Understanding Humans in Videos

From Pose to Action Recognition

Università
di Genova

Federico Figari Tomenotti

Supervisor: Prof. Nicoletta Noceti

Dipartimento di Informatica, Biongegneria, Robotica e Ingegneria dei Sistemi - DIBRIS
Università degli Studi di Genova

This dissertation is submitted for the degree of
Doctor of Philosophy in Computer Science and Systems Engineering

Università degli Studi di Genova July 2024

Ph.D. Thesis in Computer Science and Systems Engineering (S.S.D. INF/01)
Thesis Series DIBRIS-xx-2024-xx

Computer Science Curriculum
Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi (DIBRIS)
Università di Genova

Candidate
Federico Figari Tomenotti
federico.figaritomenotti@edu.unige.it

Title
Understanding Humans in Videos: From Pose to Action Recognition

Supervisor
Nicoletta Noceti
DIBRIS, Università di Genova
nicoletta.noceti@unige.it

External Reviewers
Alessandra Sciutti
Center for Human Technologies - CONTACT Unit
Istituto Italiano di Tecnologia

Liliana Lo Presti
Dipartimento di Ingegneria
Università Degli Studi di Palermo
liliana.lopresti@unipa.it

Location
DIBRIS, Università di Genova
Via Dodecaneso, 35
I-16145 Genova, Italy

Ad Alessandra che mi sostiene sempre e mi dona il sorriso.

A Francesco che mi sopporta ogni giorno e a cui voglio un mondo di bene.

A mamma e papà che mi hanno accompagnato fin'ora.

“Many that live deserve death and some that die deserve life. Can you give it to them? Then do not be so eager to deal out death in judgement. For even the wise cannot see all ends.”

J.R.R. Tolkien

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Federico Figari Tomenotti
July 2024

Acknowledgements

Innanzitutto vorrei ringraziare la mia supervisor Nicoletta per aver creduto in me fin dall'inizio e avermi dato lo sprone nei momenti in cui sembrava tutto particolarmente buio o difficile.

Ringrazio particolarmente i miei colleghi con i quali in questi anni abbiamo condiviso tanti momenti insieme e ci siamo confrontati e scambiati idee. Sarebbero troppi per menzionarli tutti e non vorrei dimenticare nessuno, quindi per tutti menziono solo i due che più mi hanno sopportato negli ultimi due anni: Larbi e Jacopo.

Infine ringrazio gli amici che mi supportano sempre e mi ricordano che la bellezza della Vita sta specialmente nelle relazioni e nelle risate in compagnia.

Abstract

This doctoral dissertation focuses on designing, developing, and evaluating methodologies for representing and understanding human motion and humans in the scene (interactions with objects or in small groups). The motivation arises from the growing interest in vision-based solutions which can understand and anticipate human behaviours, not only for robotics but also for surveillance or rehabilitation applications. Associated with specific objectives of this thesis, three distinct methodologies are presented: *MOSAIC* which focuses on hierarchical motion representation for action recognition; *ACROSS* which leverages the complexity of activity recognition using scene-graphs representations; *HHP-Net* which is a Head Pose Estimation Network with a focus on interpretability and computational costs. Experiments and discussions support all the presented methods to highlight their strengths and weaknesses and are assessed using state-of-the-art benchmark datasets.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CGU Confidence Gated Unit

CNN Convolutional Neural Network

GCN Graph Convolutional Network

GNN Graph Neural Network

GRU Gated Recurrent Units

HPE Head Pose Estimation

HPP-Net The name of our HPE method

HRI Human Robot Interaction

KL Kullback–Leibler (Divergence)

LAEO Looking at Each Other

LSTM Long short-term memory

MLP Multi-Layer Perceptrons

NN Neural Networks

RCNN Recurrent Convolutional Neural Network

Relu Rectified Linear Unit

RNN Recurrent neural networks

VAE Variational Autoencoder

Chapter 1

Introduction

1.1 Context and Motivations

Computer Vision applications are becoming ubiquitous and increasingly disruptive thanks to machine learning techniques and the widespread access to computational resources. In this fast-paced evolving landscape, there is enhanced attention on human-centric solutions across various domains and for different purposes. For example, in the medical and rehabilitation domain algorithms are emerging to help the diagnosis of some diseases Garello et al., 2021 and to assist patients in the rehabilitation process Debnath et al., 2022. In the surveillance domain, many systems already rely on computer vision approaches based on neural networks Şengönül et al., 2023, and also public authorities are using them. Furthermore, there exists a considerable demand for computational methodologies in robotics applications, spanning various domains such as factories Othman and E. Yang, 2023, industries Y. Yang et al., 2023, and human-robot interaction (HRI) K. M. Lee et al., 2023; Robinson et al., 2023. In robotics applications, it is imperative to possess the capability to discern the presence of individuals and their motions, thereby enabling safe and effective interaction between intelligent machines and humans.

In all these contexts the human is the focus of attention. In other words, in each of the mentioned applications, and probably in additional domains, the detection of human presence along with the comprehension of their behaviours and intentions stand as pivotal objectives

This thesis aims to design, develop and assess methodologies for the representation and understanding of humans in a scene. We are interested in scenarios where the focus of attention is on single subjects or small groups involved in social interactions.

We present three different methodologies, with specific peculiarities although overall they can be seen as different pieces of the same puzzle. The first methodology, called MOSAIC, is *fully motion-based*: it focuses on the use of a hierarchical motion representation exploited for

action recognition as a downstream task. The second one considers *humans in the scene* and it is called ACROSS; it is a recognition pipeline with a different flavour from the previous one; here, the relationships between the person and the environment (location, objects, etc) are used by the algorithm to learn to classify and anticipate longer activities. The third and last is focused on *Humans in groups*: we present an algorithm for Head Pose Estimation, a solution targeting the human understanding problem under a social interaction perspective. It addresses the problem of human behaviour when multiple people are present in the scene.

In the presented methodologies we are not only interested in performances but we balance effectiveness, computation and interpretability. Regarding the computational aspects, we investigate the possibility of reducing network sizes in order to reduce the economic and environmental impact. On the interpretability of the solutions, when possible, we prefer modular approaches to black-box architectures. Modularity improves interpretability because it helps to understand what can happen in the middle of the computations and has intermediate data representations. In other cases, we help the interpretation of the results by using the notion of confidence in our predictions.

1.2 Objectives

The main objectives of the thesis may be summarised as follows:

- The First one is the study of the human in its social context, to model their interactions with other people. A key to understanding social interaction is gaze, which we approximated with head pose direction. The objective is to develop a light and flexible algorithm to retrieve the head pose starting from keypoints data. We demonstrate its efficacy and applicability by deploying it onto a test bed task to detect dyadic interactions. As a result of this work, the following article has been published:"HHP-Net: A light Heteroscedastic neural network for Head Pose estimation with uncertainty" Cantarini et al., 2022, and "Head pose estimation with uncertainty and an application to dyadic interaction detection" Figari Tomenotti et al., 2024.

- The second one is the study of human motion from sparse inputs in the form of body keypoints. The goal is to learn methods to represent human motion and characterise it in a machine-understandable way. The choices we have made take inspiration from cognitive science for the data representation part, using primitives of motions. These are encoded in a latent space of a neural network which preserves similarity between input data. The outcome of this work is a submitted article to ECCV and we are working towards a journal extension.

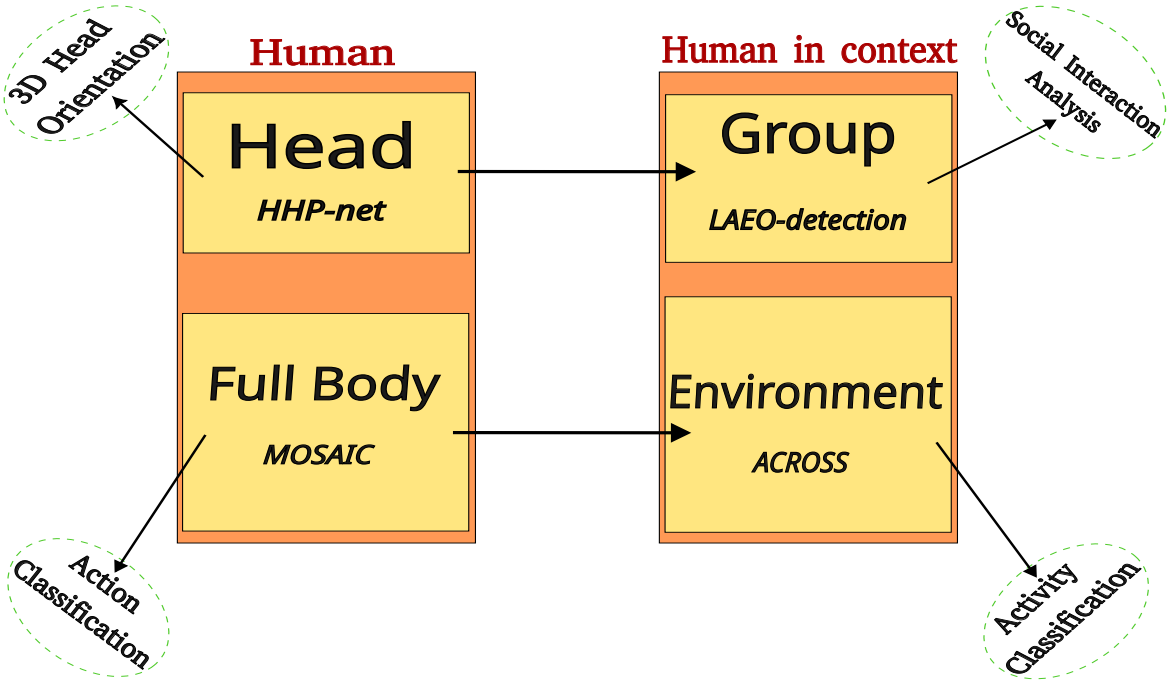


Fig. 1.1 The orange boxes are the two macro-areas of research, and the yellow define the four contributions of this thesis. The four ellipses on the corners identify the problems solved, one for each yellow box.

- The second one is the study of human action recognition in a general setting where we can derive info about people and their surrounding environment from RGB videos. In this task, we devise a model capturing subject-object relationships and add the temporal evolution of this structure which we modelled as a graph. The aim is to develop an algorithm able to anticipate human activity goals by exploiting the connections between humans and scene elements.

1.3 Structure of the Thesis

The work encompasses two main parts where the contributions of this thesis unfold, plus an introduction devoted to background material useful for general comprehension. The objectives presented in the previous section are reported following their temporal occurrences in the PhD path, while also considering the evolution of the entire project, transitioning from specific details to broader conceptual frameworks. Indeed, the project commenced with a specific focus on the head pose, subsequently expanded towards the representation of full-body motion and ultimately culminated in the integration of environmental contextual cues alongside human body motion. To have a clearer understanding of how the different parts

unfold and interconnect we can have a look at the Fig. 1.1. The two orange boxes are the two main topics in which we move, and inside we divided the work into two sub-parts each; the yellow boxes are the four main contributions of the thesis. Each one of the four contributions is associated with one well-established problem in the community. The contributions of this work are mainly in two directions: understanding human motion per se and understanding humans in context. The thesis is structured to present these two topics in an across-the-board way. The first part is focused on retrieving the head orientation and using it as a cue for social interaction analysis. In contrast, the second is devoted to the whole human, so human motion understanding takes into account the complete body and the person in the environment.

- *The Background* chapter is devoted to presenting some theoretical foundations and explaining background notions useful to understand and contextualise the rest of this work.
- *Part I* addresses the Head Pose Estimation and Social Interaction. It includes 3 chapters:
  - *Chapter 3* is the literature review of Part I and it serves as a comprehensive introduction and state-of-the-art exploration of Head Pose Estimation and its applications. It encompasses a multifaceted examination of Social Interactions Analysis under a computer vision perspective, Gaze understanding and Head Pose Estimation methodologies.
  - *Chapter 4* introduces our algorithm for Head Pose Estimation and the LAEO algorithm, which uses the head pose to understand if people in the scene are looking at each other.
  - *Chapter 5* reports the experimental analysis we performed to assess our approach. We first discuss in detail the implementation, the datasets and the experimental protocols we adopted, and then provide qualitative and quantitative results.
- *Part II* unfolds in two subparts with a common fil rouge, which is motion recognition. It includes 3 chapters:
  - *Chapter 6* presents the literature and recent advances in action classification through skeleton data and the usage of graph neural networks for action classification.
  - *Chapter 7* is methodological and it is two-folded: the first half presents the work about human motion representation from primitives of motion (MOSAIC architecture); the second half presents the activity recognition pipeline –ACROSS– focused on human-object interactions.

  - *Chapter 8* encompasses all the experiments to validate the presented methodology and it has been structured in two main parts, the first for the MOSAIC algorithm and the other devoted to the ACROSS algorithm.

- *Chapter 9* is where conclusions are drawn.

Chapter 2

Background

This introductory chapter is devolved to present some theoretical foundations and to explain background notions useful to understand and contextualise the rest of this work. The chapter is organised starting with a general introduction to Deep Learning and progressively presenting some architectures and their usage; the topics are the following:

- Deep Learning introduction
- Convolutional Neural Networks – (*used almost anywhere in Pose Estimation Networks so are a general background for this manuscript*)
- Heteroscedastic Neural Networks – (*are used throughout Part I and in particular in Chapter 4*)
- Autoencoders – (*are used in Part II, especially in the MOSAIC architectures Chapter 7*)
- Models for sequential data
- Recurrent Neural Networks
- Long-Short Term Memory Networks – (*are used in MOSAIC preliminary experiments in Section 8.1.1*)
- Transformers – (*are a main building block in MOSAIC in Chapter 7.*)
- Graph Neural Networks – (*are used in Section 7.2 as a foundation building block.*)

The chapter ends with an explanation of Human Pose Detection, which is considered background knowledge as it is an input in the presented methods. Not all the sections in this chapter are equally explained in detail because mainly only details needed for future comprehension have been included.

2.1 Deep Learning Introduction

Neural networks represent a powerful paradigm in machine learning and artificial intelligence, drawing inspiration from the structure and function of the human brain. These computational models consist of interconnected nodes, or "neurons," organized in layers. Each neuron processes information and transmits signals to neurons in subsequent layers, ultimately enabling the network to learn complex patterns and make predictions.

At the core of a neural network is the neuron, which mimics the behaviour of biological neurons. Each neuron receives input signals, applies a transformation function, and produces an output signal. The strength of connections between neurons, known as "weights," determines the impact of input signals on neuron activation.

Activation functions introduce non-linearity into the neural network, allowing it to learn complex relationships between inputs and outputs.

In the following there is a brief and more formal introduction to different types of networks which will be used in the following dissertation.

2.1.1 Feedforward Neural Networks

Feedforward neural networks (NN), or multilayer perceptrons (MLPs) are the simplest kind of deep learning models. The feedforward networks are used to approximate functions f^*. So a feedforward network defines a mapping $y = f(x, W)$, learning the set of parameters W resulting in the best function approximation.

These models are usually composed of many simpler functions which are stacked and connected together by a direct and acyclic graph, each function is called a layer and the number of layers determines the depth of the model. The first layer is called the Input layer, the final layer is called the Output layer and all the others in between are the Hidden layers (see Fig. 2.1).

Since the input data flows through the layers and there are no feedback connections in which outputs of the model are fed back in input, such models are called feedforward.

A layer can be thought of as the composition of many units (or neurons) that act in parallel, each one receives input from many other units in the previous layer and computes its own scalar value. A hidden unit can be described by a linear function with a bias term b, a vector w of d elements, where d is the length of the input vector x. Hence the computation of a unit is given by:

$$y = \sum_{i=1}^{u} x_i w_i + b \tag{2.1}$$

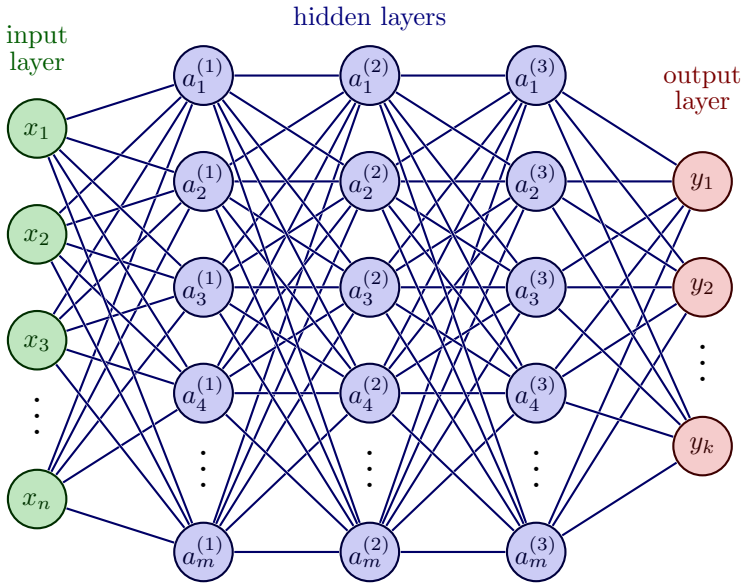


Fig. 2.1 Representation of a Neural Network with 3 hidden layers and k outputs. The units have numbers indicating their position in the network and the connections indicate the information propagation; each connection can be associated with a different weight.

To allow a non-linear parameterization we can introduce an activation function σ to be applied to the output of the unit:

$$y = \sigma\left(\sum_{i=1}^{u} x_i w_i + b\right) \tag{2.2}$$

Since each hidden unit of the same layer is associated with the same activation function σ and given the $u \times d$ weight matrix W, where u is the number of units in the layer, and b is a vector of length u, the output of a hidden layer can be represented as:

$$\Phi(x) = \sigma\left(Wx + b\right), \ \Phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^u \tag{2.3}$$

As previously said a feedforward network is a chain of hidden layers, so we can write the output of a L layer network as

$$\Phi_L(x) \circ \ldots \circ \Phi_1(x) = \sigma_L\left(W_L \ldots \left(\sigma_1\left(W_1 x + b_1\right)\right) + b_L\right) \tag{2.4}$$

Since MLP is a composition of *fully connected* layers (i.e. each unit of a layer is connected with all the units of the following layer), adding a few layers to the network increase dramatically the number of parameters to learn.

A feedforward neural network can be seen as an approach to learning data representation (i.e a features vector), thus the output layer should provide some additional transformation from the features to complete the task that the network must perform.

There are also several theorems about the usage of Neural Networks as universal Approximators, so capable of approximating any function f^* with arbitrary precision. This specifically holds for the Perceptron network with a single infinitely wide hidden layer which can approximate arbitrary functions. These results proceed from the seminal work Cybenko, 1989 by George Cybenko.

2.1.1.1 Activation functions

As previously mentioned, the activation functions allow to include in the model a non-linear parametrization. The most common activation functions σ are :

Linear Assuming the hidden unit represents the linear function $y = \sum_{i=1}^{u} x_i w_i + b$ the linear activation function is the identity function

$$\sigma(x) = x$$

With such function we are assuming the task can be modelled with a linear function. A traditional MLP uses such a σ function.

Relu or Rectified Linear Unit

$$\sigma(x) = max(0,x) \;\; x \in \mathbb{R}$$

It is widely used in computer vision tasks and placed in the hidden layers of the network, it is not differentiable in zero but differentiable anywhere else and the value of the derivative at zero can be arbitrarily chosen to be 0 or 1. The negative inputs are lost and if the weights of the unit is set to zero, it becomes stuck in a perpetually inactive state and such a phenomenon is known as **dying ReLU problem**.

Leaky Relu Mitigate the dying ReLU problem by assigning a small slope to the negative values so that the data are not completely lost.

$$\sigma(x) = \begin{cases} \alpha x, & \text{if } n < 0 \\ x, & \text{if } n \geq 0 \end{cases}, \alpha \in [0, +\text{inf}], x \in \mathbb{R}$$

Sigmoid corresponds to the *logistic function*

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad x \in \mathbb{R}$$

Since it maps the inputs in $[0,1]$ it is normally used as activation of the last layer of the network, in particular for *binary classification* tasks. It may cause problems in the weight update since the function is not zero-centred and the gradient may update the weights with different scales.

Tanh is the scaled version of the Sigmoid, it maps the input in $[-1,1]$ and it is zero-centred

$$\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad x \in \mathbb{R}$$

2.1.1.2 Learning procedure

The goal of training a NN is to find the set of parameters W that let our network approximate a target function f^*. Neural networks are usually trained to solve a minimisation problem, defining an appropriate cost function for the task, and trying to minimise it. The minimisation procedure is done by doing a forward propagation along the network of out input data x, so retrieving a value $\hat{y}$ and calculating the cost of estimating $\hat{y}$ instead of the real value y. Having this cost to solve the minimisation problem of

$$\min_{W} \quad J(W) \tag{2.5}$$

Optimisation in deep learning is done in an iterative way computing an estimate of the cost function on a subset of the data and updating the weights according to it. The problem is solved using the Gradient Descent algorithm (or all its variants: Stochastic Gradient Descent, Adam etc) Tian et al., 2023. The weights update is done by computing the partial derivatives of the cost function with respect to the weights $\nabla_W J(W)$. In modern optimisation algorithms this is done in a mini-batch fashion: computing it on a subset of the dataset. When all the dataset has passed through the network an epoch is passed. The learning usually encompasses many epochs to converge to a good solution. To examine in depth the topic we suggest having a look at Goodfellow et al., 2016; Tian et al., 2023; A. Zhang et al., 2021

2.1.1.3 Addressing Classification using NNs

When dealing with a classification task, we would like to represent the probability distribution over n different classes, this helps in calculating the cost function for the learning procedure Section 2.1.1.2: first, a linear layer predicts unnormalised log probabilities

$$z = Wx + b \tag{2.6}$$

Next, it uses the Softmax activation function $\sigma : \mathbb{R}^n \mapsto [0,1]^n$ to convert z into a probability:

$$\sigma(z)_i = \frac{exp(z_i)}{\sum_j^n exp(z_j)} \tag{2.7}$$

It applies the exponential function to each element z_i of the input vector z and normalizes these values by dividing by the sum of all these exponential, it guarantees that the sum of the components of the output vector is 1.

The z_i variables defining such a distribution over variables are called logits.

In case we are resolving a classification task we can use the point-wise **Cross entropy loss** defined as

$$H(y_x, p(x)) = -\sum_j^M y_x^j \log\left(p(x)^j\right) \tag{2.8}$$

The formula above computes loss for a multi-classification task with M different classes, where $y_x^j \in \{0,1\}$ is a binary indicator if the class j is the correct classification for the example x, $p(x)^j$ is the softmax probability for the j-th class.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks or CNNs are a specialized kind of neural network for processing data that has a known grid-like topology, such as images. CNNs use *convolution* in place of general matrix multiplication in at least one of their layers. First, let us define what a *convolution* is in the continuous case:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da \tag{2.9}$$

The first argument x is the input of the convolution, the second argument w is usually called kernel and the result of the operation is named feature map, in this case, all these elements are continuous.

In the discrete version, the input and the kernel are sampled and defined on the time index t which can assume only integer values:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.10}$$

Since in CNN the inputs are multidimensional data, usually two-dimensional images referred to as tensors, we need to use convolutions over more than one axis at a time, so given an input image I and a two-dimensional kernel K the 2D convolution is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.11}$$

Discrete convolution can be viewed as the sum of a matrix multiplications between a kernel matrix K and image patches. Or, the other way around, a sum of all the possible multiplications between the kernel and its superimposition over the image. Such kind of operation brings some advantages like sparse interactions and parameter sharing. Traditional neural networks seen in the previous sections use matrix multiplication to describe the interaction between each input unit and each output unit so that every output unit interacts with every input unit, on the other hand, *sparse interaction* is achieved when CNNs use kernels smaller than the input which both reduces the memory requirements of the model and improves its statistical efficiency because the model needs to learn fewer parameters. Instead with *parameter sharing*, we mean that the same parameter is used by more than one function in the model: in traditional NNs, each weight is used exactly one time when computing the output, instead in CNNs each member of a kernel (a weight) is used at every position of the input. The goal of a NN is to learn these sets of weights which are also called filters.

We can control the size of the output of a convolutional layer with three different parameters:

- **Depth** is the number of filters of the layer and has all the same $K \times K$ shape.
- **Stride** S is the step size the filter slide the input.
- **Padding** P is the number of rows and columns added to the input to enlarge it, it is usually used to allow the filter to be applied to each point of the original input.

Assuming the input to be of size $I \times I$, the shape of the output will be $O \times O$ where

$$O = \frac{I - K + 2P}{S} + 1 \tag{2.12}$$

A typical convolutional layer is made up of three stages. First, several convolutions are applied to the input in parallel resulting in a set of linear activations, then in the second stage,

each element of the set is transformed by an *activation function*. In the third stage, a pooling function transforms the output of the layer.

A pooling function down-samples the input to reduce the dimensions of the feature maps, and to provide invariance to small translations and variations of the input. The dimension is reduced by performing some summary statistics to nearby elements.

2.1.2.1 Modern Convolutional Neural Networks

Modern Convolutional Networks incorporate also other small strategies to overcome problems during training (e.g. vanishing gradients), or increase their performances, such as training with data augmentation or dropout. But, above all modern CNNs exhibit huge size in terms of parameters. To have an idea of some modern architectures we present some of the winners of famous benchmark competitions: AlexNet Krizhevsky et al., 2012 developed in 2021 has 62 Million parameters, VGG Simonyan and Zisserman, 2014b and Inception Szegedy et al., 2015 are both from 2014 and they have respectively 138M and 6.4M parameters; the newest one to present is ResNet He et al., 2016 from 2015 which encompasses a total of 60.3M parameters. So the dimensions are not the only way to dominate the lead, but they are of course a requirement for good performance.

2.1.3 Heteroscedastic Neural Networks

We present now a more advanced topic in NN, a class of networks capable of learning a regression function for the requested task but also incorporating a measure of the level of uncertainty of its own prediction.

Heteroscedastic Neural Networks (HNNs) represent a specialized class of neural networks designed to address the heteroscedasticity in data, where the variance of the target variable is not constant across all levels of the predictor variables. In traditional neural networks, the assumption of homoscedasticity is implicit, meaning that the model assumes a consistent level of uncertainty in its predictions irrespective of the input values. HNNs, on the other hand, introduce a novel approach by explicitly modelling and predicting the variance or uncertainty associated with each prediction. This is achieved by incorporating an additional output layer that provides a separate set of parameters representing the variance of the predicted values. By doing so, HNNs not only offer more accurate point predictions but also enable a more nuanced understanding of the underlying data distribution, especially in situations where the variability in the data is not uniform. This capability has wide-ranging applications, from

financial forecasting to medical diagnosis, where recognizing and quantifying heteroscedastic uncertainty is crucial for making reliable decisions.

Training Heteroscedastic Neural Networks involves optimizing not only for accurate mean predictions but also for accurate variance predictions. This introduces a dual-task learning paradigm, where the network simultaneously learns to model both the mean and variance of the target variable. Various loss functions, such as a combination of mean squared error and variance-related terms, are employed to guide the network during training. The incorporation of heteroscedasticity modelling in neural networks contributes to a more comprehensive understanding of the uncertainty inherent in the data. HNNs provide a valuable tool for applications where the reliability of predictions is paramount, offering a more sophisticated and interpretable approach to uncertainty quantification compared to traditional homoscedastic models. As machine learning continues to be integrated into diverse domains, the adaptability of Heteroscedastic Neural Networks to handle varying levels of uncertainty makes them a promising avenue for improving the robustness of predictive models in real-world scenarios.

