	Consult sheets
2	Turn sheets	Turn sheets
3	Take screwdriver	Take screwdriver
4	Put down screwdriver	Put down screwdriver
5	Picking in front	Catch Profile P040
		Catch Fixation FIXA1
		Catch Fixation FIXA2
		Catch Pillow block bearing PAL4019
6	Picking left	Catch Fixation FIXT
		Catch Fixation FIXL
		Catch Cover CAPO
		Catch Nut ECR8
		Catch Bolt B835
		Catch Bolt B840
		Catch Bolt B820
		Catch Fixture key LARD
		Catch Bolt B820PT
		Catch Strap clamps BRIT
7	Take measuring rod	Take measuring rod
8	Put down measuring rod	Put down measuring rod
9	Take component	Catch Fixation FIXT
		Catch Fixation FIXL
		Catch Cover CAPO
		Catch Nut ECR8
		Catch Bolt B835
		Catch Bolt B840
		Catch Bolt B820
		Catch Fixture key LARD
		Catch Bolt B820PT

		Catch Strap clamps BRIT
10	Put down component	Put down Fixation FIXT
		Put down Fixation FIXL
		Put down Cover CAPO
		Put down Nut ECR8
		Put down Bolt B835
		Put down Bolt B840
		Put down Bolt B820
		Put down Fixture key RD
		Put down Bolt B820PT
		Put down Strap clamps BRIT
		Put down Fixation FIXT
		Put down Fixation FIXL
		Put down Cover CAPO
		Put down Nut ECR8
		Put down Bolt B835
		Put down Bolt B840
		Put down Bolt B820
		Put down Fixture key RD
		Put down Bolt B820PT
		Put down Strap clamps BRIT
11	Assemble system	Place LARD on Profile P360-1
		Place FIXA1 on LARD at 160mm
		Screw FIXA1 with B820
		Place LARD on P360-1
		Place CAPO on P360-1
		Place BRIT1 and BRIT2 on P360-2
		Place FIXL on P360-2
		Place FIXA2 on P360-2
		Screw FIXL with B820PT
		Screw FIXA2 with B820
		Place BRIT1 and BRIT2 on P040
		Place P040 on P360-2
		Screw P040 with B820PT
		Place LARD on P040 (P360-2)
		Place FIXA1 on P360-2
		Place FIXA2 on P360-2

		Place DI2T on P360-2
		Place ECR8 on DI2T
		Screw P360-2 with B835
12	Take subsystem	Put down Profile P360-1
		Put down Lower Part
13	Put down subsystem	Catch P360-1
		Catch P360-2

Source: DALLEL Vincent HAVARD (2020).