2.1.3.1 Uncertainty Estimation

Different types of uncertainty arise in data and they need to be addressed differently. In particular, uncertainty in predictions can arise from various sources, including limited data, measurement errors, or model complexity. Bayesian Neural Networks provide a natural framework for estimating two type of it: epistemic uncertainty and aleatoric uncertainty.

Epistemic Uncertainty: this type of uncertainty arises from a lack of knowledge or data. BNNs model epistemic uncertainty by capturing the uncertainty associated with the model parameters. As more data becomes available, the uncertainty about the parameters decreases.

Aleatoric Uncertainty: this uncertainty is inherent in the data itself and cannot be reduced with additional observations. BNNs capture aleatoric uncertainty by associating a distribution with the output predictions, providing a measure of confidence in the model's predictions.

2.1.3.2 Bayesian vs Heteroscedastic Neural Networks

Bayesian Neural Networks (BNNs) (for further details see Duvenaud, 1999; Jospin et al., 2022) and Heteroscedastic Neural Networks (HNNs) share a common goal of enhancing neural networks' capacity to handle uncertainty, yet they differ in their approaches. Both models acknowledge and address the limitations of traditional neural networks by introducing a more nuanced understanding of uncertainty in predictions. In BNNs, uncertainty is captured through probabilistic modelling of the network parameters, treating them as probability

distributions. On the other hand, HNNs focus on explicitly modelling and predicting the variance of the target variable, acknowledging heteroscedasticity in the data. While BNNs offer a broader framework for uncertainty estimation, encompassing both epistemic and aleatoric uncertainties, HNNs specialize in capturing heteroscedastic uncertainty, which arises from varying levels of variability in the data. The key similarity lies in their departure from the traditional deterministic paradigm, allowing both models to provide richer and more informative predictions. Despite these differences, both BNNs and HNNs contribute to a more robust and trustworthy deployment of machine learning models, each tailored to address specific aspects of uncertainty in diverse applications.

2.2 Autoencoder & Variational Autoencoder

2.2.1 Autoencoder

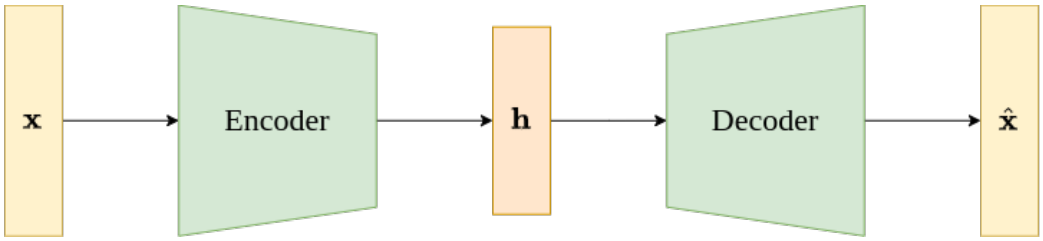


Fig. 2.2 Schematic representation of an Autoencoder network, composed of two sub-networks: an Encoder and Decoder respectively. $\mathbf{x}$ is the input vector, $\hat{\mathbf{x}}$ the reconstructed output. And in the middle, we have a usually compact representation of the input.

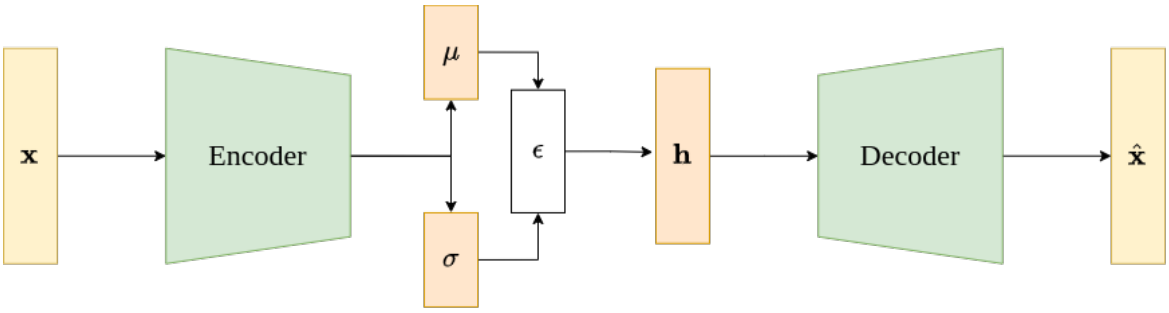


Fig. 2.3 Schematic representation of a Variational Autoencoder network, composed of two sub-networks: an Encoder and Decoder respectively, and a sampling function ε. $\mathbf{x}$ is the input vector, $\hat{\mathbf{x}}$ the reconstructed output. And in the middle we have a compact representation of the input, respecting certain statistics properties.

An autoencoder (see Fig. 2.2) is a neural network that is trained to attempt to copy its input to its output. It has a hidden layer **h** that is -a usually smaller- description of the input. The network is formed by an encoder $f(\cdot)$:

$$\mathbf{h} = f(\boldsymbol{x})$$

and a decoder $g(\cdot)$ which from the latent space retrieves a reconstruction of the input

$$\hat{\mathbf{x}} = g(\mathbf{h})$$

The loss function (or cost function) to be minimised is

$$\min_{W} \quad L(\mathbf{x}, g(f(\mathbf{x}))) \tag{2.13}$$

Where W are the network weights. However the simplest solution to Eq. (2.13) is for the two functions to be equal to the Identity, which is of no utility. Indeed, the autoencoder is forced to learn a non-perfect reconstruction of the input with two different functions. One of the most used cases of autoencoder is to do a dimensionality reduction or feature extraction. One solution to force the networks to learn a non-trivial function it is to use a penalisation term to enforce the sparsity of the representation and having small weights.

$$\min_{W} \quad L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}) \tag{2.14}$$

2.2.2 Variational Autoencoder

A variational autoencoder (see Fig. 2.3) -introduced in Diederik P Kingma and Welling, 2013a- can be assimilated to an Autoencoder for its Encoder-Decoder structure but differs from it for its mathematical structure. The VAE is a probabilistic generative network where the encoder and decoder are function approximators.
We want to find a latent space $\mathbb{Z} = \mathbb{R}^Q$ using a given set of samples $\{y_m\} \subseteq \mathbb{Y} = \mathbb{R}^R$ where $Q << R$. So a dimensionality reduction is requested. We also want to enforce that the samples in the latent space are normally distributed (computing the KL distance over a distribution). So, in short, we have an encoder and a decoder trained with the following loss:

$$\min_{W} \quad L(\mathbf{x}, g(f(\mathbf{x}))) - KL(f(\mathbf{x})||N(0,1) \tag{2.15}$$

The first term is the reconstruction term as in Eq. (2.13), and the second can be seen as a regularisation term. The rest of the sections has some of the passages to come to this equation, but are not an exhaustive mathematical description of VAEs.

Setting the problem in a probability distribution setting, the question is to find a model distribution $q(z)$ to approximate the true posterior distribution $p(z|y)$

$$q(z) = argmin_q \quad KL(q(z)|p(z|y)) \tag{2.16}$$

Where the KL is the Kullback-Leibler divergence (measure on distributions). We can rewrite the divergence to obtain a lower bound on the intractable marginal likelihood $p(y)$.

$$\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log \frac{p_\theta(x,z)}{p_\theta(z|x)}] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log [\frac{p_\theta(x,z)}{q_\phi(z|x)} \times \frac{q_\phi(z|x)}{p_\theta(z|x)}]] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log [\frac{p_\theta(x,z)}{q_\phi(z|x)}]] + D_{KL}(q_\phi(z|x)||p_\theta(z|x))
\end{aligned} \tag{2.17}$$

Evidence Lower Bound

$$\mathbb{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \tag{2.18}$$

Substituting Eq. (2.17) in Eq. (2.18), and rearranging the terms, we can notice that the loss has two terms with opposite signs. One is the reconstruction loss and the second is the KL divergence from the learned distribution to the target prior distribution. In this respect, the similarity with the Autoencoder is in the fact that both of them push the train to learn to reconstruct the input samples, however, the VAE tries also to have a latent space with variables distributed as the target prior: in many cases a Standard Normal distribution $N(0,1)$.

In the real implementation, the sampling is done via the *reparametrisation trick* Diederik P. Kingma and Welling, 2013b

$$z \sim q_{\mu,\sigma}(z) = \mathcal{N}(\mu, \sigma^2)\varepsilon \sim \mathcal{N}(0,1)z = \mu + \varepsilon \cdot \sigma \tag{2.19}$$

2.3 Classical Sequence Modelling

2.3.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data. The recurrence adds memory to the NN and it provides a way to model relationships between data, it means that the samples "seen" by the network at time step $t-1$ affect the computation at time t. Hence RNNs have two inputs: the present data and the recent past data, which are combined to determine how they respond to new data.

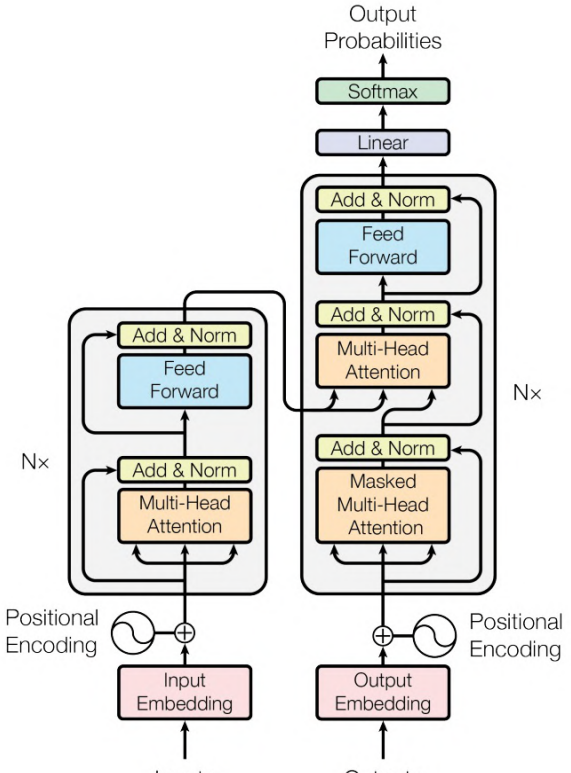
2.3.2 Long Short-Term Memory

Long short-term memory (LSTM) Hochreiter and Schmidhuber, 1997 and Gated Recurrent Units (GRU) K. Cho et al., 2014 are specially designed to find correlations between events separated by many moments, and these correlations are called “long-term dependencies”, in practice they help in learning the correlation while forgetting that information not necessary to the computation of the output. On the other hand, simple RNNs do not have the ability to "forget" leading to instability in the gradient computation.

They have been successfully used in multiple application domains like speech recognition and text-to-speech synthesis, and machine translation. Furthermore, there exist tasks combining LSTM and CNNs, such as *Automatic image captioning* Vinyals et al., 2015 generating textual description from an image.

2.4 Modern Sequence Modelling: Transformers

The Transformer architecture introduced in Vaswani et al., 2017 has taken the lead as state-of-the-art neural network architecture in many tasks. It is well suited to process sequential data, and coversely of Recurrent Neural Networks based on their capacity to model long-term dependencies on the Attention mechanism. It is basically a matrix multiplication operation between inputs and important learned weights. The advantage of the Attention mechanism is that it can be executed multiple times in parallel on the same



inputs, and this gives the flexibility of having more nuances of the same representation for a single input. For these reasons, they are employed in all state-of-the-art models for Natural Language Processing, such as OpenAI's GPT networks.

A transformer is composed of an encoder and a decoder structure, see Fig. 2.4. The encoder receives in input sequential data, and encode them in a fixed-size representation called a token (e.g. one work is a token) and then applies a positional encoding strategy (e.g. usually a multiplication by a time dependant function). Afterwards, all the tokens in parallel are passed through Attention layers. Each one is composed of a self-attention block and a feed-forward block. The former is built of attention heads, where the entire sequence is processed and attention between intra-sequence tokens are highlighted. Self-Attention is a mechanism of computing three vectors: Key, Query, and Values, and thanks to dot products between them, the output is a matrix of weights telling how much each token is in a relationship with another one. The complexity is quadratic in sequence length $O(L^2)$. The latter -the feedforward- applies a non-linear transformation to the attention output. It add expressivity and it makes up two-thirds of the parameters in a transformer model.

The decoder, instead is made up of two building blocks: the self-attention and the encoder-decoder attention block. The former we have already explained, the latter conversely is used to get the context of the sequence and to perform attention computation between the encoder-output with respect to the decoder-output. The final output of a Transformer is a representation of the same length of the input but transformed in another space. Indeed, the language-to-language transformation does exactly this. The transformer training procedure is done using an increasing learning rate and then after a few epochs of warm-up, it starts the decreasing period. The schedulers used for this architecture use often cosine or cyclical functions. After this general overview, we add some more in-depth insights and formal definitions of the different building blocks of the Transformer network.

2.4.1 Attention Mechanism

The attention mechanism is formalised per each head as follows

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.20}$$

where $\sqrt{d_k}$ is the dimension of k and q vector, which are respectively the keys and queries. The K, Q and V are the matrices composed of the vectors k q and v; the V are the values calculated through keys and queries. In Fig. 2.5a we show the operation in detail in a diagram fashion.

The multi-head attention used in the full architecture is the concatenation of many attention's heads, as in Fig. 2.5b. Formally

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{2.21}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.22}$$

2.4.2 Positional Encoding

The positional encoding can be formalised in several different ways which are equivalent from theoretical viewpoints. Often it is implemented as the original formulation of Vaswani et al., 2017 which is:

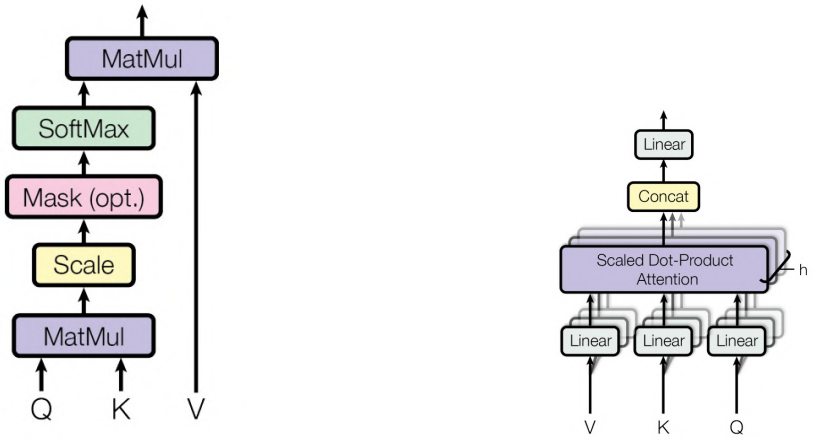
$$PE_{pos,2i} = sin(pos/1000^{2i/d_{model}})PE_{pos,2i+} = cos(pos/1000^{2i/d_{model}}) \tag{2.23}$$

where pos is the position and i is the dimension, so each dimension is have its own sinusoid.

2.5 Graph Neural Networks

Graph Neural Networks (GNNs) represent a cutting-edge class of neural networks designed to handle data structured as graphs. In contrast to traditional deep learning models, which primarily operate on grid-like data such as images or sequences, GNNs excel in processing non-Euclidean data, capturing intricate relationships and dependencies inherent in graph structures.

Graph Neural Networks are a specialized category of neural networks tailored for graph-structured data. They have gained significant attention due to their applicability in various



(a) (b)

Fig. 2.5 In Fig. 2.5a Transformer dot attention scheme. In Fig. 2.5b Transformer multi-head scheme. Both images [source Vaswani et al., 2017]

domains such as social network analysis, recommendation systems, drug discovery, and computational biology. At the core of GNNs lies the ability to learn and propagate information across the nodes and edges of a graph, enabling them to perform tasks such as node classification, graph classification, link prediction, and graph generation.

The input to a Graph Neural Network comprises a graph $G = (V, E)$, where V represents the set of nodes and E represents the set of edges connecting these nodes. Each node in the graph is associated with a feature vector, capturing information relevant to that node. Similarly, each edge may also have associated features representing the relationship between connected nodes. Additionally, the graph may possess a global feature vector encoding overall graph-level information.

The construction of a Graph Neural Network involves several key components:

1. Graph Convolutional Layers (Graph Convolution): These layers are the fundamental building blocks of GNNs. They aggregate information from neighbouring nodes and edges, enabling nodes to learn representations based on their local graph neighbourhood.

2. Message Passing: GNNs leverage message-passing schemes to propagate information between nodes in the graph. At each layer, nodes receive messages from their neighbours, aggregate these messages, and update their own representations accordingly.

3. Pooling Layers: These layers aggregate node-level representations to obtain a graph-level representation. This global representation captures the holistic characteristics of the entire graph.

4. Output Layers: Depending on the task, GNNs may have different output layers such as softmax layers for node classification, sigmoid layers for link prediction, or fully connected layers for graph classification.

GNNs offer a versatile framework for analysing and extracting insights from complex relational data, making them indispensable in domains where data exhibit graph-like structures.

2.5.1 Graph Convolutional Networks

They aim to extend the concept of convolutional operations, traditionally applied in grid-like data such as images, to irregular graph structures. One key feature of GCNs is their ability to learn representations of nodes by aggregating information from their neighbours, similar to how convolutional layers in traditional neural networks aggregate information from local regions in grid-like data. Formally the output of a node in a GCN is calculated by aggregating the features of its neighbours. GCN has the following layer-wise propagation rule Kipf and Welling, 2016:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.24}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the graph G with self-connections added (it is a requirement for the processing). $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is a degree matrix, $W^{(l)}$ is a layer-specific trainable weight matrix, or a kernel,σ is an activation function such as ReLU, $H^{(l)}$ is the data matrix in the l^{th} layer, $H(0) = X$.

2.5.2 Attention Graph Networks

Attention Graph Networks (AGNs) are a variant of Graph Neural Networks (GNNs) that leverage attention mechanisms to selectively focus on relevant nodes or edges within a graph during message passing. Unlike traditional GNNs that typically aggregate information from all neighbouring nodes uniformly, AGNs dynamically compute attention weights for each neighbour based on their relevance to the current node. This allows AGNs to adaptively attend to important nodes or edges while ignoring irrelevant ones, enabling more efficient and effective information propagation across the graph. More formally we can write the update function over the edges as:

$$\alpha_{ij} = Softmax(\sigma(W_a^T.[Wh_i^l \oplus Wh_j^l])) \tag{2.25}$$

where α_{ij} are the edge weights, $W_a^T \in \mathbb{R}^{2d'}$ and $W \subseteq \mathbb{R}^{d' \times d}$ are learned parameters and d is the embedding dimension,$\oplus$ is the concatenation operation. The combined message

aggregation and update steps are a weighted sum over all the neighbours and the node, such as:

$$H_i^{l+1} = \sum_{j \in N_i \cup \{i\}} \alpha_{ij}.WH_j^l \tag{2.26}$$

For a more in depth description have a look at Veličković et al., 2017.

2.6 Human Pose Estimation

Human Pose Estimation is a well-established problem in computer vision which overcomes simple human detection and aim at detecting single points of the human figures, such as mouth, eyes, ears, shoulders and wrists, knees and feet and of course the torso.

Nowadays, many algorithms emerged and we explain in details two of the most used: Openpose, Centernet and AlphaPose. Conversely, we only mention MediPipe for its spread usage, but its implementation details are not public. OpenPose is one of the most well-renowned bottom-up approaches for real-time multi-person body pose estimation. Just like the other bottom-up approaches, Open Pose initially detects parts belonging to every person in the image, known as key points, trailed by allocating those key points to specific individuals. The pipeline is composed of a two-branch CNN (Convolutional Neural Network): one predicts confidence of key-points presence and the other of affinity fields to connect the joints (parts association). The difficult stage is associating keypoints with the correct affinity fields (so the correct person) when multiple people are present and superimposed on the image plane, and it is achieved with a graph technique.

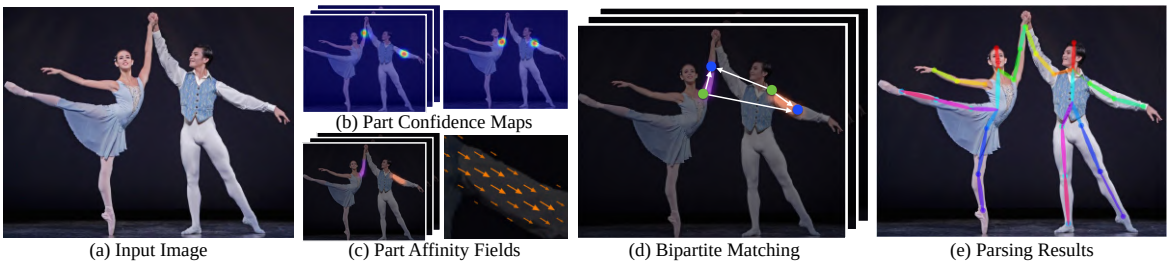


Fig. 2.6 Openpose Pipeline: Starting from the left: the input RGB image, then the part affinity fields and Confidence maps retrieval are presented. Then the matching procedure where keypoints are assigned to the correct skeletons is shown. Finally, the output drawn onto the input image. [source Z. Cao et al., 2019].

CenterNet: This results in a regression-based one-stage multiperson human pose estimator similar to the slow-RCNN. This pipeline utilises a novel heat-map regression approach to

accurately detect human keypoints such as joints and body parts. Unlike traditional methods that rely on complex multi-stage processes, CenterNet simplifies the pipeline by directly regressing the keypoints locations from image features. Through its efficient architecture and effective feature learning, CenterNet Pose Detector achieves impressive performance in real-time applications. And it is, by design, a multi-person approach. Alpha Pose: Alpha-Pose is a well-known top-down technique of pose estimation. The creators of this technique suggest that top-down methods are usually based on the precision of the person detector, as pose estimation is conducted on the area where the person is present. This is why errors in localization and replicate bounding box predictions can result in the pose extraction algorithm working sub-optimally. To solve this issue, the creators introduced a Symmetric Spatial Transformer Network (SSTN) to pull out a high-quality person region from an incorrect bounding box. A Single Person Pose Estimator (SPPE) is applied in this extracted area to estimate the human pose skeleton for that individual. Mask-RCNN: This technique is very similar to the top-down method, but the person detection step is conducted along with the part detection step. Put simply, the keypoint detection phase and the person detection phase are independent of each other.

Part I

Head Pose Estimation and Social Interaction

Part I is focused on the retrieval of the head pose as a cue of social interaction; the second half of the current Part presents an application to a real task, which employs the head pose estimation algorithm highlighting its utility and strength in real domains.

Introduction To Part I

Part I of this work is devoted to explaining motivations and goals for developing a Head Pose Estimation pipeline from keypoints. It pauses on the importance of Head Pose Estimation for many applications of Computer Vision where humans are at the core and main object of study, such as all rehabilitative and video-surveillance scenarios, but also Human-Robot Interfaces. Another focal point of the research is to build upon existing tools, to use consolidated technology to carry on and open to new application scenarios. Indeed, our pipeline relies on Human Pose Detectors and focuses its development on interoperability with several state-of-the-art Estimators. Moreover, special attention is devoted on the consumption -in terms of power and computational resources- of our solution both for environmental concern and also for ubiquitous adoptions also on embedded technology or at the edge of the infrastructure. Indeed, it adds a negligible amount of computation to the one of a Human Pose Estimator. We also present several experiments to asses our proposed solution, to demonstrate its points of strength and to investigate its weaknesses presenting a very broad and complete set of tests. The scientific curiosity has driven the experimental part to be as solid and extended as possible. Furthermore, we tested the Head Pose solution as a component in a second new algorithm we invented and tested. This second algorithm is much simpler than the first one, but it is more application-oriented and serves to understand if two people are looking at each other in a scene. This, we claim, can be beneficial in many social science experimental settings, to study groups and social interactions. The following chapters unfold as follows:

- Introduction and state of the art
- Head Pose Estimation methodology
- Experiments, results and discussions

Chapter 3

Literature Review

This chapter serves as a comprehensive introduction and state-of-the-art exploration of Head Pose Estimation and its applications, delving into various aspects crucial to the understanding and advancement of the research area. It encompasses a multifaceted examination of Social Interactions Analysis under a computer vision perspective, Gaze understanding and Head Pose Estimation methodologies. Through a synthesis of theoretical frameworks, empirical analyses, and methodological approaches, this chapter offers the reader a complete yet non-exhaustible introduction to the topics of this first part of this work.

3.1 Social Interactions

Detecting and recognising social interactions from videos is becoming a relevant topic in Computer Vision and Artificial Intelligence, mainly because a massive amount of human communication and interactions is conveyed through nonverbal cues. In this regard, and on the idea of creating intelligent machines, some scientists claim that "next-generation computing needs to include the essence of social intelligence" (Vinciarelli et al., 2009). Some others (Bolotta and Dumas, 2022) argue that "social interactions not only are largely unexplored in this field but also are an essential element of advanced cognitive ability and therefore constitute metaphorically the dark matter of AI". They also suggest that intelligence comes in time and not over time, implicitly saying that modelling time in computer vision helps to think more like humans (do we want AI to resemble human intelligence? (Cristianini, 2023)).

Following these ideas the research is moving in multiple directions: human-human interaction (both dyadic and group interactions (Corbellini et al., 2022)) and human-machine interactions (Suma, 2019). We could ask whether computer vision -and not other disciplines- should take a step forward in addressing these issues; one strong motivation is expressed in

(McMahon and Isik, 2023) which outlines “the behavioural evidence that social relations in particular are processed visually”. In this respect, computer vision is the enabling technology to push machines to be more in touch to humans. There are a plethora of small tasks that need to be addressed to solve such an ambitious goal. Our brain does everything in a very robust and natural way, with almost no effort, but machines need to learn everything from scratch. Some studies suggest that “systems that explicitly represent social primitives in their input do better than unstructured end-to-end learning systems at detecting interactions”. This approach is the one followed in our work where every task has its own importance and it is a brick to build novel and richer algorithms. In the social interaction domains, we focus on the specific case of dyadic interactions and among them on the interactions conveyed by the gaze.

3.2 Gaze Understanding

Recent literature, supported by social sciences has investigated the role of gaze in the human nonverbal communication realm; in the following, we will have a brief excursus through some of the more recent or important work addressing the topic. We commence by delineating a purely geometric approach proposed in the literature (Soo Park and J. Shi, 2015)(Park et al., 2012). In this method, the gaze is conceptualized as a cone characterized by a Gaussian distribution positioned in front of the subject. In particular, the scene information is captured by head-mounted cameras. The focal interest of the work is the study of the spatio-temporal characteristics of the birth and death of gaze concurrence (useful in human-robot interactions and in the monitoring of neurological and developmental disorders). Another work states that the eye gaze can be approximated by the head orientation $\pm 35^{\circ}$ (Massé et al., 2018), and the method to solve the problem of inferring the gaze and linking it to the interactions is based on State changes and Kalman filters for the state estimation; however the experimental environments is very much controlled.

Probably the first article using Deep Neural Networks to solve the Gaze Direction problem is "Where are they looking?" (Adria Recasens et al., 2015). The problem is formulated as a regression problem: in an image depicting many people detect where each one is focusing their attention, as a (x, y) coordinate on the image plane. (Adria Recasens et al., 2015) proposes also the GazeFollow dataset, which is nowadays a benchmark for this type of application. The dataset has been used in (H. Zhao et al., 2019) by Zhao et al., and it states that biases are present so they propose a new dataset GazeShift. And, their approach to tackling the problem is based on a CNN which projects the gaze direction on a polar plane, and traces it back to the image plane; out-of-frame target are not taken into account in this

method. In (Chong et al., 2018; Chong et al., 2020) we found a CNN which tries to grasp if a person is looking at a target which is inside the frame (or scene) or not, and a novel dataset built on purpose for the task VideoAttentionTarget is presented. Another interesting work explores six different types of interaction using gaze: single, mutual, avert, refer, follow, and share (for more details refer to (L. Fan et al., 2019). The latter six actions are consistently performed by humans, yet proving challenging for machines to discern due to their contextual dependencies, such as location and audio cues. They presented a new dataset Vacation (L. Fan et al., 2019) and a new spatiotemporal graph network to detect and classify these six subtle interactions.

A specific case within the realm of social interaction analysis is that of dyadic interactions involving two individuals. In this context, the initial phase involves identifying pairs of individuals engaged in interaction, commonly achieved through the detection of mutual gaze or "Looking At Each Other" (LAEO) as referenced in the literature (M. J. Marín-Jiménez et al., 2014; L. Fan et al., 2019). In (Manuel Marin-Jimenez and Ferrari, 2011) the TVHID-LAEO dataset has been realised, and afterwards, also the AVA-LAEO and UCO-LAEO have been proposed in (Marin-Jimenez et al., 2019) by the same authors. An early work on LAEO detection in images utilises a Gaussian process predicting yaw and pitch, generating a LAEO score per frame (M. J. Marín-Jiménez et al., 2014). More recently, LAEO-Net and LAEO-Net++, introduced by the same authors, proposed a CNN-based extension to estimate LAEO over temporal windows (Marin-Jimenez et al., 2019; M. Marín-Jiménez et al., 2020). Recent advancements include an end-to-end pipeline based on Transformers for mutual gaze detection (Guo et al., 2022) and a late fusion approach combining head and scene features encoded with variants of a ResNet (F. Chang et al., 2023).

Multimodal approaches have also been explored for mutual gaze detection. In (Trabelsi et al., 2017), authors leverage RGB data with depth information, while (Kukleva et al., 2020) presents an approach integrating vision and text to jointly address interaction detection and long-term relationship prediction. Joint learning of LAEO and 3D gaze estimation is discussed in (Doosti et al., 2021), where each face is detected and some features are extracted using a Neural Network; then all the faces detected are passed to a second network with their relative (3D) positions and features to be classified for LAEO frames.

In comparison to existing methods, our LAEO method relies solely on head orientation as a proxy for mutual gaze, which introduces limitations concerning the usage of the proper gaze. However, it offers notable advantages, such as effectiveness even when subjects are positioned at a distance from the camera, expanding its applicability to diverse real-world scenarios. Moreover, our presented approach is engineered to be effective but simple enough to be placed in cascade to our HHP-net without compromising its speed and lightweight

nature. Our HHP-net is fast and very lightweight, so the usage of a deep network as in (Guo et al., 2022) was not an option for us.

3.3 Human Pose Estimation

Human Pose Estimation, aiming to extract the semantics and topology of the human body from images, finds applications in several domains, including human motion analysis in sports (Colyer et al., 2018) and medicine (Moro et al., 2022), action recognition (Luvizon et al., 2020; L. Shi et al., 2019), human-machine and social interaction analysis (Luvizon et al., 2020; Song et al., 2021), biometric recognition (Barra et al., 2020) and driver attention detection (Campbell, 2012; J. Wang et al., 2021).

Comprehensive studies (Gong et al., 2016; Zheng et al., 2023; J. Wang et al., 2021) analyse differences in approaches like 2D (Z. Cao et al., 2019; K. Duan et al., 2019; Bazarevsky et al., 2020) versus 3D (J. Yu et al., 2017; H. Zhou et al., 2021), handcrafted features versus deep learning, and single-person versus multi-person scenarios. The focus here is on 2D pose estimation from monocular images, where we may identify bottom-up algorithms like Openpose (Z. Cao et al., 2019) and top-down approaches like AlphaPose (Fang et al., 2022). Regarding computational performances, one of the fastest and most recent methods is in the MediaPipe framework (Lugaresi et al., 2019), employing a combined heatmap, offset, and regression approach (Bazarevsky et al., 2020)). Distinctions in algorithms include the number of key points and topology, with 'standards' like COCO format (Lin et al., 2014) with 17 key points, OpenPose (Z. Cao et al., 2019) with 25 key points, and MediaPipe encompassing 33 key points. More complex approaches aim to incorporate spatial and appearance consistency (W. Yang et al., 2016) and video-based methods (Luo et al., 2018), but are out of the scope of this work.

3.4 Head Pose Estimation

Head pose estimation has been addressed by a number of relatively recent methodologies (Zhiwen Cao et al., 2021a; Y. Zhou and Gregson, 2020; Madrigal and Lerasle, 2020; H. Zhang et al., 2020; Hai Liu et al., 2021; Dhingra, 2022; Hai Liu et al., 2022), with classical applications to Human-Machine Interaction or to social interaction analysis. The more recent and comprehensive survey on the topic is probably (Abate et al., 2022).
Some methods use additional information such as depth (Gabriele Fanelli et al., 2011; Mukherjee and Robertson, 2015; Hong et al., 2018) or time (J. Gu et al., 2017), but also points clouds as in (Xu et al., 2022) or infrared as in (T. Liu et al., 2021). In the field of

driver-assistive technology and safety, infrared cameras are used to estimate the head pose(Ju et al., 2022), but with ad-hoc solutions due to the camera setup (the camera is commonly located in the centre of the rear-view mirror of the car). Differently from these approaches which use different sensors, in our work, we only employ RGB images, which guarantee less expensive and more general applications.

In alternative methodologies, the head pose is derived by fitting an image onto a 3D face model, or on some approximation of it. An estimation of a 3D model is first presented in (G. Fanelli et al., 2013), while more recently deep learning-based methods have been presented: such as 3DDFA (X. Zhu et al., 2019), a CNN able to fit a 3D model to an RGB image, or SADRNet (Ruan et al., 2021) which specifically tackles the problem of face occlusions. Furthermore, FAN (Bulat and Tzimiropoulos, 2017) is a state-of-the-art facial landmark detection method, that performs also face alignment. These approaches propose complex computational pipelines and have demonstrated the potential to yield notably accurate results.

One of the most recent challenges is in estimating pose directly from individual 2D images. In this respect, we start by mentioning a different but related task of estimating the 2D gaze: GazeFollow (A. Recasens et al., 2015) is a two-pathway CNN architecture that estimates the apparent direction of the human gaze and the object being observed; it combines saliency maps of the whole image with the position of subjects' head to obtain a pose prediction. A very efficient strategy to estimate the apparent direction of gaze is proposed in (Dias et al., 2020).

Moving to 3D head pose estimation, nowadays it is mostly obtained by deep learning architectures that start from the output of face detectors, often implemented as a Convolutional Neural Network (CNN). Besides recent few exceptions such as (Bisogni et al., 2021), the literature presents several works starting from images: (Shao et al., 2019) propose an adjustment of the ROI obtained by face detection (it incorporates an offset around the face) and a combined regression and classification loss. HopeNet is a regression method with ResNet and a joint MSE and cross-entropy loss (Ruiz et al., 2018). LwPosr (Dhingra, 2022) introduces a lightweight architecture based on a mixture of depthwise separable convolutional and transformer encoder layers, structured in two streams and three stages to provide fine-grained regression. Transformers have also been used in (Hai Liu et al., 2023) which specifically addresses challenges related to occlusions, illuminations, and extreme orientations. FSA-net (T.-Y. Yang et al., 2019) is a two-stream multi-dimensional regression network able to provide accurate fine-grained estimations. (Rahmaniar et al., 2022) presents an approach based on a combination of coarse and fine feature map classification to train a multi-loss CNN architecture. CNNs are also used in (Hsu et al., 2018), where an L2 regression loss and an ordinal regression loss are jointly employed, and in (Albiero et al., 2021), which regresses

6DoF pose in a Faster R-CNN–based framework.

In contrast to these methods, our 3D head pose estimation pipeline relies exclusively on the output of a human pose detector, similar to the strategy proposed in (Dias et al., 2020). This allows us to design a very lightweight architecture, capable of attaining accurate results, acting sequentially to a 2D pose estimator. Our approach also differs from multi-task approaches such as KEPLER (Kumar et al., 2017) – predicting facial key points and pose–, Hyperface (Ranjan et al., 2019) – simultaneously performing face and landmark detection, pose estimation and gender recognition –, or the method proposed in (Xia et al., 2022) – jointly learning Head Pose Estimation, face alignment and face tracking. Concerning these methodologies, our approach prioritises modularity, offering the flexibility to seamlessly integrate with various pose estimators.

As recently observed in (Y. Zhou and Gregson, 2020), head pose estimation is intrinsically harder on certain viewpoints. Starting from this observation, in (Ruiz et al., 2018) an approach for improving on lateral views is proposed, to obtain wide-range head pose estimation. Instead, our work follows the observation in (Dias et al., 2020): certain viewpoints are associated with different levels of uncertainty, creating a large discrepancy in accuracy. This can be formalised with the concept of aleatoric heteroscedastic uncertainty (Kendall and Gal, 2017), which depends on the inputs and may be estimated from data. Conventional deep learning methods cannot estimate the uncertainty of their inputs, consequently, Bayesian deep learning is becoming very popular as an effective approach to address this limitation. In our method we propose a multi-task approach where a task is associated with one of the three pose angles, extending (Kendall and Gal, 2017) to the multi-loss case. Indeed (Cipolla et al., 2018) reports a loss with homoscedastic uncertainty, also called task-dependent uncertainty, that is constant across different inputs. In this way, their model can learn the weight of each task.

Chapter 4

Head Pose Estimation: HHP-Net

This chapter introduces our algorithm for Head Pose Estimation and the LAEO algorithm, which uses the head pose to retrieve if people in the scene are looking at each other. The method called HHP-Net is particularly attractive due to its lightness and versatility with respect to different pose estimators.

4.1 Introduction

Social interaction analysis is becoming increasingly important in the computer vision community, thanks to the fact that some novel methods open the field of new studies in this context. The recent advances of Neural Networks and the accessibility of hardware resources drawn the interest towards the automation of some known techniques used to monitor human behaviours and interactions.

There is a lot of information that can be acquired via cameras to analyse people and interactions in couples or groups, in the social analysis domain. One of these cues is the head orientation which can give information about the location or direction of interest of a person in the scene. Furthermore, in Human-Machine interactions the amount of information acquired by the head direction -so indirectly to the focus of attention- can be fundamental: both for safety issues, a person often moves towards its focus of attention, and also for humanoid robotics, because humans use the head also to take turns in speaking, finding confirmation or pointing to the listener their focus of attention.

In this chapter, we will outline our novel approach (Cantarini et al., 2022; Figari Tomenotti et al., 2024) designed to address the task of Head Pose Estimation using single RGB images as input. The output of our methodology consists of three vectors describing the head pose orientation in the 3D World and an uncertainty value which states how much the network is confident of its output.

We will emphasise its efficiency in terms of computational time and resources and highlight its flexibility as a plug-in method for any human pose estimator. The chapter is organised in the following manner:

- Introduction to the Neural Network architecture
- Explanation and derivation of our proposed loss function

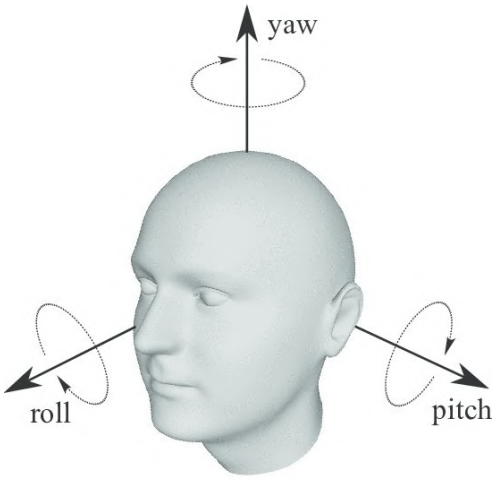


Fig. 4.1 Head Pose expressed as a triplet of angles: Yaw, Pitch and Roll [source (Fernández et al., 2016)].

4.2 Problem Formulation

The starting point of our approach is the output of a 2D pose estimator providing a set of key points describing the pose of a human body in an image (Z. Cao et al., 2019; K. Duan et al., 2019; Lugaresi et al., 2019). These detectors commonly provide also a confidence measure on the keypoint location estimate, which represents an additional source of knowledge that can be injected into our approach.

We model the estimation of the head orientation as a multi-task regression problem, where a Neural Network predicts the 3D vector of the head orientation with angles in Euler notation (*yaw, pitch* and *roll*) see Fig. 4.1. The input is formed by a set of n semantic keypoints located on the image plane: $\{(x_1^i, x_2^i, c^i)\}_{i=1}^n$, with x_1^i the horizontal and x_2^i vertical coordinates and c_i the confidence of the i-th keypoint. Coordinates are centred and normalized according to, respectively, their centroid and maximum value; c_i is provided in the range $[0, 1]$. The value of confidence is particularly important, as it encodes missing points ($c = 0$) and low confidence

points. These situations may frequently occur in real-world applications, in particular in human-human interaction, because of occlusions, self-occlusions or lateral poses.

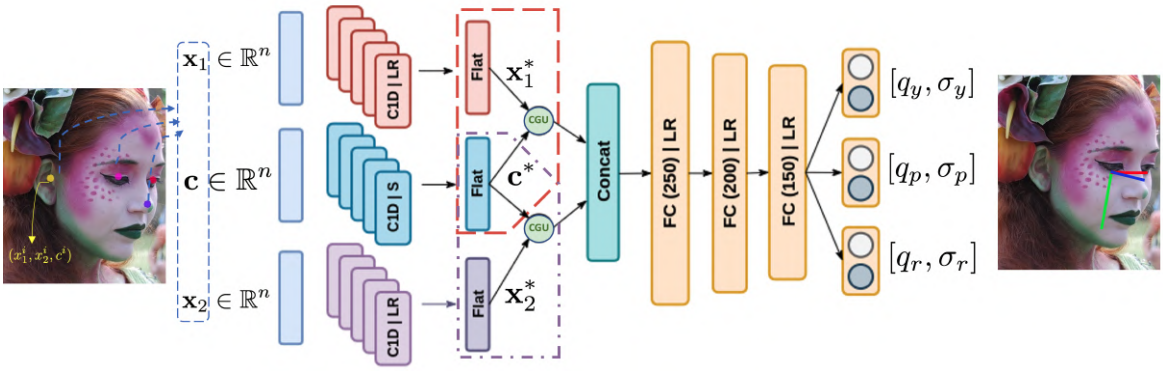


Fig. 4.2 A visual representation of our architecture. A set of keypoint locations with associated confidences $\{x_1^i, x_2^i, c^i\}_{i=1}^n$ is provided in input to the network, and processed with 1D convolutional layers. With a CGU we combine the intermediate outputs, that is then provided to the second part of the networks, composed of 3 FC layers to produce the final output, i.e. yaw, pitch and roll estimates with associated uncertainties [source (Figari Tomenotti et al., 2024)].

4.3 HHP-Net: The architecture

The Fig. 4.2 provides a sketch of our architecture. We formalize the input of the network as a triplet of vectors $\mathbf{x}_1 = [x_1^1, \ldots, x_1^n]$, $\mathbf{x}_2 = [x_2^1, \ldots, x_2^n]$ and $\mathbf{c} = [c^1, \ldots, c^n]$ incorporating positions and confidence of n key points describing a face. The input vectors are first processed in independent streams, with 5-channels 1D convolutions, followed by a Leaky ReLU for $\mathbf{x}_1$ and $\mathbf{x}_2$ – to avoid vanishing gradient issues – and sigmoid activation for the confidence vector $\mathbf{c}$ – to smoothly control the impact of different confidence values.

The outputs of the 1D convolutional layers are flattened to obtain $\mathbf{x}_1^*$, $\mathbf{x}_2^*$ and $\mathbf{c}^*$ from the independent streams. They are then combined, using an element-wise multiplication to obtain two vectors $\mathbf{v}_1 = \mathbf{x}_1^* \otimes \mathbf{c}^*$ and $\mathbf{v}_2 = \mathbf{x}_2^* \otimes \mathbf{c}^*$, following the logic of the Confidence Gated Unit (CGU) proposed in (Dias et al., 2020). The CGU is composed of ReLU+sigmoid activation functions. As visualised in Fig. 4.3, ReLU and sigmoid are applied, respectively, to coordinates (x_1^i or x_2^i) and confidence (c_i); their outputs are finally multiplied. The CGU emulates the behaviour of a gate, controlled by the confidence, as it returns values near 0 in the case of low confidence.

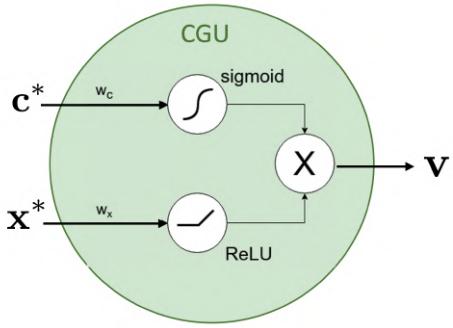


Fig. 4.3 Confidence Gated Unit (CGU) [source (Figari Tomenotti et al., 2024)].

The two gated outputs $\mathbf{v}_1$ and $\mathbf{v}_2$ are concatenated to obtain a single vector, which is provided to the intermediate part of the architecture, where a sequence of three fully connected layers consisting of 250, 200 and 150 neurons respectively is employed. Each layer includes a LeakyReLU, again to avoid vanishing gradients, as a non-linear activation function. Three output layers return the estimated angles, each of which is associated with its uncertainty value – details are reported in the next section.

4.4 The Loss Function

To train the network we design a multi-task loss function incorporating heteroscedastic aleatoric uncertainty. With respect to classical Neural Networks, a Heteroscedastic Neural Network provides an estimate of the uncertainty of each prediction. This is particularly useful to capture noise within input observations: noise in our case is related to inherent noise in key point localization which may be affected by difficult viewpoints or occlusions. Indeed, some poses are intrinsically noisier and more prone to self-occlusions (see for instance examples in Fig. 5.1). This type of uncertainty may be learned as a function of the data, thus the output will include not only the three angles (yaw, pitch, roll), stored in a vector $\mathbf{q} = [q_y, q_p, q_r]$, but also the uncertainty values associated with them $\sigma = [\sigma_y, \sigma_p, \sigma_r]$.

We now discuss how we derive the multi-task loss function starting from a simple heteroscedastic loss formulation

4.4.1 Heteroscedastic Single-Task Loss Function: general formulation

Without loss of generality, we reason on a simple regression problem where we want to estimate a function $f_\omega : \mathbb{R}^n \to \mathbb{R}$ so that

$$y = f_\omega(\mathbf{x}) + \varepsilon(\mathbf{x}). \tag{4.1}$$

The output is thus the sum between the function $f_\omega(\mathbf{x})$ – that depends on some parameters ω and the input $\mathbf{x}$ – and $\varepsilon(\mathbf{x})$, that is the noise only depending on the input $\mathbf{x}$ (Nix and Weigend, 1994).

To quantify the uncertainty, we train a model to learn from a training set $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ a function that estimates both the mean and the variance of a target distribution using a maximum-likelihood formulation of a neural network (MacKay, 1992). To this purpose, we need to assume that the errors are normally distributed $\varepsilon(\mathbf{x}_i) \sim \mathcal{N}\left(0, \sigma(\mathbf{x}_i)^2\right)$ hence the likelihood for each point $\mathbf{x}_i$ is:

$$p(y_i|\mathbf{x}_i;\omega) = \mathcal{N}(f_\omega(\mathbf{x}_i), \sigma(\mathbf{x}_i)^2) = \frac{1}{\sqrt{2\pi\sigma(\mathbf{x}_i)^2}} e^{\left(-\frac{(y-f_\omega(\mathbf{x}_i))^2}{2\sigma(\mathbf{x}_i)^2}\right)} \tag{4.2}$$

where y_i is the mean of this distribution and $\sigma(\mathbf{x}_i)^2$ is the variance. Hence, from a structural point of view, in addition to the estimation of the y_i, the heteroscedastic neural network architecture must be modified to also output a prediction of the variance: the latter quantifies the uncertainty associated with the prediction based on the noise in the training samples. Notice that the uncertainty is a function of the input e.g. if the noise is uniform over all the input values, the uncertainty should be constant.

Applying the logarithm to both sides of Eq. (4.2), we obtain a log-likelihood to maximize over the training set, i.e.

$$\max_{\omega} \quad \frac{1}{n}\sum_{i=1}^{\ell} -\frac{1}{2\hat{\sigma}\left(\mathbf{x}_i\right)^2}(y_i - \hat{f}_\omega\left(\mathbf{x}_i\right))^2 - \frac{1}{2}\log\hat{\sigma}\left(\mathbf{x}_i\right)^2 - \frac{1}{2}\log(2\pi)^{[1]} \tag{4.3}$$

where $\hat{f}_\omega$ and $\hat{\sigma}$ are, respectively, the prediction function and uncertainty estimated by the heteroscedastic neural network. Equivalently:

$$\min_{\omega} \quad \frac{1}{n}\sum_{i=1}^{\ell} \frac{1}{2\hat{\sigma}\left(\mathbf{x}_i\right)^2}(\mathbf{y}_i - \hat{f}_\omega\left(\mathbf{x}_i\right))^2 + \frac{1}{2}\log\hat{\sigma}\left(\mathbf{x}_i\right)^2 \tag{4.4}$$

An alternative formulation based on the change of variable $\hat{s}_i = \log\hat{\sigma}(\mathbf{x}_i)^2$ can be adopted to avoid exploding uncertainties during training (Kendall and Gal, 2017), leading to the final problem formulation:

$$\min_{\omega} \quad \frac{1}{n}\sum_{i=1}^{\ell} \frac{1}{2}e^{(-\hat{s}_i)}(y_i - \hat{f}_\omega\left(\mathbf{x}_i\right))^2 + \frac{1}{2}\hat{s}_i \tag{4.5}$$

[1] In the following the last term is ignored as it is a constant.

from which we derive the heteroscedastic loss function in Eq. (4.6) similarly to (Kendall and Gal, 2017)

$$\mathscr{L}_H(y, \hat{f}_\omega(\mathbf{x}), \hat{s}) = \frac{1}{2} e^{(-\hat{s})} (y - \hat{f}_\omega(\mathbf{x}))^2 + \frac{1}{2}\hat{s} \tag{4.6}$$

Notice finally that

$$\mathscr{L}_H(y, \hat{f}_\omega(\mathbf{x}), \hat{s}) = \frac{1}{2} e^{(-\hat{s})} \mathscr{L}_{MSE}(y, \hat{f}_{\omega(\mathbf{x})}) + \frac{1}{2}\hat{s} \tag{4.7}$$

where $\mathscr{L}_{MSE}$ is the classical square loss.

4.4.2 Heteroscedastic Multi-Task Loss Function:

We now specify to our problem the general formulation of the heteroscedastic loss function derived in the previous section.
We extend the model in Eq. (4.1) to represent a multi-task problem where the three components of the output are $\mathbf{q} = [q_y, q_p, q_r]$ and the associated uncertainties are $\sigma = [\sigma_y, \sigma_p, \sigma_r]$ (as usual we refer to the three angles yaw (y), pitch (p) and roll (r)). Hence, the single tasks within the multi-task formulation refer to the estimation of the three angles separately. In our solution, we estimate them by optimizing a unique function and exploiting their synergies. The input is composed of $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{c}$, that are respectively the coordinates of the key points detected on the face and the confidence in their detection.
We can now derive the multi-task heteroscedastic loss function we employ in our method:

$$\begin{aligned}\mathscr{L}_{HMT}(\mathbf{q}, \hat{\mathbf{q}}, \hat{\sigma}) &= \sum_{k \in \{y,p,r\}} \mathscr{L}_H(q_k, \hat{f}_k(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}), \hat{s}_k) \\ &= \sum_{k \in \{y,p,r\}} \left(\frac{1}{2} e^{(-\hat{s}_k)} (q_k - \hat{f}_k(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}))^2 + \frac{1}{2}\hat{s}_k \right).\end{aligned} \tag{4.8}$$

where

$$\hat{\mathbf{q}} = \hat{\mathbf{f}}_\omega(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}) = [\hat{f}_y(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}), \hat{f}_p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}), \hat{f}_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c})] \tag{4.9}$$

and

$$\hat{\sigma}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}) = [\hat{\sigma}_y(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}), \hat{\sigma}_p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c}), \hat{\sigma}_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c})] \tag{4.10}$$

are, respectively, function and uncertainty estimated by the heteroscedastic neural network, and for $k \in \{y, p, r\}$

$$\hat{s}_k = \log \hat{\sigma}_k(\mathbf{x}_1, \mathbf{x}_2, \mathbf{c})^2. \tag{4.11}$$

With this formulation, we obtain a data-driven uncertainty estimation for each angle, used as a weight of each sub-loss. The uncertainty can increase the robustness of the network when

dealing with noisy input data, we will empirically show a correlation between uncertainty and estimation error

4.5 Human Interactions: LAEO detection

We finally discuss a task where our method finds a natural application, i.e. the analysis of social interactions, for which the head directions represent a strong visual cue of non-verbal human-human communication (Abele, 1986). We consider scenarios where a small group of people is involved in a social experience, and we pay particular attention to people *looking at each other* (LAEO). Particular requirements for developing this task were devoted in finding a technical solution with very few computational requirements, to be added to our Head Pose Estimation pipeline without compromising its lightness and computational efficiency. So the goal was not to have a state-of-the-art solution such as (M. Marín-Jiménez et al., 2020) but to use it as a benchmark for our performances.

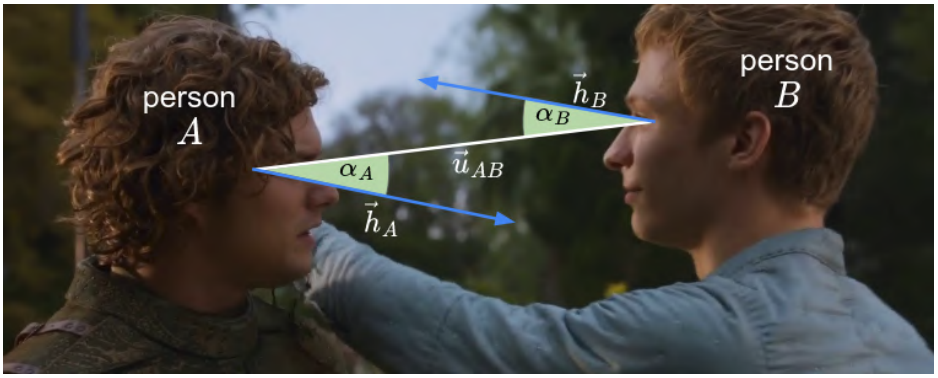


Fig. 4.4 A visual sketch with our formulation of the LAEO detection task (for readability the vectors are denoted with arrows) [source (Figari Tomenotti et al., 2024)].

LAEO algorithm. Fig. 4.4 provides a visual sketch of our formulation of the task. Let us consider the two people present in the scene, A and B in our example, whose positions can be compactly described with the head centroids (x_A, y_A) and (x_B, y_B). We start from the head pose estimated for each of them, $\mathbf{q}^A$ and $\mathbf{q}^B$ respectively, and obtain a projection of the corresponding direction on the image plane: for the subject A, given the triplet of angles (q_y^A, q_p^A, q_r^A), we derive the end-point of the head direction on the image plane (x'_A, y'_A) as $x'_A = \sin(q_y^A)$ and $y'_A = -\cos(q_y^A)\sin(q_p^A)$. The projection is computed as $\mathbf{H}_A = (x'_A - x_A, y'_A - y_A)$. Similarly we obtain $\mathbf{H}_B$ for the other subject.
Then, we estimate a measure of interaction between each pair of people considering the

vector $\mathbf{u}_{AB}$ connecting the two head centroids, the vector $\mathbf{H}_A$ and the angle α_A between the two: the measure of the interaction is given by the cosine of the angle α_A. The same applies to person B with $\mathbf{u}_{BA} = -\mathbf{u}_{AB}$ and α_B. The average between the two measures gives the LAEO value and thresholding on such measure allows us to detect LAEO pairs.
We build our approach on this baseline method, incorporating knowledge from the uncertainty associated with the 3D angles (the method is sketched in Algorithm 1). Given the triplets of uncertainties associated with the two heads poses, (s_A^y, s_A^p, s_A^r) and (s_B^y, s_B^p, s_B^r), we compute the averages, $\hat{s}_A = \frac{1}{2}(s_A^y + s_A^p)$ and $\hat{s}_B = \frac{1}{2}(s_B^y + s_B^p)$; the roll component is discarded because it does not affect the gaze vector projection on the image plane. Following the intuition that estimates with high uncertainty should be less reliable, we compute a weight to adjust the contribution of each subject to the interaction measure depending on the confidence we have in it, essentially deciding a threshold above which the estimate is considered unreliable. For the subject A this can be formulated as $w_A = \mathbb{1}_X(\hat{s}_A)$ where $X = [0, \delta]$ with δ an appropriate threshold on the uncertainty, and $\mathbb{1}_X : \mathbb{R} \to \{0, 1\}$ the indicator function on the interval X. δ is computed as the average uncertainty plus the standard deviation, both of them computed on the entire training set (in the experiments $\delta = 7$).

Algorithm 1 Fast LAEO Detection

1: **Input**: Head centroids (x_A, y_A) and (x_B, y_B); projections of head directions (x'_A, y'_A) and (x'_B, y'_B); uncertainty weights w_A and w_B
2: $\mathbf{u}_{AB} \leftarrow (x_B - x_A, y_B - y_A)$
3: $\mathbf{H}_A \leftarrow (x'_A - x_A, y'_A - y_A)$
4: $\mathbf{H}_B \leftarrow (x'_B - x_B, y'_B - y_B)$
5: $cos(\alpha_A) \leftarrow \frac{\mathbf{u}_{AB} \cdot \mathbf{H}_A}{|\mathbf{u}_{AB}| \cdot |\mathbf{H}_A|}$
6: $cos(\alpha_B) \leftarrow \frac{-\mathbf{u}_{AB} \cdot \mathbf{H}_B}{|\mathbf{u}_{AB}| \cdot |\mathbf{H}_B|}$
7: Compute the level of mutual interaction $LAEO_{value} = w_A cos(\alpha_A) + w_B cos(\alpha_B)$
8: Return $LAEO_{value}$

4.5.1 Extension to 3D

This small section presents two ideas to extend in 3D what has been done in 2D until now.

The former 2.5D extension can be done for the LAEO algorithm by only extracting some contextual information and trying to understand if the two people we want to calculate the LAEO for, are on the same plane also in the 3D World. Because, one of the main weaknesses of our method is that two parallel planes, coplanar with the camera are indistinguishable and so people not in a real LAEO interaction can be counted generating a false detection. To this

respect, a simple check on the skeletal dimensionality may help to have at least a couple of different planes (foreground and background) to do the computations.

The full 3D extension of the LAEO algorithm is straightforward to theorise because instead of evaluating one angle and computing its cosine, we need to use two angles and a 3D reference system. It is less straightforward to implement, especially with computational constraints, indeed we leave it for future work. The idea should be to have a full reconstruction of the z-coordinate, so having a depth map for the entire image and using our computed 3D head pose in a real 3D World. So having one LAEO algorithm working on the xy plane (camera plane) and one working on the yz plane (a bird-view plane), and then detecting a LAEO interaction if and only if on both planes the detection is positive and above threshold.

Chapter 5

Experiments on Head Pose estimation and LAEO Detection

In this chapter, we report the experimental analysis we performed to assess our approach. We first discuss in detail the implementation, the datasets and the experimental protocols we adopted, and then provide qualitative and quantitative results. In particular, we perform ablation studies to show the benefit of each element in the method, discuss the role of the uncertainty and the relation with the estimated error, and evaluate the transfer capability of the model across datasets.
It is worth observing that there are no free parameters to be tuned in our method.

5.1 Implementation Details

Unless otherwise stated, we adopt OpenPose (Z. Cao et al., 2019) as a key points extractor, as it provides a good balance between efficiency and accuracy. Among the 25 body key points it provides, in this work we focus on the five located on the face – left and right eye, left and right ear, nose – thus obtaining a triplet of input vectors $\mathbf{x}_1 = [x_1^1, \ldots, x_1^5]$, $\mathbf{x}_2 = [x_2^1, \ldots, x_2^5]$ and $\mathbf{c} = [c^1, \ldots, c^5]$.
For the initialization, the weights of each layer are randomly sampled from a normal distribution with $\mu = 0$ and $\sigma^2 = 0.05$. The network has been trained for a number of epochs that ranges from 100 to 1000 depending on the dataset. We used Adam as an optimizer, with a learning rate 0.001, and a batch size of 64. The weights associated with the best validation loss have been selected as the final model[1].

[1] Code and pre-trained weights are available at https://github.com/Malga-Vision/HHP-Net

5.2 Datasets and Protocols

Fig. 5.1 Sample frames from the public datasets we adopted in our experimental analysis: BIWI (top row), AFLW-2000 (middle row), and 300W-LP (bottom). For readability of the figures, we report their greyscale version with an arrow in red which is the 2D projection of the head direction. Being the projection of a 3D vector, it can also be a point, e.g. like in the top-left image where the direction of view is 'outside' the page [source (Figari Tomenotti et al., 2024)].

We evaluate the effectiveness of our approach on three different datasets (see sample frames in Fig. 5.1):

- BIWI (Gabriele Fanelli et al., 2011) includes $\sim 15K$ images of 24 people acquired in a controlled scenario. The head pose orientation covers the range $\pm 75^o$ for the yaw angle and $\pm 60^o$ for the pitch. The ground truth has been obtained by fitting a 3D face model.

- AFLW-2000 (X. Yin et al., 2017) contains the first 2000 images of the in-the-wild AFLW dataset (Martin Koestinger and Bischof, 2011), a large-scale collection of face images with a large variety in appearance and environmental conditions. The annotation has been obtained by fitting a 3D face model.

- 300W-LP (Sagonas et al., 2013) is a collection of different in-the-wild datasets, grouped and re-annotated to account for different types of variability, such as pose, expression, illumination, background, occlusion, and image quality. A face model is fit on each image, distorted to vary the yaw of the face.

For all the datasets, the ground truth takes the form of a triplet of angles in Euler notation expressed with respect to a reference frontal pose
According to previous works (e.g.(T.-Y. Yang et al., 2019)), in the comparative analysis we adopt two main protocols:

P1 Training is performed on a single dataset (300W-LP), while BIWI and AFLW-2000 are used as test.

P2 Training and test set are derived from the BIWI dataset using the split 16-9 sequences, for training and test respectively, following the procedure proposed in (G. Fanelli et al., 2013).

5.3 Method assessment

In this section, we present an experimental assessment to discuss the core properties of our approach.
The output is visualized by projecting the angles on the image plane according to the Tait-Bryan angles. The projections are computed as

$$\begin{aligned} x_r &= \cos q_y \cdot \cos q_r + \Delta_x \\ y_r &= \cos q_p \cdot \sin q_r + \cos q_r \cdot \sin q_y \cdot \sin q_p + \Delta_y \\ x_g &= -\cos q_y \cdot \sin q_r + \Delta_x \\ y_g &= \cos q_p \cdot \cos q_r - \sin q_y \cdot \sin q_p \cdot \sin q_r + \Delta_y \\ x_b &= \sin q_y + \Delta_x \\ y_b &= -\cos q_y \cdot \sin q_p + \Delta_y \end{aligned} \tag{5.1}$$

where (x_r, y_r), (x_g, y_g) and (x_b, y_b) are the image coordinates of the endpoints of red, green and blue vectors, while (Δ_x, Δ_y) is the application point they have in common.

In the following, we provide an assessment of the properties and meaningfulness of the uncertainty measures.

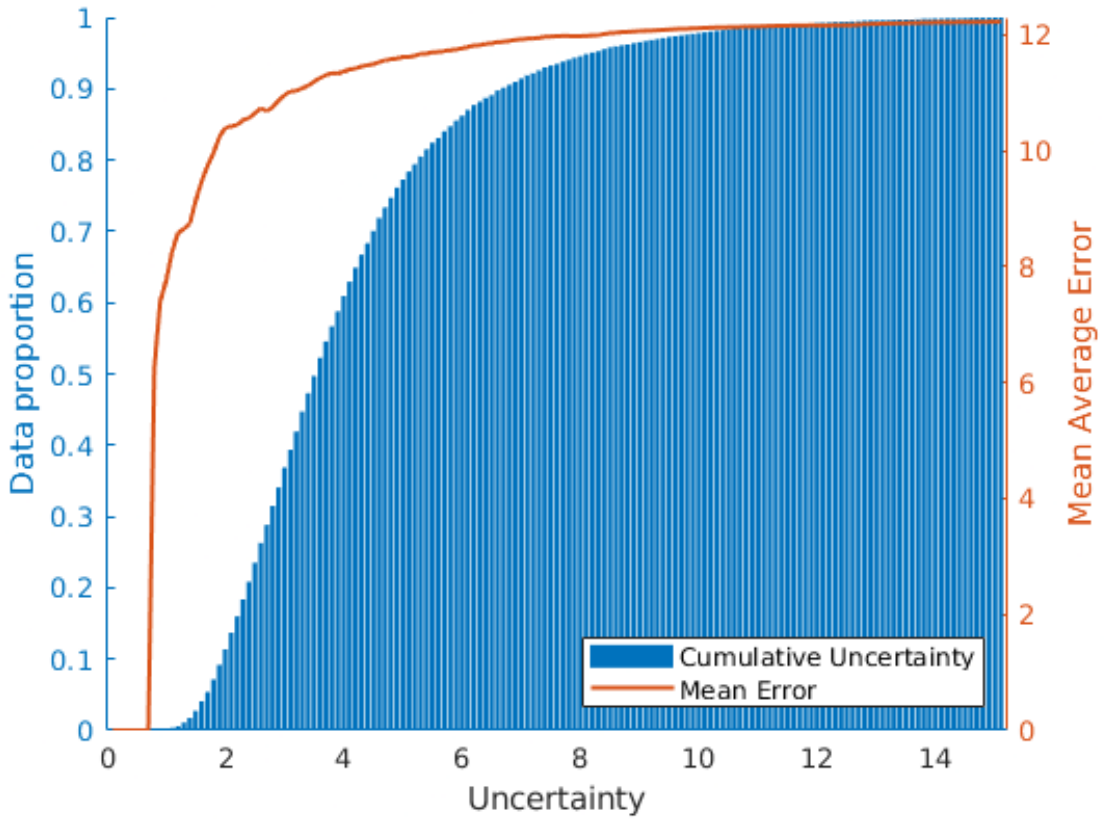


Fig. 5.2 Cumulative angular error as a function of the average uncertainty (red). And data proportion with at least the uncertainty reported on the x-axis (blue) [source (Figari Tomenotti et al., 2024)].

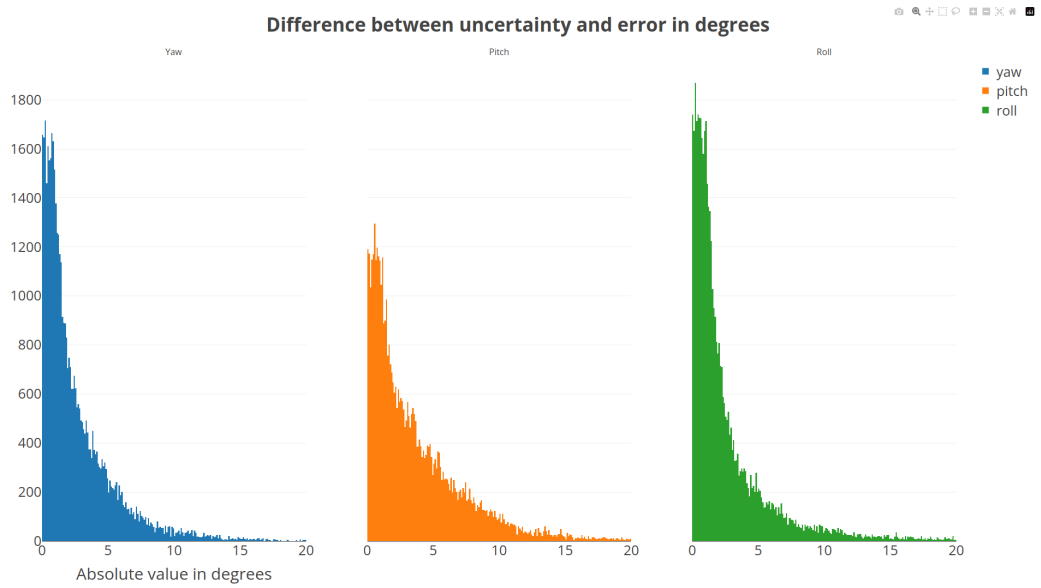


Fig. 5.3 Shows the occurrences of test data divided in bins of absolute difference between Uncertainty (in degrees) and the Error (in degree); the zeroth bin on the x-axis is when error and uncertainties coincide. Left: Yaw angle. Centre: Pitch angle. Right: Roll angle. [source (Figari Tomenotti et al., 2024)].

5.3.1 Uncertainty Estimations Quality

Fig. 5.2 reports a cumulative analysis of the amount of data associated with a given uncertainty, highlighting how the average error grows with the uncertainty – in agreement with what has been reported in (Dias et al., 2020).
Given the assumptions in Section 4.4.1 of a normal distribution for the errors, the $\sigma(\mathbf{x_i})^2$ is the variance of this distribution and the parameter we are going to estimate for each angle in the regression process. So, fixed one angle (e.g. yaw) it can be seen as the variance of the retrieved angle (yaw). Under this perspective, it is straightforward to read it in degrees. However, having implemented the algorithm and defined the uncertainty as $log(\hat{\sigma}(\mathbf{x_i}))^2$ or better $log(\hat{\sigma}(\mathbf{x_1}, \mathbf{x_2}, \mathbf{c}))^2$, we retrieved the degree information as

$$log(\hat{\sigma}(\mathbf{x_1}, \mathbf{x_2}, \mathbf{c}))^2 = s_i \Leftrightarrow \sigma = \sqrt{e^{s_i}} \tag{5.2}$$

Hence, the interpretability of our uncertainty measure is strengthened by the fact it can be expressed in degrees, as the estimated angles. In this way, the two outcomes of our model can be directly compared. In Fig. 5.2 (bottom) we report the histogram of the absolute value of the difference between the angle and corresponding uncertainty. It can be observed that it is predominantly very low, in 71% of the cases below 3 degrees, 88% below 5 degrees and 98%

below 10 degrees. This shows that the uncertainty measures can be adopted as an indicator of the reliability of our estimated angles. To have an idea about how much the uncertainty and the error coincide we plot the difference between one and the other in Fig. 5.3. The more data fall on the small x-axis values, more uncertainty and error coincides.
Similarly to what was observed in (D. Feng et al., 2019), we also notice a strong correlation between the uncertainty values associated with the three predicted angles. To quantify the correlation we computed the Pearson correlation between the uncertainties of all pairs of angles, obtaining 0.72 for (*yaw*, *pitch*), 0.78 for (*yaw*, *roll*), and 0.92 for (*pitch*, *roll*).

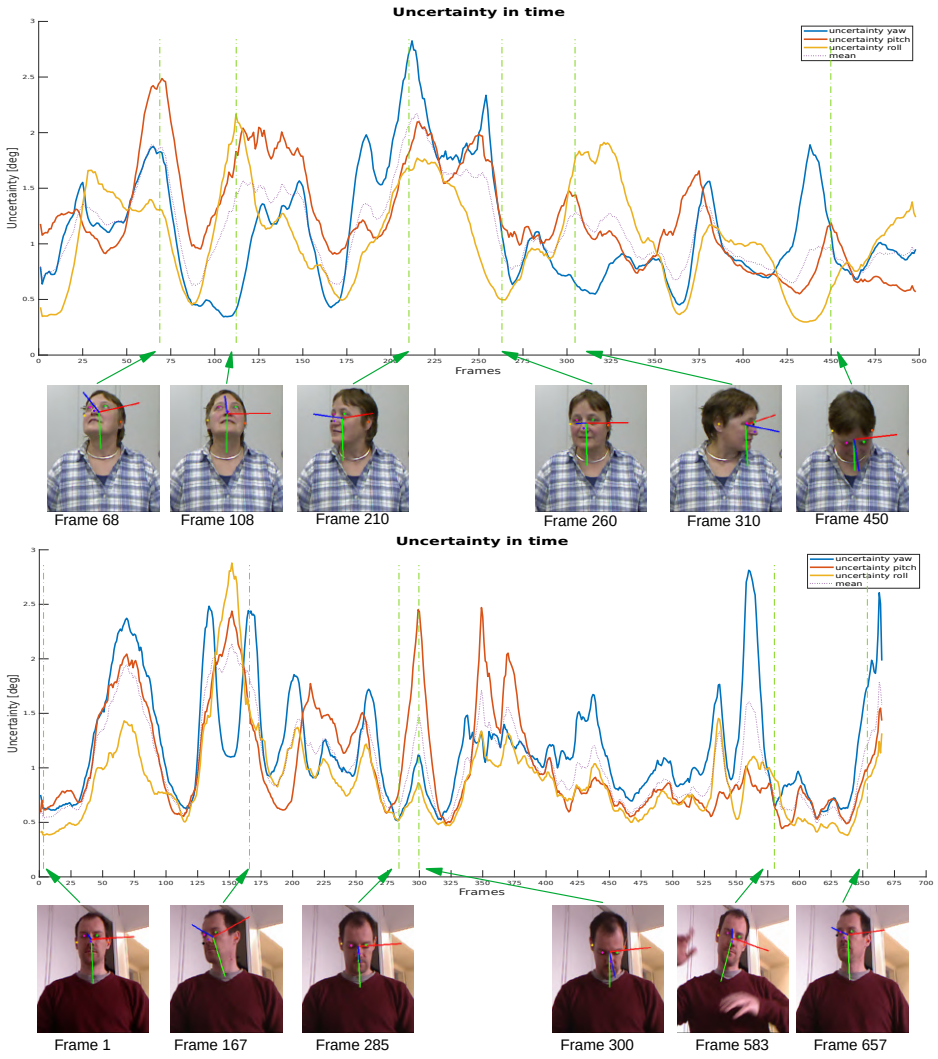


Fig. 5.4 Examples of how the uncertainties (in degrees) are influenced by the instantaneous head pose of a subject moving in front of a camera over time. We report in blue the yaw uncertainty, in orange the pitch uncertainty and in yellow the roll uncertainty. In dotted-purple we mark the mean uncertainty. The scale is in degrees of uncertainty. It can be observed that the uncertainties are very close to zero for the neutral head pose (frame 1 of the first sequence) and start to increase when the head rotates [source (Figari Tomenotti et al., 2024)].

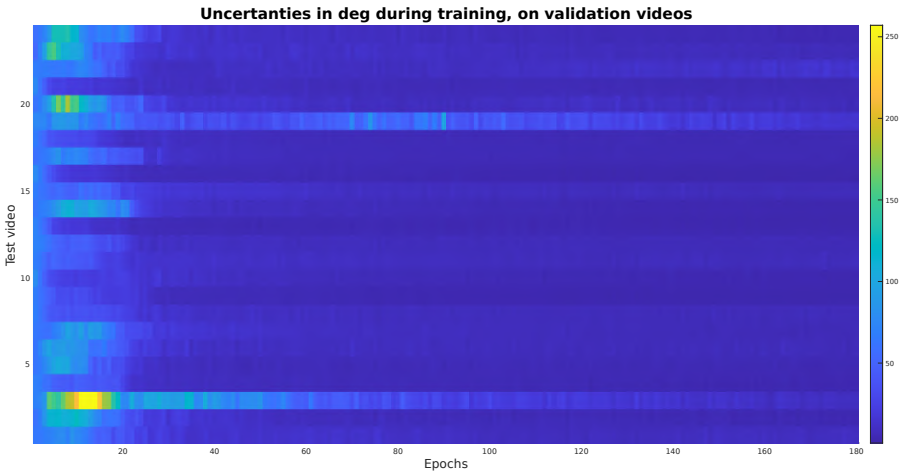


Fig. 5.5 This plot shows the uncertainties (sum on the three angles) during the training. On the x-axis, there are the epochs. On the y-axis, there are the videos (from which the validation data is composed). The colour code shows in blue a small uncertainty and the more yellow the higher the errors.

In the Fig. 5.5 we can appreciate how the network learns the uncertainty estimation. We know that in each epoch, each validation video (validation data) is still the same with the same amount of complexities and the same keypoints, but the network learns how to use the keypoints to extract the correct angles and get more sure about its predictions. So we can state that the rows exhibiting the more yellow components are the ones with more difficult subjects to be estimated. The few which remain a bit more 'yellowish' also at the end have the subject with some ambiguous or difficult pose to be estimated.

5.3.2 Uncertainty Estimation and Model Interpretation.

We now analyse the factors that may influence the uncertainty estimation, with a focus on the characteristics of the head pose to be predicted. In Fig. 5.4 we report the trend of the uncertainty associated with the prediction obtained from video sequences where a subject rotates the head offering different test poses to the method. Representative frames, providing an intuition about the transitions between poses in the sequence, are reported below the plot. It is easy to observe that for some poses (the ones associated with ambiguous views or partial occlusions that hide some key points on the face) the uncertainty is higher. The lowest uncertainty values are associated with frontal views, the ones providing the most visible and non-ambiguous key points.
Inspired by these observations, we now evaluate the dependence of the uncertainty and the

error on the quality and quantity of the input key points.

5.3.3 On the Number of Keypoints.

We observe the influence of the quality and quantity of input semantic features on the final head pose estimate. In Fig. 5.6, we analyse the performance of our method in terms of uncertainty values (bottom) and absolute angular error (top) as the number of available key points changes. On the left, we cluster faces according to the number of key points detected by OpenPose. When only 3 key points are available the uncertainty is rather high on average. Increasing the number of points uncertainty is progressively reduced, with a similar trend shown by the error. This confirms the intuition that the more input points the method has, the higher its confidence in the prediction, which is more reliable and accurate.

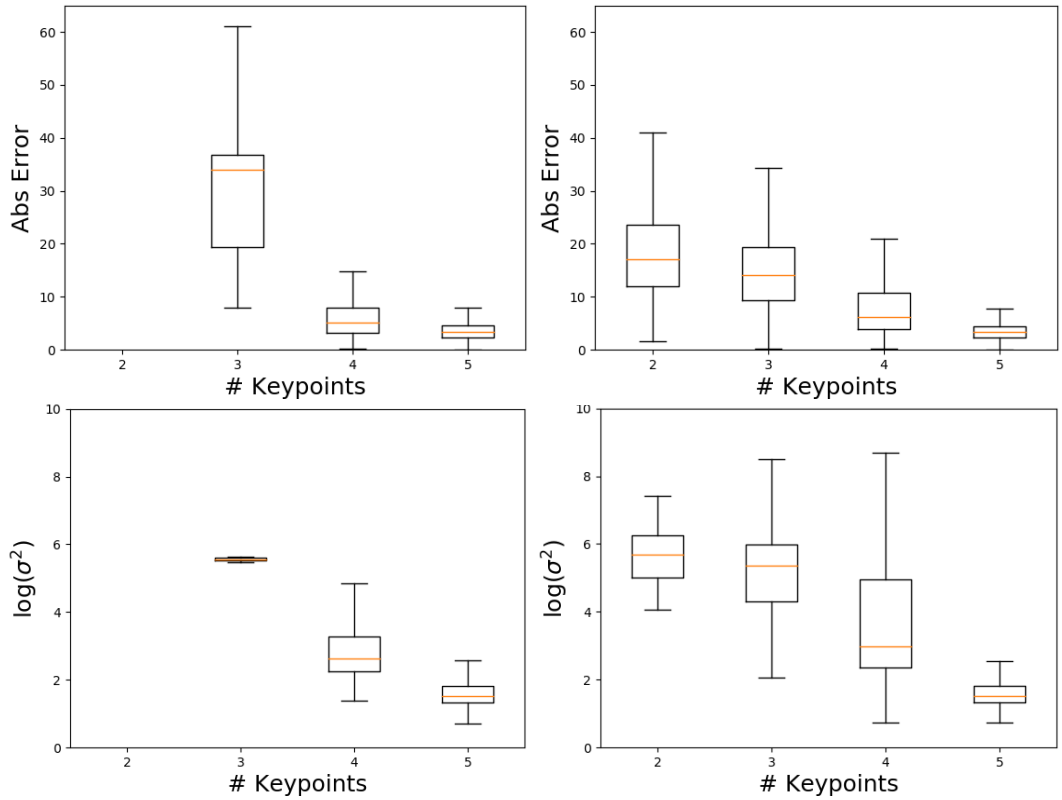


Fig. 5.6 Performance of our method (top row: mean angular error in angles, bottom row: uncertainty) with respect to the number of input points, considering the outputs of OpenPose (left) and randomly dropping points (right). Training: 300W-LP Test: BIWI. Uncertainty is presented in a log scale visualization for a clearer view [source (Figari Tomenotti et al., 2024)].

On the right, we randomly drop points from the available input to simulate an even more challenging scenario for our method. When points are randomly dropped, we only consider samples with more than two points. When all 5 key points are available, the uncertainty is compactly lower (confirming what was already observed in the previous experiment) as the method can rely on a more comprehensive representation of the input. In the intermediate cases – where we may have 2, 3, or 4 key points available in input – the uncertainty progressively decreases, but we also have a higher degree of variability, as some key-point configurations are more significant than others and thus the amount of information they provide to the model may be unevenly reflecting the concept that the noise could be different for each input sample. With respect to the plots in the left column of Fig. 5.6, the box plots at right show a higher standard deviation since randomly dropping points from the input we simulate a higher variability in the input configurations with respect to the ones usually provided by OpenPose and from the datasets we used.

5.3.4 Plugging in Different Pose Detectors.

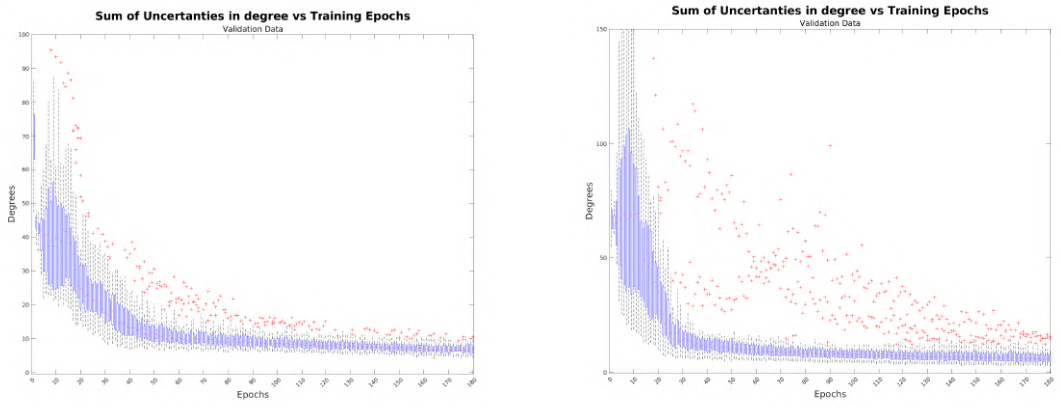


Fig. 5.7 Both images represent the degrees of uncertainties in the output of the network during the training. On the x-axis, there are the epochs; on the y-axis, there are the sum of the uncertainties on the three outputs. The blue bars are the mean with their standard deviation represented as dashed lines and the outliers are in red. **Top** Openpose, **Bottom** Centernet.

Here we assess the robustness of our approach to different choices of 2D pose estimators. More specifically, we employ OpenPose (Z. Cao et al., 2019), CenterNet (K. Duan et al., 2019), and MediaPipe (Lugaresi et al., 2019) and consider all the pairs for train-test. The results are reported in Table 5.1. If we read the table row-wise we may analyze the behaviour of models obtained from the different pose detectors on the output of different nature. It

shows that CenterNet and Openpose are rather interchangeable (OpenPose in particular provides very similar results when tested on itself or Centernet), while Mediapipe is not. The reason is that its output is rather different in terms of localization of the key points and behaviour in the presence of occlusions (MediaPipe never provides zero confidence for occluded key points), reducing the benefit of the Confidence Gated Unit.

Table 5.1 Training and testing HHP-Net with inputs from different 2D pose estimators on the BIWI dataset. In the table, we report the MAE (Mean Absolute Error in degree) averaged over the three angles and the standard deviation.

Train \ Test	Centernet	Mediapipe	Openpose
Centernet	3.33 ±0.91	7.57 ±2.48	6.73 ±5.88
Mediapipe	13.31 ±7.96	5.99 ±1.56	14.55±7.59
Openpose	4.64 ±1.99	7.08 ±2.37	4.51±1.27

In Fig. 5.7 we show how the network learns the correct data distribution, also learning to be more precise in its predictions with the epochs. More in-depth we can see in the top image how Openpose converges more easily especially having less uncertainty on its predictions. In contrast, the keypoints extracted with Centernet retain a certain value of uncertainty also after some epochs. We can also see how the uncertainty does not decrease to zero, but due to the data distribution, remains higher than this, indicating that the network cannot have a zero error on all the samples.

5.4 Removing the Uncertainty: an ablation study

We perform an ablation study by removing the uncertainty from our model. To this purpose, we consider two variations of the method (with *y*=yaw, *p*=pitch and *r*=roll):

MSE: we directly regress the three angles adopting a loss computed as the sum of the Mean Squared Error (MSE) on each angle:

$$\mathscr{L}_{MSE-MT}(\mathbf{q},\hat{\mathbf{q}}) = \sum_{k\in\{y,p,r\}} (q_k - \hat{f}_k(\mathbf{x}_1,\mathbf{x}_2,\mathbf{c}))^2. \tag{5.3}$$

where $\mathbf{q} = [q_y, q_p, q_r]$, and
$\hat{\mathbf{q}} = [\hat{f}_y(\mathbf{x}_1,\mathbf{x}_2,\mathbf{c}), \hat{f}_p(\mathbf{x}_1,\mathbf{x}_2,\mathbf{c}), \hat{f}_r(\mathbf{x}_1,\mathbf{x}_2,\mathbf{c})]$.

COMB: we employ an alternative loss function $\mathscr{L}_{COMB}$ proposed in (Ruiz et al., 2018) which has been proved to be very successful on the same estimation task. The loss

allows for jointly solving a multi-class classification (with N classes corresponding to binned angles) and a regression task, and it can be formalized as follows :

$$\mathscr{L}_{COMB}(\mathbf{q},\hat{\mathbf{q}}) = \mathscr{L}_{CE-MT}(\mathbf{q},\hat{\mathbf{q}}) + \alpha * \mathscr{L}_{MSE-MT}(\mathbf{q},\hat{\mathbf{q}}) \tag{5.4}$$

where

$$\mathscr{L}_{CE-MT}(\mathbf{q},\hat{\mathbf{q}}) = \sum_{k\in\{y,p,r\}} \left[-\sum_{j=1}^{N} q_k^j \log\left(\hat{f}_k^j(\mathbf{x}_1,\mathbf{x}_2,\mathbf{c})\right)\right] \tag{5.5}$$

is the cross-entropy loss adapted to our multi-task problem, while $\mathscr{L}_{MSE-MT}$ is the multi-task square loss of Eq. (5.3). Hence, the loss combines the cross entropy, computed between the binned angles, and the MSE loss, computed between the scalar angles; α is a hyperparameter that controls the weight of the regression loss. According to the original work, in the experiment, we set $\alpha = 1$.

In Table 5.2 we report the angular errors we obtain with the three different losses. As can be observed, learning the angles associated with the uncertainty provides the best average performance, showing the benefit of the uncertainty not only in terms of the interpretability of the model but also as a way to improve its effectiveness.

Table 5.2 Comparison among different loss functions (see text). **All errors are expressed in degrees** (°): err_y= yaw error, err_p=pitch error, err_r= roll error, MAE = Mean Absolute Error.

Train	Val	Loss	err_y	err_p	err_r	MAE
BIWI	BIWI	$\mathscr{L}_{MSE}$	2.90	4.80	3.34	3.70
BIWI	BIWI	$\mathscr{L}_{COMB}$	3.15	4.85	3.40	3.80
BIWI	BIWI	$\mathscr{L}_{HMT}$	3.04	4.79	3.21	**3.68**
300WLP	BIWI	$\mathscr{L}_{MSE}$	4.75	6.65	4.45	5.28
300WLP	BIWI	$\mathscr{L}_{COMB}$	4.67	8.08	4.87	5.88
300WLP	BIWI	$\mathscr{L}_{HMT}$	4.14	7.00	4.40	**5.18**
300WLP	AFLW2000	$\mathscr{L}_{MSE}$	5.72	10.41	8.08	8.07
300WLP	AFLW2000	$\mathscr{L}_{COMB}$	5.55	10.39	8.18	8.04
300WLP	AFLW2000	$\mathscr{L}_{HMT}$	5.26	10.12	7.73	**7.70**
AFLW	AFLW2000	$\mathscr{L}_{MSE}$	7.60	6.43	4.76	6.26
AFLW	AFLW2000	$\mathscr{L}_{COMB}$	7.31	6.55	4.68	6.18
AFLW	AFLW2000	$\mathscr{L}_{HMT}$	7.40	6.63	4.47	**6.16**

5.5 Comparisons With Other Approaches

We now perform a comparative analysis with state-of-the-art head pose estimators. For a fair comparison, we consider methods that use RGB images as inputs or features extracted from

them.

Table 5.3 Comparison following Protocol P2: BIWI is both training and test. Our model is the smallest (∼0.4MB) while providing only a small degradation with respect to the best result (∼ 0.4°). †,‡ *data respectively from* (T.-Y. Yang et al., 2019),(Dhingra, 2022)

Method	MB	Par. $\times 10^6$	err_y	err_p	err_r	MAE
D-HeadPose (Mukherjee and Robertson, 2015)	-	-	-	5.67	5.18	-
Drounard et al (Drouard et al., 2015)	-	-	4.9	5.9	4.7	5.16
DFA(J. Gu et al., 2017)	500†	138‡	3.91	4.03	3.03	3.66
DMLIR (Lathuiliere et al., 2017)	500	-	3.12	4.68	3.07	3.62
FSA-Caps-Fusion (T.-Y. Yang et al., 2019)	5.1	1.2	2.89	4.29	3.60	3.60
FND (H. Zhang et al., 2020)	5.8	-	3.0	3.98	2.88	3.29
img2pose (Albiero et al., 2021)	-	-	4.57	3.55	3.24	3.79
LwPosr (Dhingra, 2022)	-	0.15	3.62	4.65	3.78	4.01
QTNet (Hsu et al., 2018)	-	-	4.01	5.49	2.94	4.15
Ruiz (Ruiz et al., 2018) (α=2)	-	-	3.29	3.39	3.00	3.23
TriNet (Zhiwen Cao et al., 2021b)	-	26‡	2.44	3.04	2.93	2.80
Our approach	**∼0.4**	**∼0.09**	3.04	4.79	3.21	3.68

Table 5.4 Comparison following Protocol P1, where 300W-LP is the training, while BIWI is the test. Our method is still the smallest and performs better than all other approaches but (T.-Y. Yang et al., 2019). The latter is however associated with a model significantly larger than ours.
†,‡,††,∗ *data respectively from* (T.-Y. Yang et al., 2019), (Dhingra, 2022), (Y. Zhou and Gregson, 2020), (Ruiz et al., 2018))

Method	MB	Par. $\times 10^6$	err_y	err_p	err_r	MAE
Shao(K=0.5)(Shao et al., 2019)	93	24.6††	4.59	7.25	6.15	6.00
Ruiz (Ruiz et al., 2018))(α=2)	95.9†	23.9	5.17	6.98	3.39	5.18
Ruiz (Ruiz et al., 2018))(α=1)	95.9†	23.9	4.81	6.61	3.27	4.90
LwPosr α (Dhingra, 2022)	-	0.15	4.41	5.11	3.24	4.25
LwPosr (Dhingra, 2022)	-	0.15	4.11	4.87	3.19	4.05
FSA-Caps-Fusion (T.-Y. Yang et al., 2019)	5.1	1.2	4.27	4.96	2.76	4.00
TriNet (Zhiwen Cao et al., 2021b)	-	26‡	3.05	4.76	4.11	3.97
FND (H. Zhang et al., 2020)	5.8	-	4.52	4.70	2.56	3.93
WHENet-V (Y. Zhou and Gregson, 2020)	-	4.4	-	-	-	3.48
Our approach	**∼0.4**	**∼0.09**	4.14	7.00	4.40	5.18

The analysis is reported in Table 5.3, Table 5.4, and Table 5.5, where all errors are expressed in degrees (err_y= yaw error, err_p=pitch error, err_r= roll error), the model size is reported in MegaBytes (MB), and the MAE is the Mean Absolute Error.
As a first important observation, notice that our approach produces a significantly smaller model (0.4 MB). This was the main purpose of our work and it has been clearly achieved, as our method is about ∼12 times smaller than the closest model in the literature. According to the protocol followed by other works – all requiring a face detector but not including its size in their analysis – the size of our model does not include the pose estimator.
In terms of performances, Table 5.3 reports a comparison with respect to Protocol P2 (BIWI

Table 5.5 Comparison following Protocol P1, where 300W-LP is the training and AFLW 2000 is the test (note: ✠ = Trained on AFLW - AFLW2000). The performances show a slightly higher worsening with respect to alternative approaches, but the difference is still very limited (always less than 3°). †, ‡, †† *data respectively from* (T.-Y. Yang et al., 2019), (Dhingra, 2022), (Y. Zhou and Gregson, 2020)

Method	MB	Par. $\times 10^6$	$\mathbf{err}_y$	$\mathbf{err}_p$	$\mathbf{err}_r$	MAE
3DDFA (X. Zhu et al., 2019)	-	-	5.40	8.53	8.25	7.39
Ruiz(Ruiz et al., 2018)(α=1)	95.9†	23.9	6.92	6.64	5.67	6.41
Ruiz (Ruiz et al., 2018)(α=2)	95.9†	23.9	6.47	6.56	5.44	6.16
Shao(K=0.5)(Shao et al., 2019)	93	24.6††	4.59	7.25	6.15	6.00
FSA-Caps-Fusion (T.-Y. Yang et al., 2019)	5.1	1.2	4.50	6.08	4.64	5.07
Shao(K=0.5)(Shao et al., 2019)	93	24.6	5.07	6.37	4.99	5.48
WHENet-V (Y. Zhou and Gregson, 2020)	-	4.4	-	-	-	4.83
LwPosr (Dhingra, 2022)	-	0.15	4.80	6.38	4.88	5.35
LwPosr α (Dhingra, 2022)	-	0.15	4.44	6.06	4.35	4.95
TriNet (Zhiwen Cao et al., 2021b)	-	26‡	4.20	5.77	4.04	4.67
FND (H. Zhang et al., 2020)	5.8	-	3.78	5.61	3.88	4.42
Our approach	**∼0.4**	**∼0.09**	5.26	10.12	7.73	7.70
Our approach✠	**∼0.4**	**∼0.09**	7.40	6.63	4.47	6.16

dataset for training and test): the results we obtain are superior to (Mukherjee and Robertson, 2015; Drouard et al., 2015; G. Fanelli et al., 2013) and slightly below (J. Gu et al., 2017; Lathuiliere et al., 2017; T.-Y. Yang et al., 2019; H. Zhang et al., 2020) (less than 0.1 degrees of difference for the first three, less than 0.4 for the latter).

Table 5.4 refers to Protocol P1 (training carried out on 300W-LP, BIWI for the test): the experiment mainly evaluates the transfer potential to a different dataset with different properties. The table reports results obtained with methods relying on the estimation of 3D face models (X. Zhu et al., 2019; Kumar et al., 2017; Kazemi and Sullivan, 2014; Bulat and Tzimiropoulos, 2017) and methods based on analysing RGB image portions obtained by face detectors, such as (Shao et al., 2019; Ruiz et al., 2018; T.-Y. Yang et al., 2019).

We share with the latter group the main motivation for designing simple and more efficient procedures while keeping competitive performances. In this sense, our approach does not require complex pre-processing steps or highly resource-demanding training, but at the same time, it wisely leverages structural information on the face. Table 5.4 reports results that are more accurate than all methods with the exception of FSA-Caps, although the difference is on average only slightly above 1 degree. This small accuracy loss is counterbalanced by the benefits in terms of a smaller size, and it may be explained by the simplicity and compactness of our input: while nicely behaving in the majority of non-ambiguous situations, our sparse input is more severely influenced by occlusions, and missed or noisy detections.

Finally, Table 5.5 follows again Protocol P1, on a more complex test set, AFLW2000, where images are acquired in a less controlled environment. In this case, our methodology is reporting slightly worse results, but with a loss always less than 3 degrees on average. We

noticed this is due in particular to keypoint detection errors, as the synthetic data manipulation introduced artefacts.
To further evaluate the transfer potential of our approach we also report the result we obtained on the same test set when training the network on a related dataset (AFLW without the AFLW2000 section): the results are in this case comparable to the previous experiments.
We conclude by mentioning that we do not include in our comparison the approach in Fanelli et al.(G. Fanelli et al., 2013) since it uses the depth as input, and the methods Dlib (Kazemi and Sullivan, 2014) and FAN (Bulat and Tzimiropoulos, 2017) that solve a different problem (face alignment). Also, among the very recently proposed approaches, our analysis does not mention EVA-GCN (Xin et al., 2021) and KEPLER (Kumar et al., 2017) as they solve a different problem (jointly solving different tasks, one of them being head pose estimation), and 3DDFA (X. Zhu et al., 2019) that uses a richer input (image and 3D model).

5.6 Model Size and Inference Time

We now show the robustness of our method with respect to reductions of size, which may be needed when the available computational resources are very limited. More specifically, we analyse how the performance changes as we reduce the size of the model. We choose

Table 5.6 Comparison among models with different sizes (Protocol P1: 300W-LP train, BIWI test). β = neurons reduction factor (see text), MAE = Mean Absolute Error.

β	MAE	Parameters	MB
1	5.18	94031	0.385
0.6	5.43	37206	0.158
0.2	5.54	6006	0.032

300W-LP training and BIWI test (protocol P1) for their larger training and test sets and decrease the number of neurons in the fully connected layers so the backbone remains the same as proposed in the paper, while its size decreases. Given a reduction factor $\beta \in (0,1)$, we obtain a “reduced" version of our architecture by multiplying the original number of neurons in each layer (250, 200 and 150 in, respectively, the first, second and third layer) by β.

By varying β in the range $(0,1)$ we reduce the model size (the number of parameters) and thus also the number of sum and multiplication operations. Table 5.6 compares our baseline ($\beta = 1$) with two reduced models (overall size in MB up to $10\times$ smaller) causing a very limited degradation in the MAE (below 1 degree). This experiment highlights the possibility of further reducing the size of the architecture, with a very limited performance loss, if required by the system.

We finally briefly mention the computational performance of our method, which is an average of 142 fps (approximately an inference time of 7×10^{-3} s per frame). In the full inference pipeline, we should also consider the cost of running the key points detection, which depends on the specific approach. Empirical estimation of inference times can be found in (Lugaresi et al., 2019) for Openpose and Mediapipe, and in (K. Duan et al., 2019) for Centernet.

5.7 LAEO Experiments

Table 5.7 The performance of our method for LAEO detection on the UCO-LAEO dataset. The reported metrics are Precision, Recall, F1 score and AP estimated as in (M. Marín-Jiménez et al., 2020), $\tau = 0.93$.

Method	**PREC**	**REC**	**F**	**AP**
LAEO-Net (Marin-Jimenez et al., 2019)	–	–	–	0.80
LAEO-Net++ (M. Marín-Jiménez et al., 2020)	–	–	–	0.87
Gaze Pattern Rec. (F. Chang et al., 2023)	–	–	–	0.80
Baseline (Ours)	0.77	0.80	0.78	0.86
With uncertainty (Ours)	0.80	0.72	0.76	0.88

Fig. 5.8 Examples of LAEO detections. The arrows represent the head direction estimated by HHP-Net and projected on the image plane and are green if the corresponding person has been found involved in a LAEO. The prediction of our method for LAEO detection is reported in yellow and, in the case of LAEO, it specifies the identifier of the other interacting person. The identifiers are in red close to the subjects. In the last row, we report examples of failures, due to the ambiguities of the information on the image plane [source (Figari Tomenotti et al., 2024)].

We evaluate our method on the UCO-LAEO dataset (Marin-Jimenez et al., 2019), which includes sequences from four popular TV shows in the form of 129 shots of variable length. The annotation is provided at a frame level – *is there a pair of LAEO people in the frame?* – and at a pair level – i.e. each head pair is labelled as LAEO or not. The task we solve is a binary classification task: for each frame in the sequence, we consider all pairs of people detected in the frame and label them as LAEO or not using the method in Algorithm 1. Finally, a threshold τ, selected on the ROC curve of the training set, is used to detect the LAEO pairs.

We report in Fig. 5.9 examples to show how our LAEO measure smoothly changes during the interaction event.

We report in Table 5.7 the performance provided by our baseline method and the one incorporating the uncertainty on the test set. The results suggest that using the prior knowledge derived from the uncertainty allows us to significantly reduce the number of false positives (-6%, with a slight increase of the precision) to the price of a small reduction of true positive (-7%, with a small reduction of the recall). Overall, the uncertainty brings improvements as the AP increases ($+0.02$). As a reference, we also show in the table the results provided by (Marin-Jimenez et al., 2019; M. Marín-Jiménez et al., 2020; F. Chang et al., 2023).
Examples of the obtained results are reported in Fig. 5.8, where we show that our method is tolerant to the presence of more than 2 people, and to the scene variability.

5.8 Preliminary Experiments on Anticipation and HRI

(a) (b) (c) (d) (e)

Fig. 5.10 The five frames are taken from a *transport action* and a yellow circle is drawn on the table where the head is pointing. It is possible to see how the yellow circle anticipates the hand position in the whole action.

This section qualitatively mentions some works currently under development which target the usage of the HPP-Net in two different tasks. The first one is an assessment of how much the head orientation is informative when we want to anticipate or predict human motion or intentions. We find this research direction very promising and we have some preliminary cues about it thanks to our in-house dataset we collected (we describe it in Appendix B). In this preliminary experiment, we performed an anticipation experiment. We noticed that during an action performance, the head was turning before the hand even moved, indicating that the head direction has a prediction power in human motions. We show only some qualitative experiments Fig. 5.10 where we estimate where was the focus of attention (yellow circle) of the person in the video clip, and it is interesting to see that the head points towards the final direction of the movement before the movement even starts.

From the Fig. 5.10, it is quite evident that the head turns towards the action goal or the object needed to reach the goal quite before the action starts to be performed.

The second direction we are exploring is the implementation of our HHP-Net algorithm to perform both HPE and LAEO detection on the iCub Humanoid Robot (Natale et al., 2019)

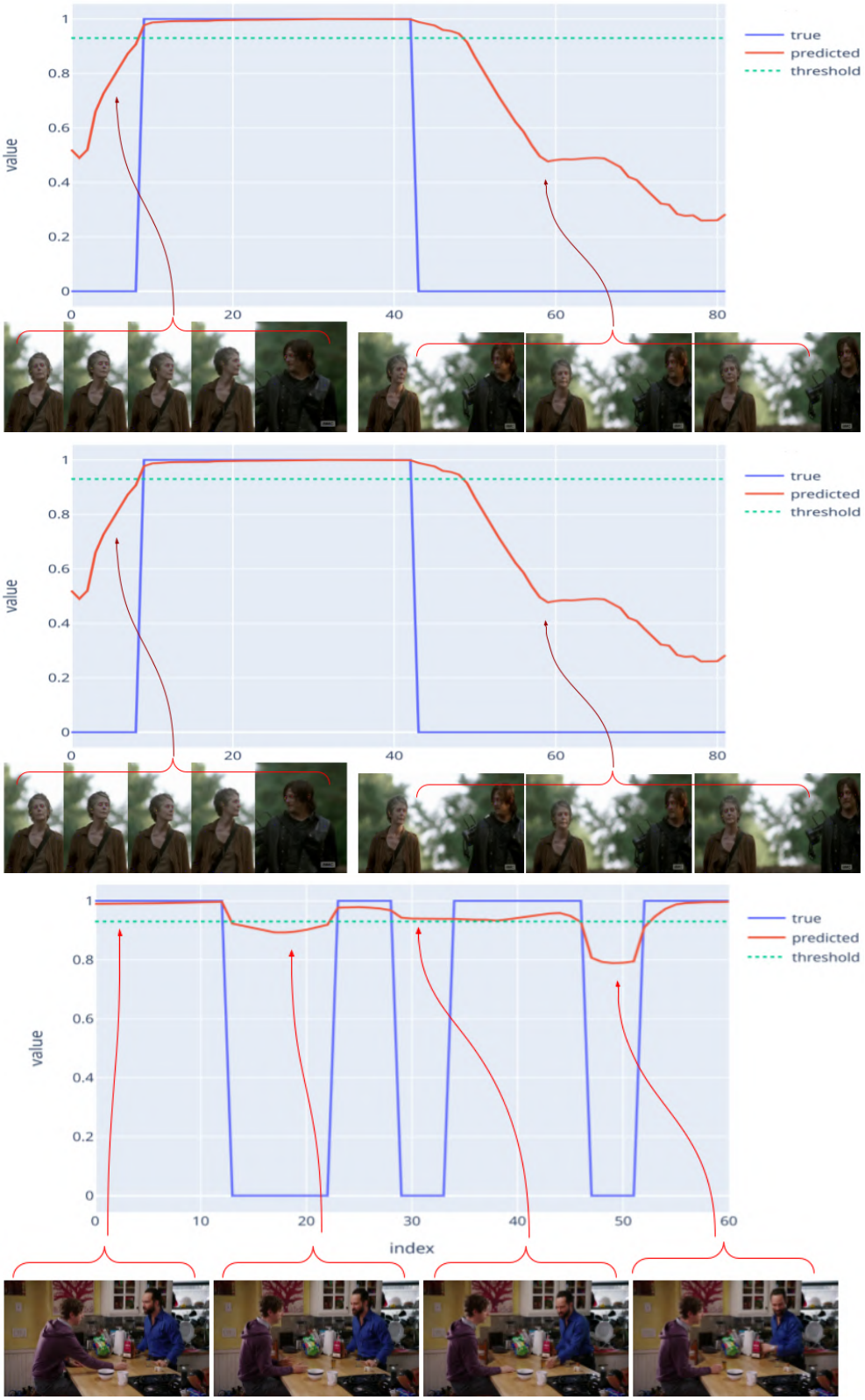


Fig. 5.9 Examples of LAEO measures over time with the corresponding frames of the video clip. In the plots we report in blue the ground truth, and in red our LAEO measure. In green, we mark the threshold we adopted [source (Figari Tomenotti et al., 2024)].

at the Italian Institute of Technology to perform experiments in the Human-Robot Interface domain. Where the robot is engaged with one person or in a a group of people and needs to detect the attention and the focus of the action which is involved into. For this, we cannot provide preliminary results.

5.9 Discussion

This first part presented two main contributions, HHP-Net for head pose estimation from keypoints and the LAEO detection.
The former contribution is a lightweight and fast neural network based on convolution operation to retrieve the head pose of people from the extracted keypoints. We believe that this tool has nice potential to be integrated into human pose detectors or put in cascade with them, also thanks to its interchangeability. Another interesting feature which is not present in other state-of-the-art approaches is the possibility of having an uncertainty value for each prediction thanks to the appropriate loss function. We have also shown how the uncertainty can be translated into degrees of uncertainty around the prediction.
The network achieves state-of-the-art results proposing a novel approach and introducing uncertainty, which also helps achieve higher accuracy. The extensive ablation study demonstrated the robustness of the number of keypoints in inputs and their correlation with the uncertainty.
The latter contribution is the LAEO algorithm, which is an easy but effective way of retrieving a 'Looking at each other' interaction on the image plane. Experimental results show how this measure changes smoothly if a pair of subjects in the foreground are targeted, but also its robustness when many people are present. Indeed, our methodology which estimates with great precision the head position helps in achieving a reliable value for LAEO interactions. This can be of course used as a proxy for more complex human interactions.
The major limitation in this regard is the fact that it works on the image plane and does not take into account the fact that two people can be at different depths inside a frame; moreover, the head pose estimation with keypoints suffers the presence of people completely turned 'inside' the frame, those which are seen only from the nape of the neck, because no keypoints are really visible. To overcome these issues, the first proposal is the usage of a very basic proxy looking at the height of people in the frame and estimating a mutual depth difference based on their differences in dimension (of course children can be an issue), otherwise using a CNN or a Diffusion model to estimate the depth inside images such as (H. Fu et al., no date; Ke et al., 2024), at a cost of having a much more heavy solution.
To overcome issues in the HHP-Net one way would be to use better and newer data for

the training phase in order to have a broader amount of angles at disposal. Of course, one intrinsic drawback is the fact that human pose detectors cannot (since now) characterise face poses which are seen from behind, from the nape of the neck. Or if they do, they did not fix any behaviour for occluded keypoints and that can lead to have random configuration.

Finally, as an additional outcome, we developed a demonstrative program which estimates the human pose and the head pose from videos (or images), everything at real-time speed, details are provided in Appendix A, a smaller and simpler demo with the same functionalities is also accessible in Hugging Face Spaces at this address https://huggingface.co/spaces/FedeFT/Head_Pose_Estimation_and_LAEO_computation.

Part II

Human Motion Representation & Recognition

Part II presents two contributions, both in the action recognition domain; the former focused purely on motion and kinematics features and with a peculiar attention to the representation of motion itself; and the latter centred on human-objects relationships in time, where graph neural networks are employed to extract information from the data.

Introduction to Part II

Action classification within the realm of computer vision is the main topic of this Part II. It stands as a pivotal challenge, seeking to endow machines with the ability to comprehend and categorise human activities from video data. This interdisciplinary field intersects computer science, machine learning, and image processing, striving to bridge the gap between raw visual information and meaningful interpretations of human actions. The significance of action classification extends across diverse domains, from surveillance and robotics to human-computer interaction, making it a central focus within the broader landscape of computer vision research (Al-Faris et al., 2020; Weinland et al., 2011; Gupta et al., 2021).

The essence of action classification lies in the extraction and analysis of temporal patterns and spatial configurations embedded within video sequences. To narrow our domain of investigation we decided to deal with human-centric actions and human motion representation. Moreover, the focus of the research is also at the edge of behavioural sciences from where we took inspiration for this work. The idea is not that machines should reason likewise humans, but we take inspiration from it, in order to replicate some successful patterns: our brain is the inspiration for many definitions of intelligence, and for sure a benchmark to asses if something else is intelligent.

The path we followed is to look in the cognitive science domain to see how humans perceive and understand actions, and temporal patterns and replicate it in a computer science - machine learning fashion. For example, humans perceive actions as the sum of shorter (in time) movements, and activities as a sum of different actions (Zacks and Swallow, 2007; Newtson and Engquist, 1976). So hierarchical representations are used. To help machines have a better understanding of semantics information, we preferred explicitly inputting this type of knowledge when possible, also to increase interpretability.

There is also a witness to meet other criteria that are becoming urgent and crucial in the community, such as the reduction of carbon footprint and the attention to privacy issues. Indeed, the attention in the methodology is not solely oriented towards performances but also to the above criteria.

Moreover, the motion alone cannot explain fully what is happening, because the same motion pattern can indicate different things if contextualised in different environments. Conversely, environmental cues serve to contextualise and refine the interpretation of motion, thereby limiting the range of plausible action categories. Guided by these principles and with a nod toward interpretable methodologies we decided that graphs-based representations have the power of abstraction and richness to be employed in this work.

Chapter 6

Literature Review

This chapter of the literature review faces the problem of presenting recent advances in action classification through skeleton data and the usage of graph neural networks for action classification. It encompasses several different foundational topics for the development of the following method.

6.1 Introduction

Part II of the thesis is dedicated to analysing human motion employing deep learning methodologies informed by insights from cognitive sciences and human perception, while also prioritising the explainability of these methodologies. In pursuit of explainability, we advocate for modularity in our solutions to enable finer control over the pipeline. The test bed of this work is its usage in an action classification task.
The second half of Part II delves into the analysis of scene understanding under compositional assumptions using graph representations. The choice is motivated by the belief that a modular approach can help the action representation task.
Thus, the two components of this Part II serve as foundational blocks for constructing a comprehensive action recognition pipeline, which focuses on human motions and on their surrounding environment, while employing distinct methodologies to represent different data types. For the time being the two algorithms are not interconnected, but it is easy to observe how in the future the two representation can talk to each other.

Part II is divided as follows:

- An introduction to Action Recognition in computer vision (current section)
- Literature Review of the major themes in the chapter Chapter 6

- Proposed methodology for the two tasks Chapter 7
- Experiments for the two tasks Chapter 8

6.2 Minimal Action Classification

The following literature review is quite extensive and covers a broad range of topics and not all are essential to understand the results and achievements of this work. This work sets mostly in the realm of action classification from skeleton data Section 6.2.1.2

6.2.1 Action Classification

In the ever-expanding realm of computer vision and artificial intelligence, the recognition and interpretation of human actions have emerged as a pivotal area of research with extensive implications in various domains, including surveillance, human-computer interaction, robotics, and healthcare. Human Action Classification, a subfield within computer vision, endeavours to impart machines with the ability to discern and categorise the diverse range of actions performed by individuals in visual data. This interdisciplinary pursuit draws upon computer science, signal processing, machine learning, and psychology to bridge the perceptual gap between human understanding and computational analysis of dynamic visual content.

The intricate nature of human actions, characterised by their temporal dynamics, variability, and context dependency, presents a formidable challenge for computational systems. Unlike static object recognition, the classification of human actions necessitates an understanding of motion patterns, spatial configurations, and the nuanced context in which actions unfold. Researchers in this field strive to develop algorithms and models that can not only accurately identify and categorise human actions but also generalise across diverse datasets and adapt to dynamic real-world scenarios.

One of the fundamental challenges in Human Action Classification lies in capturing the temporal evolution of actions. Traditional image classification methods are ill-suited to handle the dynamic nature of actions, prompting the development of novel approaches such as video-based analysis, temporal modelling, and spatiotemporal feature extraction. These techniques aim to encode the temporal dependencies inherent in actions, enabling machines to discern subtle variations and recognise actions across different speeds and durations.

Another critical aspect of Human Action Classification is addressing the inherent variability and context sensitivity of human actions. The same action can manifest differently based on individual style, environmental factors, or cultural nuances. Achieving robust classification, therefore, requires models that can adapt to these variabilities and discern the

essential features that characterise a given action. This has led to the exploration of deep learning architectures, ensemble methods, and attention mechanisms, aiming to capture both global and local contextual information in the visual data.

Furthermore, the practical applications of Human Action Classification are vast and impactful. Surveillance systems facilitate the automatic monitoring of public spaces, ensuring the rapid detection of anomalous activities and enhancing safety. In human-computer interaction, it enables more natural and intuitive interfaces, fostering a seamless interaction between humans and machines. Moreover, in healthcare, it holds the potential to assist in the early diagnosis and monitoring of motor disorders, contributing to improved patient care and rehabilitation.

As the field of Human Action Classification continues to evolve, researchers grapple with intricate challenges, seeking innovative solutions that push the boundaries of computational perception.

Here we decided to structure the literature review by dividing the approaches depending on the different data modalities exploited. The literature addresses this problem with mainly these modalities (Z. Sun et al., 2022): RGB, depth, skeletal data (2D and 3D), event stream, point cloud and infrared. Each of these can be processed alone or combined with others, and each one conveys a different and peculiar type of information. In the following paragraphs, we are going to see the pros and cons of some of the mentioned modalities, enlightening also some recent research from the literature. We are going to start with RGB data, proceeding with skeleton data, which are the main used throughout this work. To all the others we suggest starting from (Z. Sun et al., 2022; Kong and Y. Fu, 2022; Yi Zhu et al., 2020).

6.2.1.1 RGB data

The RGB modality is very powerful and a plethora of works based their methodology on it. In the pre-deep learning era, handcrafted features were in use, as in (Blank et al., 2005) where silhouettes -retrieved using space-time saliency, shape structures and orientations- are tracked through time. On the contrary, nowadays deep learning approaches are preferred and based on Convolutional Neural Networks (CNN). One of the first works is (Simonyan and Zisserman, 2014a) where a two-streams NN is used: one for the temporal information (motion) and the other for the spatial information (objects/people in the scene). The temporal stream is applied to the optical flow information retrieved as a B/W image between consecutive frames or also as a field of motion, this makes the recognition easier because “the network does not need to estimate motion implicitly” (Simonyan and Zisserman, 2014a). The spatial stream can be assimilated to a feature extractor for context and people detection. Then the information is aggregated and the networks are trained jointly with the class labels. Then

more advanced solutions were proposed using a multiscale resolution frame to have a more robust representation (L. Wang et al., 2015). The motivation is that the RGB modality is very rich but suffers from illumination and colour changes. So to enhance the invariance property, some solutions have been proposed. Another drawback of these types of networks is the computation expense of the optical flow calculation which has been tried to be mitigated in (B. Zhang et al., 2016) computing a motion vector which is less fine-grained but carries similar information. (B. Zhang et al., 2016) presented also a way to refine this type of additional information and to learn to mimic some optical flow characteristics.

In this interesting work (Feichtenhofer et al., 2016) different pooling and fusion strategies are compared for different purposes. Afterwards, the two streams' architecture evolved to become 3D CNNs, one of the first works presenting them is (Tran et al., 2015) where they found that small kernels (3x3) perform better than all the other solutions and that their 3D features were superior to any of the 2D CNN state of the art method. In tackling the challenge of memorising long-term relationships (Diba et al., 2017; Varol et al., 2017) presented some new approaches, at the cost of reducing the spatial resolution. And also in this context, to enhance the capacity of the models -but retaining simplicity in the training phase- many two streams 3D CNN were proposed such as (Carreira and Zisserman, 2017).

More recently also Transformer architectures for videos have taken the lead in the community and different works came out proposing to model the time dependence in different ways (we are referring only to solutions using RGB as inputs). In fact, in this paper (Bertasius et al., 2021) the research question is indeed if Action recognition can spare CNNs, in favour of Transformers. They created an architecture modelling attention in space and in time that achieves excellent results also in video longer than a minute, fully demonstrating their thesis.

A very new work (Shen Yan et al., 2022) is based on extracting tokens from the input video over multiple temporal durations. They use use transformer encoders of varying sizes to process each view, and they found that "it is better (in terms of accuracy/computation trade-offs) to use a smaller encoder (e.g. smaller hidden sizes and fewer layers) to represent the broader view of the video while an encoder with larger capacity is used to capture the details". However, Transformers are ubiquitous nowadays so to examine in depth the argument we suggest (Shabaninia et al., 2022; Ulhaq et al., 2022). Many other approaches focused on reducing the computation burden of the methods or enhancing the robustness against viewpoint variations and background clutters. Some others focused on extracting informative features from RGB frames, such as (Feichtenhofer et al., 2019) where a two-stream network is used to enrich the representation: one uses high-quality images as input but at a slow frame rate (2fps) and the other is instead focused on motion and injects in a

3D CNN 15fps but of low-quality images; then a late fusion approach is used and a fully connected classifier performs the classification stage.

6.2.1.2 Skeleton Data

Furthermore, a solution to overcome some of the challenges of RGB data is the usage of skeleton data. The skeleton modality is based on the usage of skeletal joints extracted by a Human Pose Estimator (see Section 2.6) on RGB data or acquired with motion capture systems. The former are nowadays ubiquitous and mobile versions running on smartphones are also available, however, they are viewpoints dependent. The latter, instead, produce really precise skeleton data and are illumination/view independent but are more difficult to acquire.

The advantages of skeletal data are multi-facets: the representation is scale-invariant (a skeleton has always the same topology despite the camera distance from the scene), it is simple and compact, it is robust against clothing and appearance in images and it is intrinsically privacy oriented. Early works on skeletal data for Human Action Recognition focus on their data structures as time series and use LSTM and Recurrent Network in general for their analysis (Du et al., 2015; J. Liu et al., 2017). Some works tried also to leverage the complexity using CNNs such as (Lea et al., 2017) where an encoder-decoder network has been explored, and even efficient solutions have been proposed (F. Yang et al., 2019) which run much more than real-time. In this domain, huge interest emerged from the usage of Graph Neural Networks (GNN) (X. Zhang et al., 2019). GNNs have been used extensively to study the problem, and because of the nature of skeleton data, this type of neural networks is very much suitable. Starting from (Sijie Yan et al., 2018) where a convolution over graphs is proposed and applied to neighbouring nodes till having a global representation of a skeleton. In this work also different types of partitioning are proposed in order to perform convolutions (e.g. partition by semantic neighbours of a node or by distance in the image plane from a chosen root node). Some other solutions based on GNN emerged where basically the network develops a spatial reasoning mechanism which passes information between neighbouring nodes to retrieve discriminative features (Si et al., 2018). Another example is (M. Li et al., 2019) where an encoder-decoder architecture processes skeleton data presented in two streams: one with structural links between joints (using a graph which uses the connection as the the natural body) and the other which connects correlated joints during some motions (e.g. feet and hands are correlated during walking). The newest approaches of a similar kind, having two branches, one to intercept temporal activity and the other for the body semantics have been implemented using transformers and attention methods. This recent work (Plizzari et al., 2021) proposes the usage of two different blocks of self-attention inside a Transformer, the former called Spatial Self-Attention which applies self-attention inside

each frame, and the latter Temporal Self-Attention which applies the self-attention between consecutive frames. Another work (Jiaxu Zhang et al., 2023) uses three types of attention to model the data and aggregate them with a transformer, the focus is on the representation part which is also highlighted by the attention matrices exposed and commented. Before the Transformer ubiquitous deployment, an interesting paper investigated different types and usages of attention layers (S. Cho et al., 2020), enlightening also some relationships of attention between joints and actions. Other works also focused on a slightly different task, focusing on some representation and high-level feature retrieval pipelines (Y. Chen et al., 2022). The skeleton data are of uttermost importance when the action is body-centred, but it does not provide enough information when some small object is involved, or the environment plays a pivotal role in recognising the action. Moreover, the data source may be noisy and it is by definition sparse. However, a very relevant aspect of skeleton data is its intrinsically privacy-oriented format which enables its usage in privacy-driven studies (like (Golda et al., 2022)), where the anonymity of the people should be preserved. This key feature will become crucial in the near future when this technology will be ubiquitous.

6.2.1.3 Benchmark Datasets

In the community, benchmark datasets play a crucial role in evaluating the performance of action classification algorithms. These datasets serve as standardised platforms, providing a diverse array of video clips capturing a multitude of human activities in various settings.

Prominent as one of the first big among these benchmark datasets is the UCF101 (Soomro et al., 2012) dataset, encompassing 101 action categories and over 13,000 video clips. It is composed of YouTube videos, and for contemporary standards the video resolution is low quality.

Similarly, the HMDB51 (Human Motion Database with 51 action classes) (Kuehne et al., 2011) dataset offers a comprehensive collection of realistic videos spanning 51 action categories. The inclusion of diverse actions such as sports, household activities, and interactions contributes to the dataset's richness and complexity. Both UCF101 and HMDB51 have become standard benchmarks, fostering the development and assessment of action classification methodologies.

The evolution of benchmark datasets has paralleled the advancements in action classification research, with newer datasets addressing the limitations and challenges posed by real-world scenarios. The Kinetics dataset, with its large-scale collection of high-quality videos, has emerged as a prominent benchmark, featuring over 400,000 video clips distributed across 600 action categories. The diversity and complexity of actions portrayed

in the Kinetics dataset mirror the intricacies of human behaviour, presenting a formidable challenge for contemporary action classification algorithms.

More recently also NTU RGB+D (Shahroudy et al., 2016; J. Liu et al., 2019) was released, and it has RGB and depth so it is possible to use for 3D human action recognition and it has two different sets: one with 60 action categories and another one, which is a superset f the previous, with 120 action categories. It has also multiple views: front view plus two other angles. The bigger version has 106 subject performing 120 different actions for a total of 114'000 videos; also 3D skeletal data has been released.

The final dataset under discussion, BABEL, merits attention despite being less famous and adopted than the others. BABEL comprises 3D skeletal data designed for action recognition, with a version featuring 120 classes to align somewhat with NTU. However, this version exhibits a significant class imbalance issue, as detailed in Chapter 8. The underrepresentation of BABEL in scholarly works may stem from its novelty compared to other datasets and its lack of RGB video coordinate sets. Consequently, BABEL is exclusively compatible with monomodal pipelines utilizing skeletal data. Furthermore, its pronounced imbalance, coupled with instances that may be associated with multiple labels, presents a formidable obstacle to its widespread adoption.

As we navigate through this comprehensive exploration of action classification in computer vision using video data, the subsequent sections will delve into the methodologies, techniques, and recent advancements that propel this dynamic field forward. The major focus of the following section will be on Action Classification starting from RGB and skeleton data; then we will explore the literature encompassing action classification through primitives of motion.

6.2.2 Primitives of Motion Decomposition

Deep learning has revolutionised the landscape of action recognition, providing powerful tools for understanding and replicating complex human activities. As we strive to enhance the capabilities of artificial intelligence in interpreting human actions, the need for a nuanced understanding of event segmentation and the utilisation of primitives becomes paramount to overcome some of the open problems, such as long-term dependencies, and multi-granularity recognition.

Within the broader landscape of action recognition, specific investigations have probed into hand-actions, unveiling the intricate reasoning processes that underscore how humans predict actions through a grammar structure. In light of these advancements, this section unravels the symbiotic relationship between deep learning, event segmentation, and the utilisation of primitives, offering insights that are crucial for advancing our understanding

and implementation of artificial intelligence in the realm of human action recognition (Lan et al., 2015) within long-term activities, retrieval (Ikizler and Forsyth, 2008) and generation.

This branch studies the intricate process of human action recognition (Lillo et al., 2014), focusing on the integration of behavioural sciences, cognitive sciences, and event segmentation. Event boundaries, identified as hierarchically structured in space and time, play a pivotal role in understanding and replicating complex actions (Zacks and Swallow, 2007). The study presented in (Newtson and Engquist, 1976) involved participants watching a movie and segmenting meaningful events, revealing the reliability of judgements across viewers and within viewers over time (Newtson and Engquist, 1976; Speer et al., 2003). Successful segmentation extends beyond identifying event boundaries; it necessitates tracking how fine-grained events group into larger meaningful units (Hard et al., 2006; Lozano et al., 2006). Recent research underscores the significance of this grouping for learning new activities and retaining details of recent experiences, "those individuals who are better able to segment an activity into events are better able to remember it later"(Hard et al., 2006).

The automatic nature of segmentation in the human brain is emphasised, where sensory features are segmented bottom-up, while conceptual features, such as goal-oriented actions, are segmented top-down (Zacks et al., 2001). Examining the cognitive aspects, the text delves into the understanding, recognition, and replication of complex actions. Interestingly, brain regions traditionally associated with language production, like Broca's Area, are identified as crucial for action planning (Fogassi et al., 2005). In (Summers-Stay et al., 2012), it is suggested that a fundamental, innate representation is central to both language and activity understanding, requiring a generative grammar for learning actions by example. The minimalist program, as proposed by Chomsky, aims to uncover such a universal generative grammar with broader cognitive implications.

Examining in depth (Summers-Stay et al., 2012) we find that the core importance of action segmentation is highlighted by defining an action's initiation when a hand establishes contact with an object and concluding when the contact ceases. The recognition process of this work involves constructing a tree, where each leaf corresponds to a contact event with the root node. Recognition relies on measuring similarity using the Edit distance. The text introduces a novel dataset featuring five complex manipulation activities, each decomposable into simpler actions, incorporating RGB and depth data from two different sensors. Noteworthy works in the field have focused on specific action types, such as hand-actions, revealing how humans predict actions through reasoning with a grammar structure (Wörgötter et al., 2020b).

Moving now towards a more vision-centred scenario, we can find (Nair et al., 2020). The challenge tackled by this work is double-faced, on one hand, they investigated how a machine can see the similarity in actions described only by keypoints motions. And, on the

other hand, they compared the reliability of the machine and humans in solving this task. In this context, action primitives are conceived as kinematics descriptors (such as velocity and its derivatives), and they are learnt automatically using dictionary learning techniques. The actions used in the study are all hand actions, like squeezing a lemon, cleaning the table, cutting bread etc. An interesting approach using Hidden Markov Models for primitives detection and learning is proposed in (Kulić et al., 2012). Motion Capture movements are acquired and an online algorithm for segmentation is used (Kulic and Nakamura, 2008). In cascade, a Hidden Markov Model models the transition probabilities between each primitive of motion. Their solution can be used for action recognition but also generation. As we have seen since now, in the literature there is no unitary definition of motion primitive: some works interpret them as a semantic concept (Wörgötter et al., 2020b), others more as a kinematic structure of the motion. So, primitives are hierarchical entities which are context-dependent in the literature. With the advent of modern deep learning techniques, works focused on automatically learning them in an unsupervised or self-supervised fashion. Quite novel research (Saunders et al., 2021) relies on a mixture of experts (Jacobs et al., 1991) made of transformer encoders which learn different motion patterns and weights to achieve a translation goal in a bigger network (from text to sign language representation). In this case, primitives can be seen as a subset of cheremes (the basic units of sign language) and by combining them it is possible to obtain a single chereme or sentences.

In conclusion, we can say that the exploitation of Primitives of motion has been deeply explored in the Robotics research domain under a motor control perspective (Dessalene et al., 2023; Schaal, 2006) but under a perception perspective there are still many unexplored avenues to investigate.

6.3 Graph Neural Networks

Graphs are a broad matter of study in machine learning in several domains of application; there exist many theoretical papers studying how to extend usual deep learning networks to graph-structured data; for example (Defferrard et al., 2016) proposes a convolution on graphs focused on having a constant complexity regardless of the input graph structure. Other works developed Graph Convolution from a spectral viewpoint (Bruna et al., 2013) and others starting from a spatial viewpoint (Kipf and Welling, 2016). In more recent studies, encouraged by Transformer architecture for NLP and images, in (X. Fan et al., 2020) they develop an attention layer for graphs, showing its usefulness in a graph classification task. Despite many methods focusing on graph classification (chemical graphs, etc) or node feature predictions, in this section, we will focus on graphs in computer vision applications

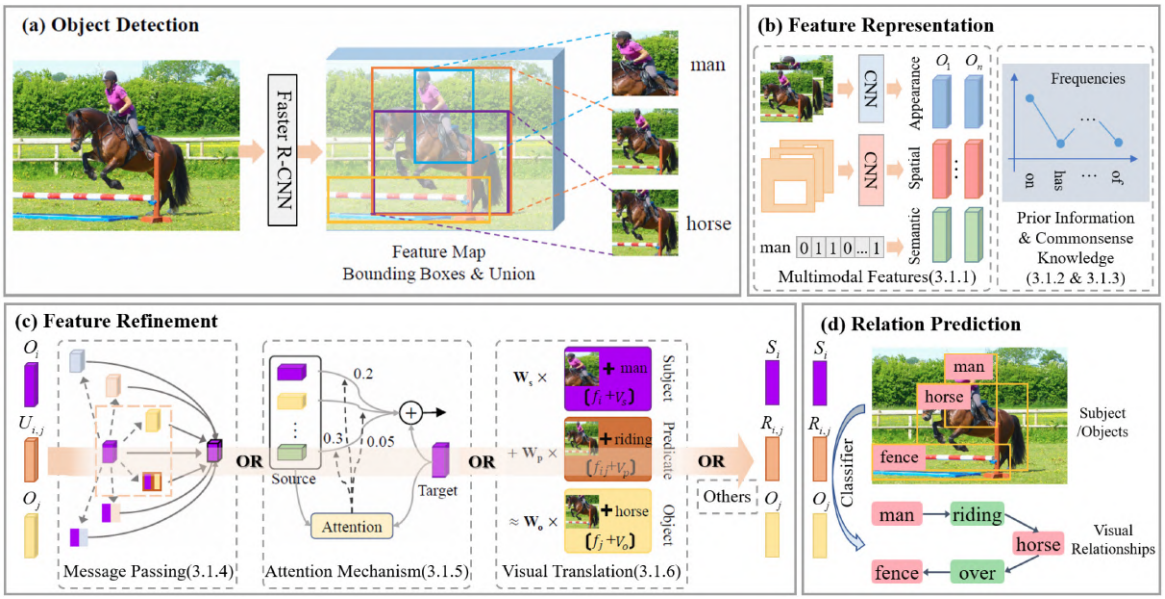


Fig. 6.1 Example of a general pipeline of Scene Graph Generation [source (G. Zhu et al., 2022)].

and restrict also to our domain of application: action recognition and scene understanding. Scene graph generation is the task of starting from an RGB image and transforming it into an abstract representation having bounding boxes representing objects of interest and connecting them with relationships. Then this becomes a graph structure. "From the point of view of graph theory, a scene graph is a directed graph with three types of nodes: object, attribute, and relation. However, for the convenience of semantic expression, a node of a scene graph is seen as an object with all its attributes, while the relation is called an edge" (G. Zhu et al., 2022). There are two approaches to scene graph generation. The former is a two-stage method (general example in Fig. 6.1), e.g.(T. Chen et al., 2019), where one network such as a Fast RCNN (Girshick, 2015) extracts object bounding boxes and labels, and a second one regresses the relationships between them. The latter approach is a one-shot method, where one neural network does all in one step, regressing objects and relationships as well as positions, two examples are (Cong et al., 2023; Hengyue Liu et al., 2021). The two-stage methods rely on different characteristics and use several mechanisms: multimodal feature aggregations to ensemble the descriptions of different objects from the object detector. In fact, features may be spatial, semantic and context; in this respect (Yaohui Zhu et al., 2017) the authors investigate how spatial features and different positions of objects can aid the relationship reconstruction. To understand this we should highlight that the spatial distribution of objects in the scene not only reflects their positions but also their structural information. Another approach (Ji Zhang et al., 2018) investigated how to combine visual,

spatial and semantic features and show explicitly how each feature contributes to the final prediction. Other methods, such as (G. Yin et al., 2018), stated that using also context around objects or ensembling multi-object features helps to have better representations and extracting more precise relationships. On this problem, even extending dataset dimensionality does not give a huge boost to some of the categories involved, because of the natural long-tailed distributions of the relationship occurrences (R. Yu et al., 2017). So (Zellers et al., 2018) tried to look for statistical features able to disambiguate and regularise the recognition task. But other ideas come from language priors, so here (Liao et al., 2017) thinking that linked objects should find sense in the semantic world and not just be similar or near in the embedding space. Other works tried to extend this to the open-set vocabulary setting, such as (Y. Zhang et al., 2023). One of the biggest datasets in terms of variety of relationships is the Visual Genome dataset (Krishna et al., 2017), but many others are present. However, some are composed of single images (Kuznetsova et al., 2020) and others of videos(Krishna et al., 2017), so having or lacking a temporal component linking the data. To go deeper into the topic and find related material we suggest looking at these two comprehensive works (X. Chang et al., 2021) (G. Zhu et al., 2022).

Concerning the action recognition from graphs, there exists one main way: using the human keypoints and bones to define a graph and applying a Graph Convolutional Network (GCN) on this representation in space and over time, but we have already seen the major works on this specific topic in Section 6.2.1.2. Very few other works focus on recognising actions from graph descriptions taken from scene graphs. (Arnab et al., 2021) proposes the use of a GCN with a message-passing behaviour which takes in input features from a 3D CNN. It uses an explicit representation of objects when available and implicit otherwise. They test their pipeline on spatio-temporal action detection on the AVA dataset (C. Gu et al., 2018) and on video scene-graph classification on the Genome dataset. Another work developing a similar technique in a different setting is (Materzynska et al., 2020), where the focus is on modelling the evolution in time between subjects and objects and their geometric relations. The task they face is the recognition of manipulatory actions.

Chapter 7

Proposed Methodology

The methodology chapter is two-folded: the first half presents the work about human motion representation from primitives of motion; the second half presents the action recognition pipeline focused on human-object interactions modelled as scene graphs and processed using graph neural networks.

7.1 MOSAIC: Minimal Action Classification

7.1.1 Introduction

In this chapter, we present a core contribution of the thesis which is about action representation and classification. The idea comes from behavioural sciences, cognitive science and computer science literature and aims to demonstrate how using human-like structures of data can be beneficial to machine intelligence. The goal is to perform action classification using a new feature space where primitives of motion are emphasised and used as building blocks for action representation. In comparison to the literature exposed in Chapter 6 our method provides some advantages because it completely relies on automatic feature extraction with few hyper-parameters for the primitives extraction. This task is based on two assumptions: actions can be decomposed in primitives either in a kinetics sense (Nair et al., 2020) or in a grammar structure fashion(Wörgötter et al., 2020a)); and, every action has a unique signature if described with primitives. These assumptions are encoded in our problem in the data manipulation part: the primitives' decomposition is obtained by dividing (along the temporal axis) data into several smaller clips and encoding them to find similar movements. Furthermore, the fact that each action has a unique signature in our representation means that we can perform action recognition in this new space we created. But why do we need to encode them or map them in a new and different space? Because with the skeleton data (see

an example in Fig. 7.1) temporally divided, we need to make a 'dictionary' of primitives; in other words, we would like to find a space where similar parts of the movements are encoded nearby in the space and dissimilar encoded far-apart. Indeed, we use a neural network that maps similar data close to one another in the new feature space.

One fundamental aspect we need to focus is the amount of temporal information we need to encode in a primitive. Modern architectures are very likely to focus on the wrong and uninteresting parts of images or videos for classification purposes, however, feeding them with already structured and meaningful information may help to improve performance in different tasks. In this way, the network concentrates only on high level task and not more on pixel-level feature retrieval. In some sense it is a way to use 'deeper' architecture but with the agility of modularity. First attempts may deal with using the same architecture of this paper and change input modality and for example, exploits: optical flow, pose estimated via keypoints, context elements such objects.

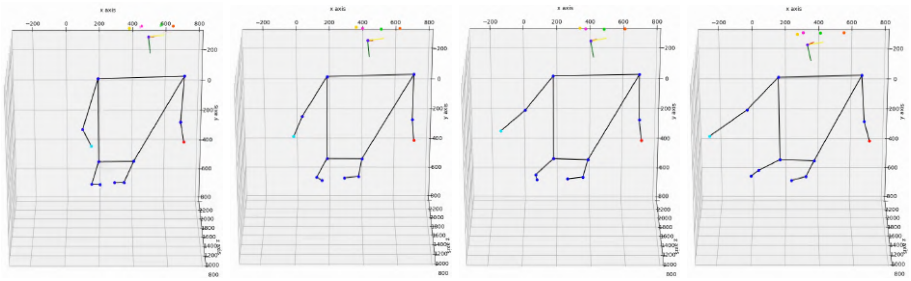


Fig. 7.1 Example of a skeleton sequence from an upper-body action from the dataset in Appendix B. Each skeleton represents a different frame.

Our methodology is focused on the classification of actions as a test bed for our representation. The pipeline can be divided into two consecutive parts. The first one is the Primitives extraction where the data are pre-processed and divided into windows and finally passed through a representation model which is an autoencoder. In the second half the representation of each window is extracted in the latent representation of the autoencoder and used as the input token of a Transformer network. The pipeline is not end-to-end because it needs the representation learning part to be performed offline. However, if the representation is powerful enough, the network trained on some data can easily be extended towards new unseen data. Moreover, the representation retrieved in this way can also lead to experiments of motion generation, but they are out of the scope of this dissertation.

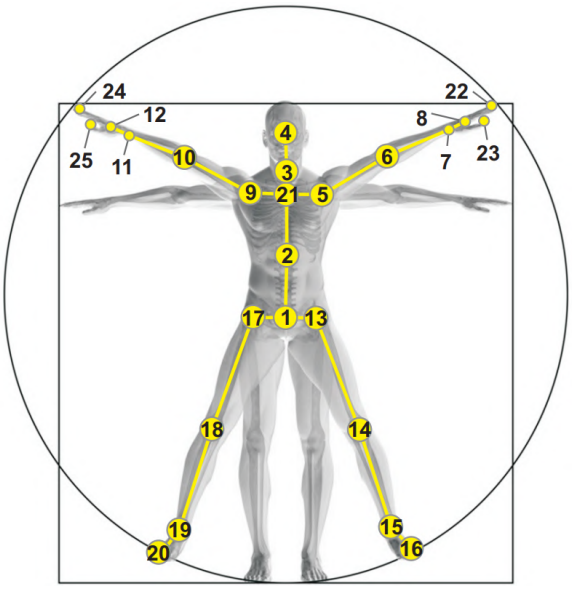


Fig. 7.2 Illustration of the 25 joints of the NTU format. Configuration of 25 body joints. The labels of the joints are: 1-base of the spine 2-middle of the spine 3-neck 4-head 5-left shoulder 6-left elbow 7-left wrist 8- left hand 9-right shoulder 10-right elbow 11-right wrist 12- right-hand 13-left hip 14-left knee 15-left ankle 16-left foot 17- right hip 18-right knee 19-right ankle 20-right foot 21-spine 22- tip of the left-hand 23-left thumb 24-tip of the right hand 25- right thumb [source (Shahroudy et al., 2016)].

7.1.2 Problem Formulation

The aim is to recognise the action category starting from 3D (or 2D) skeleton data. In the following, we will call video or sequence the entire sequence of skeleton data composing one action, i.e. coupled with one label.

More formally, we start from a set of N skeleton streams associated with an action label, $\{(\mathbf{v}_i, y_i)\}_{i=1}^{N}$, with $y_i \in \{1, ..., C\}$, C is the number of classes. For each i, $\mathbf{v}_i \in \mathbb{R}^{T \times J \times d}$ with T the video length in frames, J the number of skeleton joints (e.g. 25 if employing the NTU format (Shahroudy et al., 2016), see Fig. 7.2), living in a space of dimensionality d. According to (Shahroudy et al., 2016) we transform the joint's location to a new reference system with origin in the *middle of the spine* joint.

The pre-processing of the joint has been carried on according to the NTU process, so the joints from the camera coordinate system have been translated to the body coordinate system with its origin on the "middle of the spine" joint. The labels we work with are at the sequence level and are assumed as ground truth, except stated differently. The first step of the pipeline

is to divide the input into temporal segments and represent them thanks to an autoncoder compression, and then classify them thanks to a Transformer, as described in the following.

7.1.3 Primitives of Motions

The theoretical idea behind this approach is well expressed in some of the works cited in Section 6.2.2, for example in (Nair et al., 2020; Kulić et al., 2012; Kulic and Nakamura, 2008) where motion primitives are established as kinematics primitives of the human body. In this sense, we define what should be a primitive for us, in a general sense, and we would investigate if it can demonstrate the actual utility of them in computer vision systems. Undergoing this operation we selected human action recognition as a downstream task to asses our representation. The assumption is that if the representation is as rich and complete as possible, it can be an advantage to a human recognition system.

Definition 7.1.1 (Primitives of motion)**.** A primitive of motion is defined as a short piece of motion such that putting many primitives in sequence defines an action.

In this respect, we are really close to (Zacks et al., 2001), because kinetics primitives can be seen as a type of sensory feature. The kinematics can be described using positions and all its derivatives. In our work, we chose to start from the velocity because it is position invariant and gives enough information to be used. The final point would be to use primitives of adaptive length, but let us start with a fixed-length algorithm. For now let us assume to have a fixed time window of duration $\tau < T$, where T is the video duration. We divide our sequence of skeleton frames in $T/\tau = W$, W is the number of windows or clips in each video. To express mathematically what we do:

$$\mathbf{v}_i = \{\{\mathbf{c}_{j,i}\}_{j=1}^{W}\}_{i=1}^{N}$$

N is the total sample number, W is the number of windows, each spanning τ frames. The choice of the window size W depends also on the sampling frequency of our data; usually, selecting a window of length 1 second is a good compromise between informative content and size; moreover, at the level of common knowledge, human movements have a speed which permits an action evolution from fractions of a second to few seconds(Cieślik and Łopatka, 2022; Gaveau and Papaxanthis, 2011; Allingham et al., 2021).

7.1.4 Encoding

The following step of the pipeline is the usage of a variational autoencoder which enables us to project our clips in a feature space where the similarities between windows should be

highlighted. The VAE consists of six 1D convolutional layers, each strategically configured with channel dimensions of 128, 128, 256, 512, 256, and 256 respectively. This configuration shrinks the input data into a latent space of 128 dimensions, facilitating effective feature representation. Following the convolutional layers, the encoder structure integrates a reshaping function to prepare the data for a fully connected layer, subsequently feeding into μ and σ layers implemented as linear layers. The architectural decisions are meticulously crafted to enhance information extraction across temporal dimensions, employing a uniform kernel size of dimension 3 for all layers. Our VAE model encompasses approximately 14 million parameters.

The exact procedure (see Fig. 7.3) is the encoding of a single clip, to build a hidden space built around a limited temporal part of the motion and not representing an entire action. In this regard, it is not easy to look at the feature space and its clusters, because it does not cluster 'following' action labels. Indeed, each action should be composed of a series of these embeddings. To use an example, we are building a sort of dictionary where each embedding is a word we will later use to build a sentence: our motion sequence. Writing this passage explicitly we can state that each clip is encoded in the latent space

$$\mathbf{e}_{j,i} = f(\mathbf{c}_{j,i}) \tag{7.1}$$

recalling that $f(\cdot)$ and $g(\cdot)$ are respectively the encoder and decoder functions, and with a slight abuse of notation we can write the loss as.

$$Loss = MSE(\mathbf{c}_{j,i}, g(f(\mathbf{c}_{j,i}))) + \beta \cdot KL(N(\mu,\sigma)||N(0,1)) \tag{7.2}$$

where MSE is the reconstruction parameter and it stands for Mean Squared Error $\sum(\hat{\mathbf{c}}_{j,i} - \mathbf{c}_{j,i})^2$ and KL is the Kullback–Leibler divergence (see Section 2.2.2. The β is a hyper-parameter which varies across the epochs and helps to focus on the reconstruction at the beginning of training ($\beta = 0$) and then linearly approaches 1, weighting equally the two loss contributions.

7.1.5 The Transformer

The transformer architecture has been chosen because of its nice properties in dealing with temporal data. Let us start the section by recalling one of the main points and strengths of the method which is the hierarchical approach and its compositional structure. So, likewise human language –where Transformers have their top performances (Gasparetto et al., 2022) and usages– we would like to think of human actions in a similar way. Our primitives can be seen as words, and their temporal concatenation as sentences. This is a metaphor we will use also in the following to explain some passages, just to make them clearer for the reader.

In our setting, see Fig. 7.4, we use only a Transformer Encoder with a classification head working on a CLS token. So, the clips embedding are taken and a CLS token is appended at the beginning of the sequence, afterwards, everything passes through the positional encoding and it is finally processed by 3 encoder layers, each one with 16 heads of attention. The dimensionality of each token is the same as the latent space of the VAE network. At the end of the Transformer, a fully connected network with 64 neurons and ReLu -as activation- classifies the entire sequence taking in input only the CLS token.
Formally, $[z_{i,1}, \cdots, z_{i,n_k}]$ are given in input of the Transformer, pre-appending a CLS token and using the positional encoding. We tested several loss functions and reported results for each of them in the following. The first is a standard cross-entropy loss:

$$CE = -\sum_{i=0}^{C} p_i log(f(v)_i) = -log(f(v)_i) \tag{7.3}$$

The first part is the cross entropy for a retrieved distribution $f(v)$ versus the real one p_i; the second part we assume to have the labels encoded in one hot fashion, so only one element each time is different from zero.

The second is a balanced cross-entropy, where each class is weighted by its presence in the training set.

$$((1-\beta)/(1-\beta^n)) \cdot CE \tag{7.4}$$

where *beta* is a hyper-parameter, and n is the number of samples per class. This loss is useful when training with imbalanced datasets.

Last we tried the focal loss, introduced here (Lin et al., 2017a).

$$Focalloss = -\alpha_t \cdot (1-p_t)^{\gamma} \cdot log(p_t) \tag{7.5}$$

where p_t is the probability of being classified to the true class; $p_t = p$ (if true class), otherwise $p_t = 1-p$. When $\gamma = 0$ the focal loss reduces to the CE loss. This loss, similar to the balanced one, reduces the loss contribution from easy examples (by the factor $1-p_t^{\gamma}$) and increases the importance of correcting misclassified examples.

Concluding on the architecture, in training mode, the VAE is a full Encoder-Decoder structure, and in inference, we use only the Encoder part, as shown in Fig. 7.4. Here, the structure of the Transformer is more detailed with respect to Fig. 7.3.

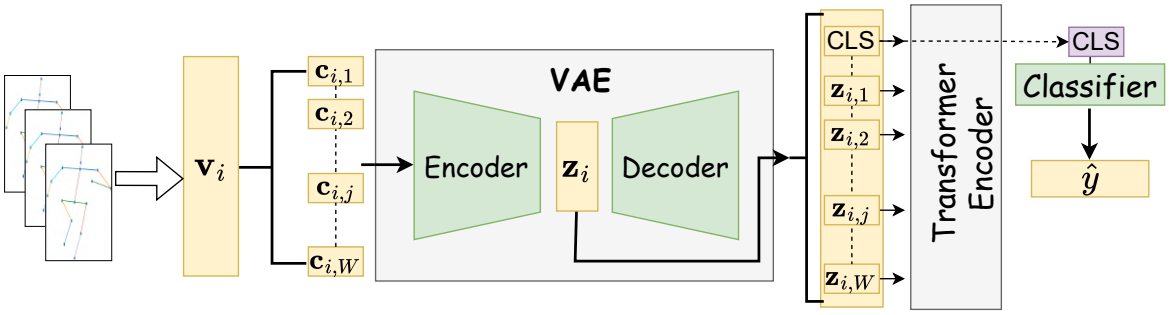


Fig. 7.3 Overview of the full pipeline composing MOSAIC. From left to write, kinematics primitives are extracted from skeletons and a VAE encodes the representations in its latent space. Then, we sample the space to 'rebuild' an entire sequence representing an action and it is passed to the Transformer for the classification.

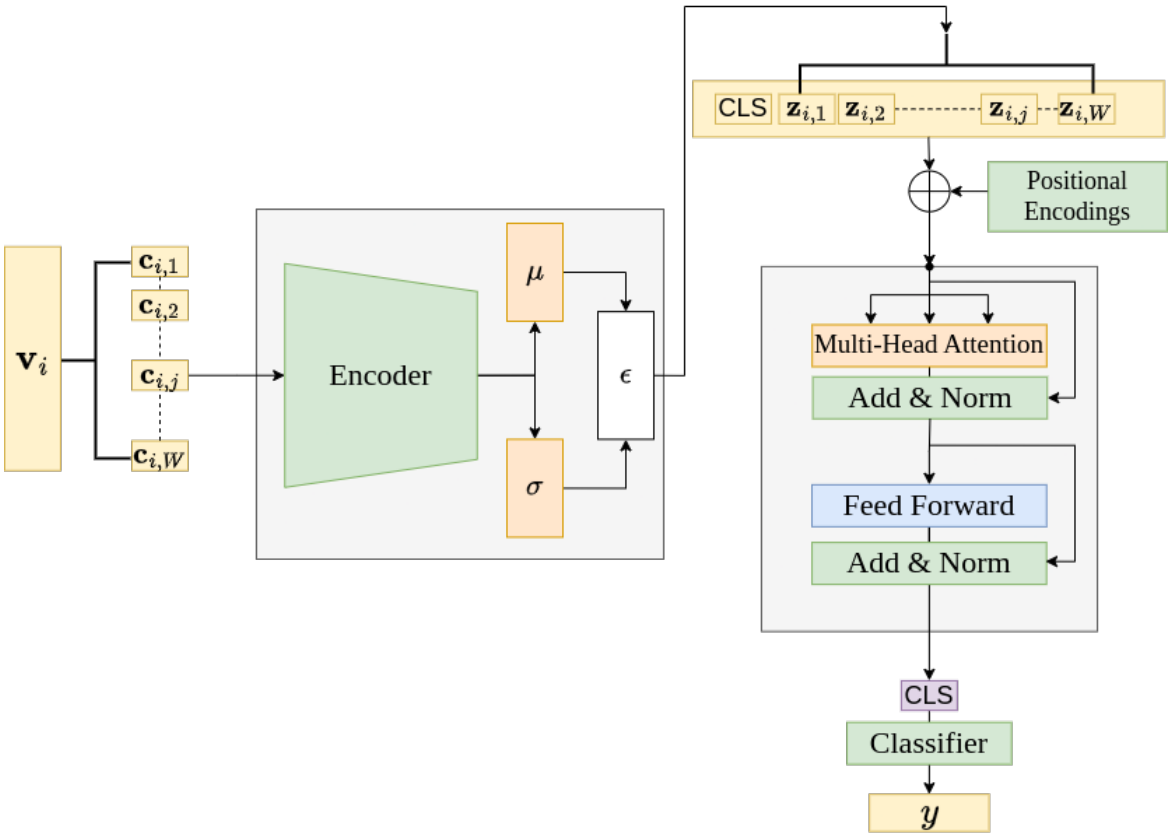


Fig. 7.4 MOSAIC pipeline in Inference mode, VAE encoder is frozen. From left to right: the data are divided in windows and input to the encoder one by one. Then the VAE representation is sampled and the entire action is 'rebuilt' in the form of a sequence of embeddings and ingested in the Transformer encoder and lastly classified.

7.1.6 Kinematics Features

One of the most challenging parts is the data shape and the features chosen to be the input of our pipeline. Because, for the validation part of this methodology, one of the main point is the assessment of the usage of different kinematics features as descriptors, as we will see in the experimental section. We formally stated this data shaping part as the application of $\Phi(\cdot)$ on our input data. It is a pre-processing function that can map a clip $\mathbf{c}_{j,i}$ or a video $\mathbf{v}_i$ into different kinematic spaces. So the input of our VAE can be represented as $\Phi(\mathbf{v}_i)\}_{i=1}^{N}$ and then divided into clips. When Φ is the identity function, the embedded primitive representations are learned from sequences of joint position. Alternatively, one can use the sequence of the instantaneous velocities of each joint, either considering their different components or magnitude.

We started using as input sample the magnitude of each keypoint in time, this is the matrix V:

$$\begin{bmatrix} v_1^1 & v_1^2 & \dots & v_1^T \\ v_2^1 & v_2^2 & \dots & v_2^T \\ \vdots & \ddots & & \\ v_K^1 & v_K^2 & \dots & v_K^T \end{bmatrix} \tag{7.6}$$

where $v = \sqrt{x^2+y^2+z^2}$, and $k \in [1,25]$ are the keypoints, and $t \in [0,T]$ indicates the frames: so each action is represented as a matrix of 25 time-series (one each joint).
Then we define the data matrix $\mathbf{V}$ -where each $\mathbf{v}$ is a column vector $\mathbf{v} = [x,y,z]^T$- as

$$\begin{bmatrix} \mathbf{v}_1^1 & \mathbf{v}_1^2 & \dots & \mathbf{v}_1^T \\ \mathbf{v}_2^1 & \mathbf{v}_2^2 & \dots & \mathbf{v}_2^T \\ \vdots & \ddots & & \\ \mathbf{v}_K^1 & \mathbf{v}_K^2 & \dots & \mathbf{v}_K^T \end{bmatrix} \tag{7.7}$$

so, expanding this notation $\mathbf{v} = [x,y,z]^T$ the previous matrix $\mathbf{V}$ becomes:

$$\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^T \\ y_1^1 & y_1^2 & \dots & y_1^T \\ z_1^1 & z_1^2 & \dots & z_1^T \\ \vdots & \ddots & & \\ z_K^1 & z_K^2 & \dots & z_K^T \end{bmatrix} \tag{7.8}$$

We define this data matrix as **P** The same can be done by combining the modulus of the velocity with the positions by axis:

$$\begin{bmatrix} v_1^1 & v_1^2 & \dots & v_1^T \\ x_1^1 & x_1^2 & \dots & x_1^T \\ y_1^1 & y_1^2 & \dots & y_1^T \\ z_1^1 & z_1^2 & \dots & z_1^T \\ \vdots & \ddots & & \\ z_K^1 & z_K^2 & \dots & z_K^T \end{bmatrix} \tag{7.9}$$

Using the notation for block matrices we define this matrix as [**V**|**P**], which stands for velocity and positions. Of course, it is also possible to do the same using the speeds instead of the positions for the three axes, so substituting the derivatives for each axis, in this case doing this substitution $x \to \dot{x}$ we can also define two new matrices $\mathbf{V}_{xyz}$ and [**VV**]. The former is composed of the derivatives along each axis, and the latter is similar to [**V**|**P**] where the velocity components have replaced the positions.

7.2 ACROSS: Action Classification via Human-Context Information

This section [1] marks the beginning of an exploratory endeavour aimed at enhancing Action Classification pipelines through the integration of graph representations. The objective is to harness environmental information and the interrelations among objects and people to augment the action classification process. Up to this point, our focus has been primarily on recognising actions solely based on motion features. However, a significant limitation arises from the inherent ambiguity concerning the location and context of action execution. Thus, this segment of the thesis complements the Minimal Action Classification sections, mutually enriching each other's insights, albeit lacking direct intercommunication at present.

7.2.1 Problem Formulation

We defined this task as an Action or Activity Classification problem, having as input features and as output an action label, however, we decided to start from an underused (or not used at all) data format: scene graphs evolving in time. In the following, we will use the term Actions and Activity interchangeably because what differentiates one from the other is the temporal duration of the two, and it strictly depends on the data used. Because our approach is independent to the temporal extent both names can be used. We found a dataset built for Image Scene Understanding and graph generation but annotated also with action categories and we tailored it to our needs.

The dataset is the Homage Dataset (Rai et al., 2021) which is a large-scale multi-view video database of indoor daily activities. The annotations in this dataset are dual-faceted: object-human relationships and action labels at two different granularities. The relationship annotation encompasses a human and the objects around him as bounding boxes and relationships between the human and objects. This annotation can be described as triplets such as person, object, verb, e.g. person, chair, sitting on. The action annotations instead have a video-level action or task performed during that clip and more fine-grained action (e.g. take the iron, take the iron board etc). For our task, we used the bounding boxes and relationships as data to build our scheme graph for each frame and then we passed the graph inside our GCN training to predict the action label, at the video level. Describing our framework more in-depth, individual subjects and objects, persistently tracked over time, serve as nodes, while the associations between them are encoded as relational links. However, to effectively integrate temporal information into our graph representation, we deemed it necessary to

[1] undertaken in collaboration with Clara Mouawad during her Master's Thesis internship

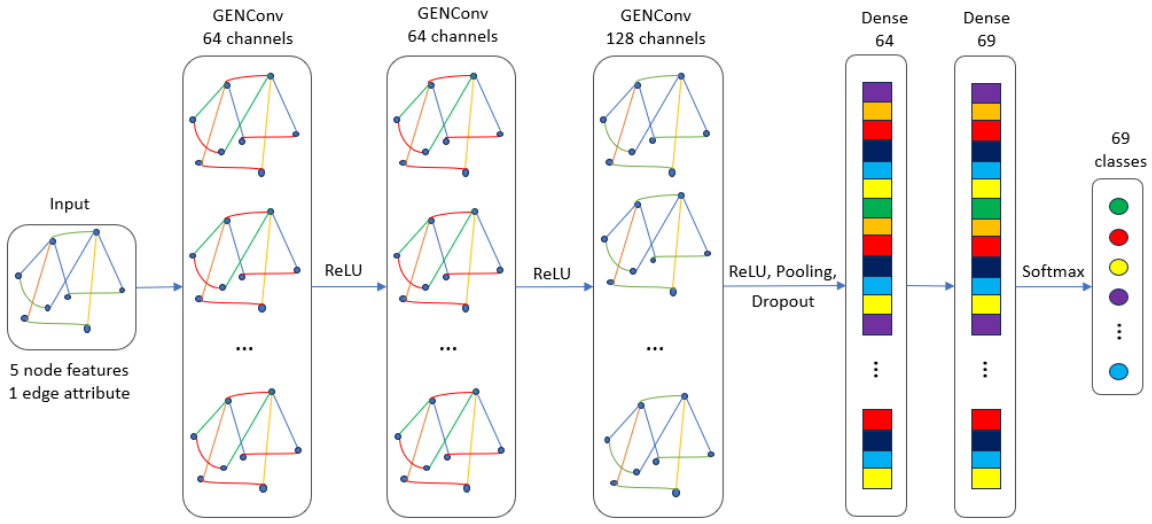


Fig. 7.5 Overview of the ACROSS method. On the left there is the input graph, then the graph convolutional layers are drawn and finally, two fully connected layers for the classification are stacked together.

incorporate temporal links alongside relational links. Temporal links are inter-frame links connecting the same instance/object with itself during the evolution of the scene in time, whereas relational links are only intra-frame connections. Consequently, each video segment (equivalent for each action) is encoded in a big graph, where temporal links are the way to distinguish between frames.

The fundamental objective is to develop a model capable of accurately identifying and categorising diverse actions performed within the video sequences, leveraging the inherent spatial and temporal dependencies encoded within the graph representation. This task necessitates the exploration of Graph Convolutional Networks (GCN) architectures tailored to effectively capture and propagate information across the graph nodes and edges, thereby enabling robust action recognition performance in this complex and dynamic visual domain.

7.2.2 Proposed Methods

We decided to use some pre-defined layers in the PyTorch Geometric library for our network creation. Given the spatio-temporal nature of the graphs, the incorporation of edge attributes during model training emerges as pivotal, alongside the intention to use Convolution layers in our pipeline. After some preliminary search with trial and error followed by preliminary evaluation of some of them, we decided to use the GENConv (G. Li et al., 2020) Layer which

is quite flexible and formally can be presented as:

$$\mathbf{x}_i' = \mathrm{MLP}\left(\mathbf{x}_i + \mathrm{AGG}\left(\left\{\mathrm{ReLU}\left(\mathbf{x}_j + \mathbf{e_{ji}}\right) + \varepsilon : j \in \mathcal{N}(i)\right\}\right)\right) \tag{7.10}$$

where *AGG* is an aggregation function (e.g. softmax, mean etc), and $\mathcal{N}$ is the set of neighbouring nodes, ε is a small constant equal to $10^{-}7$, x_i' is the updated version of x_i when x_j are its neighbours.

The model's input comprises scene graphs, delineated as x, edge_index, and edge_attr, while the output corresponds to the activity class. The network is composed of several GenConv layers stacked and after that a Flattening layer and two fully connected layers, as in the scheme in Fig. 7.5. The number of GenConv layers used is equal to three, with respectively [input, 64, 128] channels. And then two dense layers of 128 and 64 channels each. The novelty of this preliminary work is the new usage of this dataset and this type of data to show how it is possible to enrich action recognition using environmental cues and information between the subject and its context. In the experimental section, we describe how we achieved it, giving more details about the implementation.

Chapter 8

Experiments: Dataset and results

This chapter encompasses all the experiments to validate the presented methodology and it has been structured as the Chapter 7 in two main parts, the first for the MOSAIC algorithm and the other devoted to the one based on the scene graph data.

8.1 MOSAIC: Minimal Action Classification

8.1.1 Preliminary observations

Before presenting the conclusive outcome attained through the proposed approach, we deem it pertinent to outline a preliminary method upon which our efforts were initially concentrated. This preliminary method eventually drew us to the final version described above. The data, used only in this section, come from a small dataset of 10 upper-body actions we acquired in our laboratory with a zed-camera. The full details about it are reported in Appendix B. As a preliminary experiment, we tried to input our data divided into primitives of motion for one or two hands(using only the velocity profiles, or modulus of the velocity) into an LSTM network composed of a few layers. The input data of our network resembles the one in Fig. 8.1, where a time series represents one action.

As a classifier, a Fully Connected head was attached to the LSTM. The classification has been done for 10 different classes. We decided to plot the representation after the LSTM network and before the classifier to see how the classes were represented Fig. 8.2. However, Fig. 8.2 represent the same embeddings but the colour code for the left and right-hand sides are different. On the left, we grouped the classes by action categories: blue are eating actions with the hand going from the table to the mouth repetitively, green are two-hand actions, red are touch actions, and pink represents transport actions. We can clearly see a

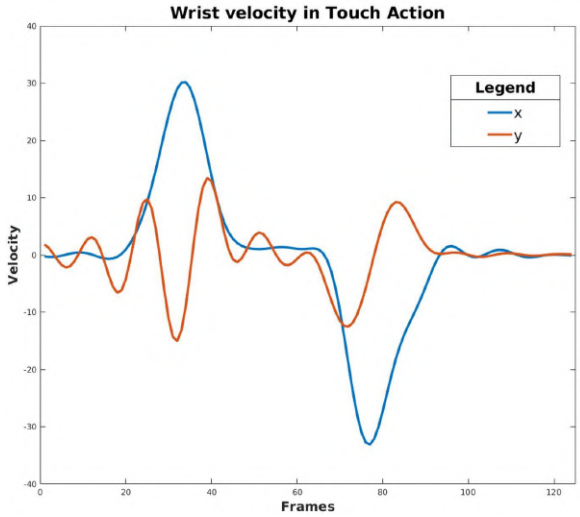


Fig. 8.1 Velocity for the right wrist while performing a Touch Action. We can see the x dimension clearly indicating a 'go' (positive bell) and then a 'come back' motion (negative bell).

clustering between them, which is in no way guided by a supervised classification because the labels were the ones reported in the legend (not the categories just mentioned) and this representation is taken before the classifier layers. The right-hand side of Fig. 8.2 shows the same embeddings but with a colour code indicating the classes. This experiment has been a preliminary step to assess the possibility of doing action classification only using kinematics primitives extracted from keypoints deriving from a pose estimator. The usage of only kinematics primitives is tough because it is a very bare representation, so it is interesting in itself to see how powerful they are. The fact that the representations clusters also per category of actions is a small proof that it is possible to appreciate differences between these categories thanks to kinematics, and probably in the number of primitives to do an action.

8.1.2 Datasets

The BABEL dataset (Punnakkal et al., 2021) is a large-scale dataset providing very precise 3D key points – acquired with motion capture data. Unlike other benchmarks, like (J. Liu et al., 2019), BABEL more closely resembles real-world conditions providing a very challenging test-bed for our approach.
It is derived from the AMASS dataset (Mahmood et al., 2019), a collection of different marker-based mocap datasets that have been represented using a common parametrization to convert the data into realistic 3D human meshes. The AMASS dataset was mainly designed for animation, visualization, and data generation purposes. In BABEL, the sequences have

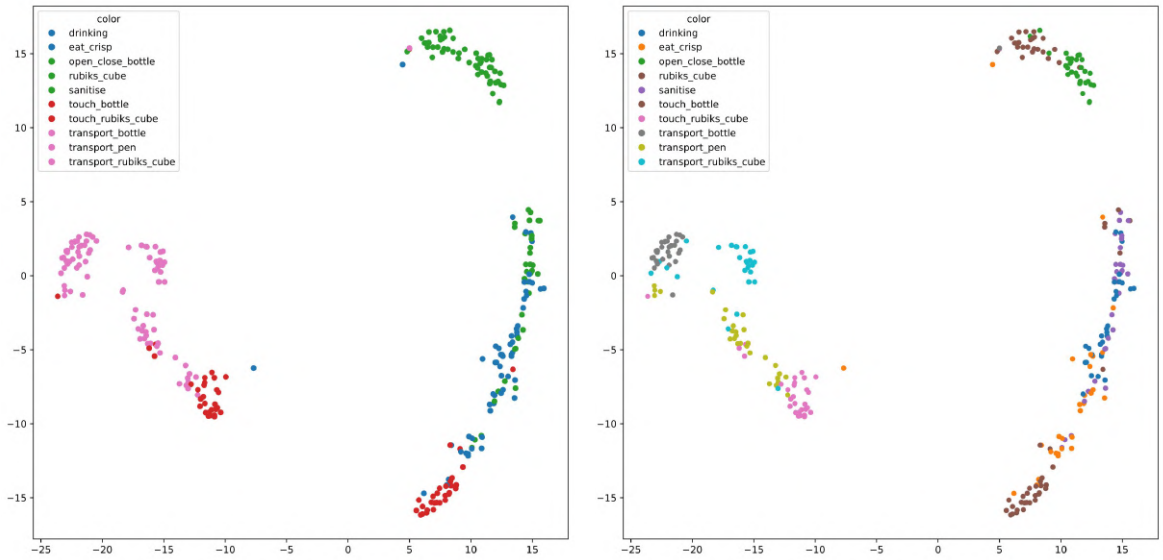


Fig. 8.2 Embeddings where each dot is the representation of an entire action taken from a bigger latent space to a space of dimensionality 2 thanks to a transformation performed with the t-distributed stochastic neighbour embedding (t-SNE) algorithm.

been annotated at a global level (an action label for each sequence) and at a frame level (an atomic part of the action that can be short in time or instantaneous).
However, AMASS is a motion capture dataset and BABEL a keypoint one; to pass from one representation to the other the enabling piece of engineering is SMLP(Loper et al., 2015). SMLP is a software able to render human meshes from a sparse mocap representation; it has the feature of building up a rendered human from two sets of data: human parameters (gender, height, and many others) and motion parameters. This particular mesh has the possibility to sample different sets of keypoints on the human body.
There exist two versions of the BABEL dataset for action recognition: 120-actions and 60-actions, following the NTU(J. Liu et al., 2019) dataset scheme. Both of them are imbalanced and have long-tailed distributions but have different numbers of classes. In the following, where not explicitly stated the 120-classes version is used.

For our task, we considered the sequence-level annotation and the two versions of the dataset, 60-actions and 120-actions. The actions included in the dataset show high variability and complexity, in terms of the granularity of the actions but also of the number of samples. A data split is available and concerning the 60-actions dataset it encompasses 45473, 17067 and 15647 sequences for training, validation and testing respectively; while for the larger dataset, the split is 48978, 18363, and 16839, all these numbers are resumed in Table 8.1. Notice that while doubling the number of classes the total number of samples remains stable, showing the gap in complexity between the two versions of the dataset. The splits are pre-defined in

the dataset itself and we suppose they are built using a cross-person procedure.
The action classes are divided into 8 semantic categories of uneven complexity: simple dynamic actions, static actions, object interaction, body part interactions, body part, type of movement, activity, and abstract actions.
Despite its precision, BABEL presents different challenges. First, the classes are highly imbalanced, with actions having just 19 samples, while for others there are 6260 samples available (some details are reported in Table 8.2, see also Fig. 8.3 and Fig. 8.4). The complexity of the actions and their variability is very wide. For all these reasons, we consider the dataset an ideal test-bed to assess MOSAIC.

Table 8.1 Babel dataset number of samples per split.

dataset version	training set	validation set	test set
60-actions	45'473	17'067	15'647
120-actions	48'978	18'363	16'839

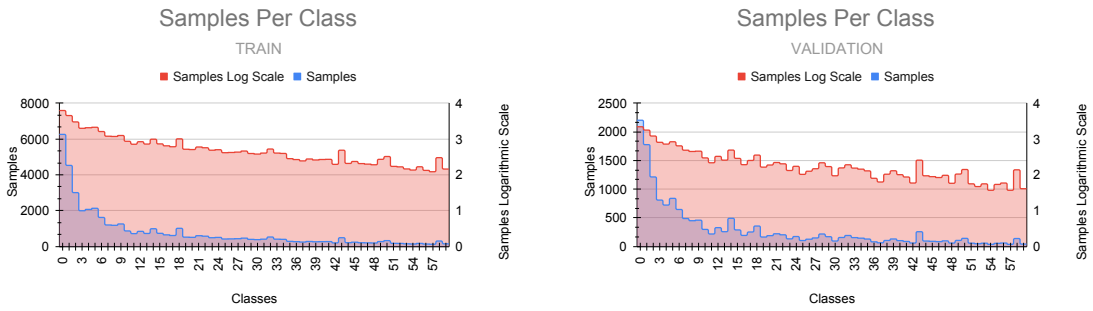
Table 8.2 From left, minimum, maximum, average and median number of samples per class in the BABEL dataset. The dataset is very challenging due to the presence of classes highly under-represented.

BABEL	split	Min.	Max.	Avg.	Med.
120 actions	train	19	6260	408	133
120 actions	val	1	2203	153	51
60 actions	train	122	6260	758	418
60 actions	val	37	2203	284	151

The data in BABEL are already pre-processed with the standard NTU procedure: padding the empty frames (if any) with the previous frame, parametrise the data to express all the skeletons with respect to the spine joint (joint 1), parallel the hip (joint 0) and spine joint with the z-axis and then parallel the bone between the right and left shoulder (joints 8 and 4 respectively) to the x-axis.

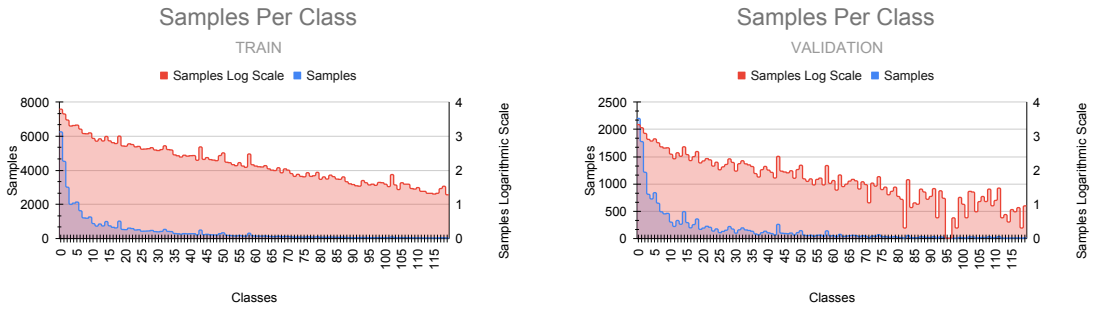
8.1.3 Experiments

The majority of experiments presented are conducted on the BABEL dataset because it is big and diversified and there are few chances to overfit (or customise) a model onto the data.



(a) Training (b) Validation

Fig. 8.3 Class Distribution on the training set and on the validation set of the 60-classes Babel dataset



(a) Training (b) Validation

Fig. 8.4 Class Distribution on the training set and on the validation set of the 120-classes Babel dataset

Indeed, the strong unbalance of the classes and the large number of them, make it a tough test bed to test an architecture.

The metrics used to assess the Action Classification task are taken from the literature, especially from (Punnakkal et al., 2021), and are common choices for the action classification domain.

- Top1 accuracy: the action class predicted is in accordance with the label. This metric states how many correct classifications the model does. It is the conventional accuracy.
- Top5 accuracy: the metric takes as correct the samples where the label is in the top 5 predictions of the model. So, it takes into consideration noise in the annotation labels or -in the case of the Babel dataset- the fact that sometimes more than one label can be present inside a sample.

- Top1 norm: it is the mean top1 accuracy across categories. It is the mean of the per-class accuracy, over all the classes.

In this respect, the difference between the top1 and top1 norm accuracy can be seen as a measure of unbalance in the model predictions (in this case due to the nature of the data used). In fact, top1 expresses the accuracy divided by the total number of samples, respectively of the classes, whereas the top1 norm takes the class accuracy and then averages them.

Furthermore, let us write down a reference configuration, where the more influential parameters are reported Table 8.3. To assess the benefit of them in the pipeline, all of them have incurred in the process of experimental validation through many trials.

Table 8.3 Standard configuration of the MOSAIC architecture.

classes	latent_dims	lr	label smooth	loss	scheduler	classifier	feedforward	heads	encoders	dropout
120	128	0.002	0.25	balanced cross entropy	cosine	64	256	16	3	0.01

The standard configuration Table 8.3 presents -from left to right-: the number of classes of the dataset; the dimensionality of the representation (latent space in VAE/feature dimension of each token); the learning rate; the percentage of label smoothing in the loss calculation; the loss chosen; the scheduler controlling the decay of the learning rate of the optimiser; the number of neurons in the fully connected head performing classification; the dimensionality of the feedforward of the transformer; the number of transformer's heads of attention and the number of encoder layers; and finally the dropout percentage in the transformer and in the classifier. Regarding the data used for training and testing, we would acknowledge the following. For the training procedure of the autoencoder we used the *babel-training* set split as training and validation data and we tested on the *babel-validation* set (in order not to corrupt the downstream task evaluation). For the Transformer and the Classifier, we used the *babel-training* as the train set, and the *babel-validation* as the validation set. The test set labels are private, so the results in Table 8.14 have been evaluated on the evaluation server provided by the Babel dataset creators. All the other tables are fed with results calculated on the *babel-validation* set.

The Fig. 8.5 shows the different schedulers used in the experimental pipeline.

There are many other hyperparameters chosen in the architecture (for more details see Appendix C, which can be seen in the code available on GitHub, or if important, they will be mentioned in the following.

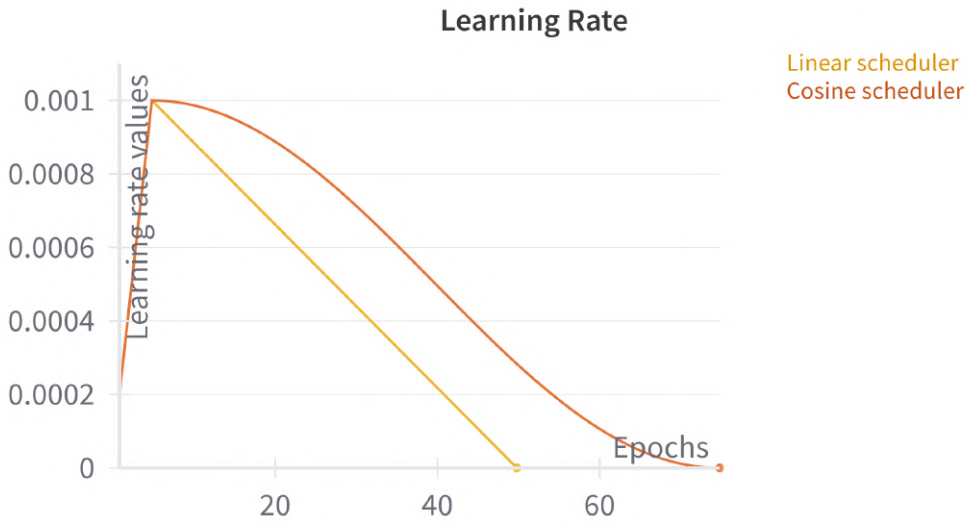


Fig. 8.5 Two different schedulers used: Linear, Cosine. They decay the learning rates as function of the epochs elapsed as illustrated in the figures. In this graph, the Linear is instantiated for a training of 50 epochs, whereas the Cosine for 75 epochs(only for drawing purposes).

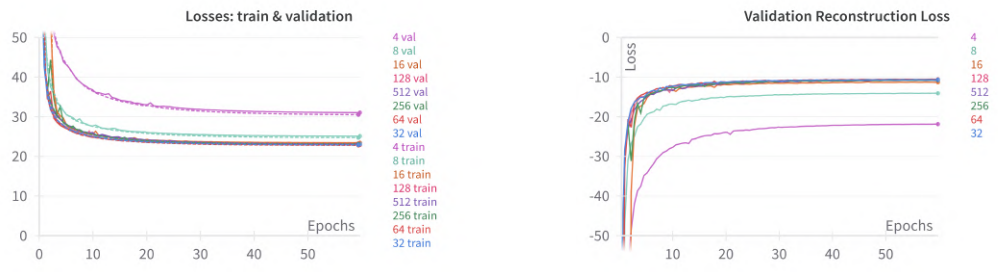
8.1.3.1 Experiments on the VAE

VAE losses

Before choosing the latent dimensionality on the basis of the downstream task, i.e. action recognition, we made sure to have our VAE in the correct operation setting. We picked several latent representation dimensions and we assessed the reconstructions and the losses. As we can state from Fig. 8.6, all the VAEs with a latent dimensionality bigger than 16 achieve the same amount of loss: capacity of reconstruction and KL-divergence. The total loss used for the training is the sum (and not the mean) per batch of all the errors in reconstruction, which is not very informative but -numerically- high enough to balance the contribution of the KL loss without using any adjustment weights (as used sometimes in literature (Asperti and Trentin, 2020)).

The losses are also measured in a mean-per-clip fashion to understand how capable each network was of reconstructing every single word (or clip). However, rather than the sum loss, the per-clip-mean is a real measure and a way to assess if a word's length is better than others. But, here, the main message we wanted to convey is that in each experiment the VAE was trained till full convergence and till a satisfactory reconstruction error.

One idea we found to work quite well and improves our results in reconstruction is the usage of a weight between the reconstruction loss and the KL-divergence loss. The Beta β



(a) Training & Validation Losses (b) Reconstruction Loss

Fig. 8.6 Losses for different dimensions of the latent dimensionality.

parameter we are going to introduce is the weight multiplied by the KL-loss. We decided to start training the architecture with one epoch where the two losses weigh the same, then varying β as observed in the Fig. 8.7. The β definition is as follows for a 120 epochs training and considering x the epoch steps:

$$\beta = \begin{cases} 0 & x = 0 \\ \frac{1}{25}x - \frac{1}{25} & 1 \leq x \leq 26 \\ \frac{1}{25}x - \frac{27}{25} & 27 \leq x \leq 30 \\ 1 & 31 \leq x \leq 120 \end{cases} \tag{8.1}$$

Pooling vs no Pooling

The Autoencoder dimensionality has been proved using different latent space and kernel sizes, keeping in mind that a small and fixed kernel is often the choice. The motivation is that the small kernel can 'see' the small variations in the data series, and with the increase of the depth of the model it can appreciate also big patterns (due to the convolution mode of operation). However, to validate also this type of choice we tried to use pooling layers across the intra-word time dimensionality. So shrinking the word length layer by layer, to achieve a more complete representation in the latent space. However, not every word length fits inside the pooled version, only words longer than a threshold which allows them not to collapse in a zero-dimensional vector. The results between using pooling and not using it are very similar, so we decided to use the simplest and more explainable approach. Another variation, which may be found in the literature in a very similar setting can be the pooling of the feature dimensions i.e. the number of keypoints represented. Indeed, in the (Cardoso et al., 2022) they start with a skeleton dimensionality of 25 nodes and then they end having only one node as representation. However, we did not follow this path, leaving it still open for explorations.

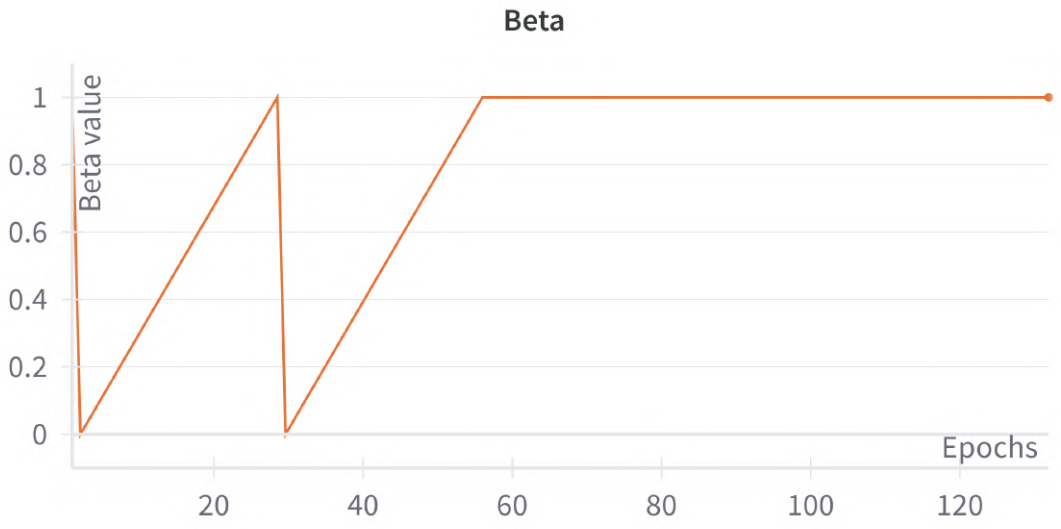


Fig. 8.7 Beta value of the VAE for a configuration of 120 epochs with two cycles of 25 epochs each. (It is the parameter controlling the learning of the reconstruction and the statistical distribution of the embeddings.)

Table 8.4 Classification metrics with different input data for Babel with 60 Classes.

model	dataset version	VAE	VAE train	top1 (%)	top1_norm (%)	top5 (%)
ours	V P	VAE	120	40.31	25.47	70.30
ours	V P	VAE POOL	120	-	-	-
ours	V P	VAE	60	38.79	23.04	68.52
ours	V P	VAE POOL	60	39.06	21.77	68.73

8.1.3.2 Experiment on the Word length

We start with an assessment of our method under different conditions, one of which is the word length or clip length dimensionality. What we performed is a change in the value of τ, i.e. the length of the time window used to split the sequence in clips (see Section 7.1.3). The results are shown in Fig. 8.8, the input for this experiment is a sequence of joints velocities (the most challenging input). Here we may observe that some values perform significantly better than others, and it can be appreciated as a trend. However, it is not obvious to conclude that similar scores mean similar output, indeed the top-1norm does not decrease as much as the others for $\tau = 75$. This may lead to the conclusion that different window size may contribute to the classification of different classes. Furthermore, we may hypothesize this is because there is not a single appropriate time window for all the action classes. We tried to investigate how differences are present in the per-class accuracy, as we reported in Fig. 8.9b, Fig. 8.9c and Fig. 8.9d. In this regard, we can say that thanks to the balanced loss, also

Table 8.5 Classification metrics with different input data for Babel with 120 Classes.

model	dataset version	VAE	VAE train set	top1 (%)	top1_norm (%)	top5 (%)
ours	V P	VAE	120	37.43	20.20	65.88
ours	V P	VAE POOL	120	37.15	19.78	65.50

on poorly represented classes the model exhibits an acceptable amount on learning. Also, it is worth taking into account the fact that in the validation dataset, all the classes after the 60th, have less than 50 samples each, and after the 85th they have an average of 20 or fewer samples per class. The different time scales used grasp different types of actions, for example, one of the three peaks in Fig. 8.9b is the action label 'stumble' which is an instantaneous action. In this regard, probably, having more time windows (so shorter ones) does not help, because they are all 'empty', or probably misinterpreted as walking, and the only one in which someone stumbles, has not enough weight to make the model converge for that decision.

Table 8.6 Classification metrics for different clip lengths for Babel 120.

dataset	clip length	top5(%)	top1(%)	top1 norm(%)
120	149	61.93	34.80	15.14
	75	65.48	37.01	19.96
	25	67.92	38.61	19.11
	10	68.28	39.01	19.83
	5	45.79	19.75	5.07

8.1.3.3 Experiment on the Transformer's hyperparameters

Token size

In Table 8.7 we analyse the effect of choosing different token sizes (we remind that the token is essentially the word in the action sentence, hence its length is the size of the VAE latent space). Changing the size of the latent space has only a minor effect on the performance, slightly more significant in the case of the top-1 norm. This suggests that the choice of the size of the latent space is not extremely critical in our method.

Losses

Furthermore, we investigate the use of two different loss functions, as done also in (Punnakkal et al., 2021). The first one is the standard cross-entropy loss, possibly weighting the contributions of different classes to deal with data imbalance. The second one is the focal loss (Lin et al., 2017b), an extension of the previous one that helps the model to pay particular attention to hard misclassified examples which are usually the poorly represented classes.

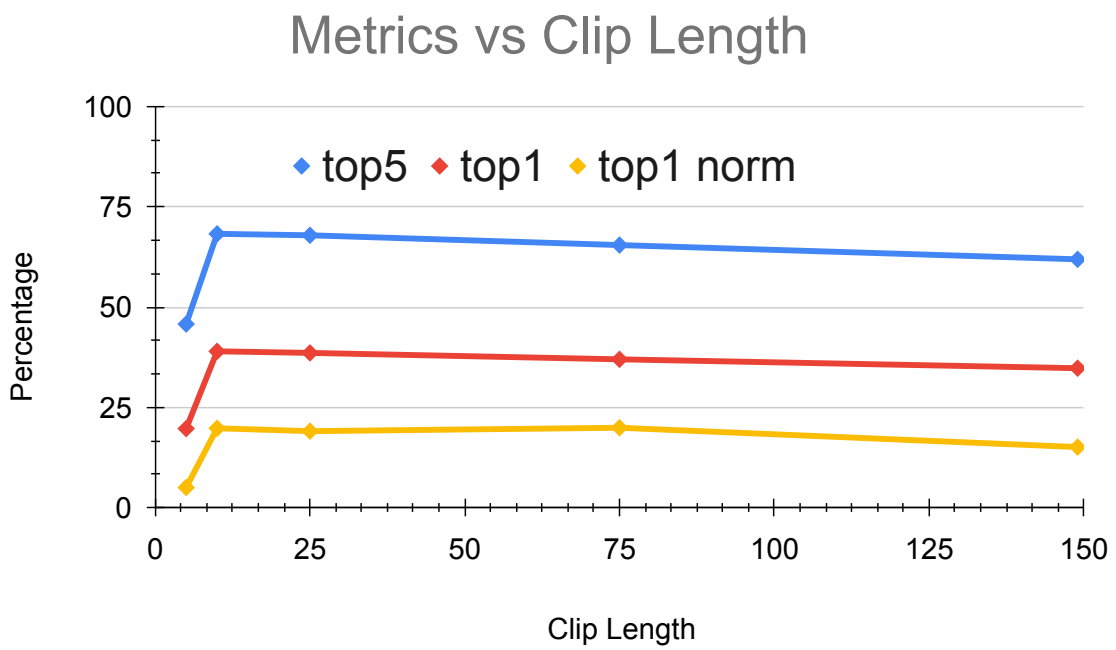


Fig. 8.8 Classification metrics with respect W (the clip length chosen) on the x axis.

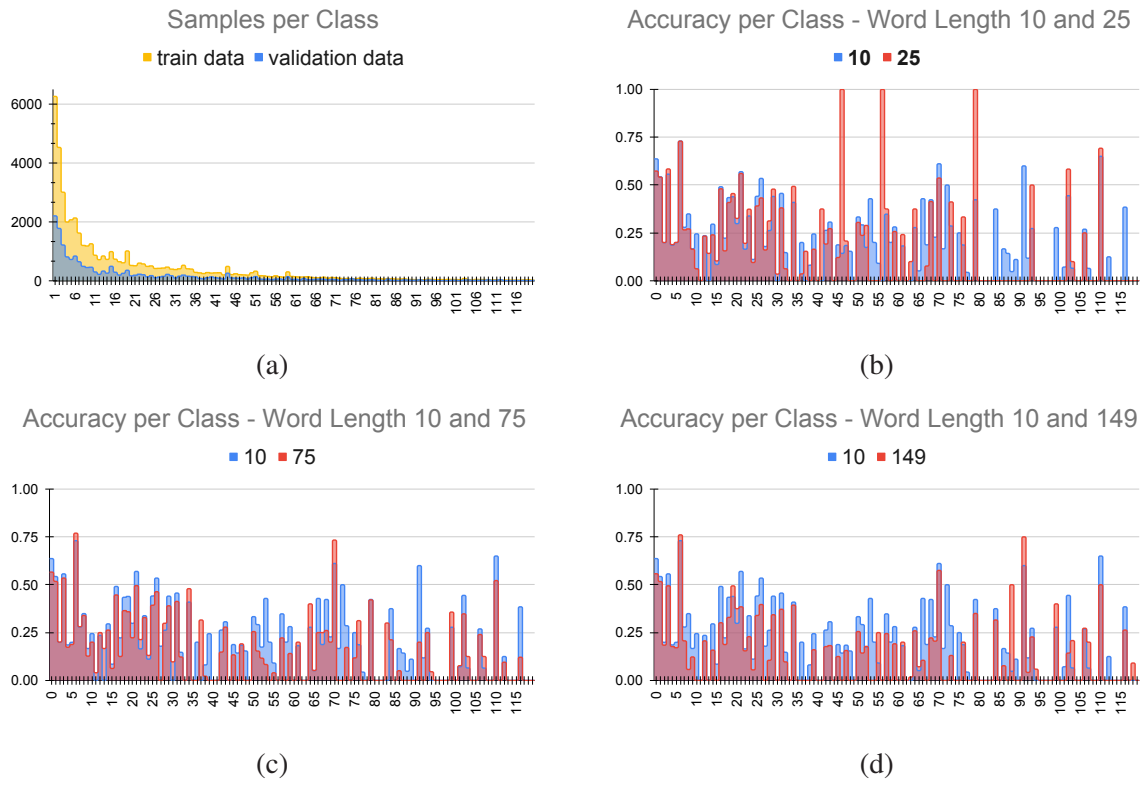
Table 8.7 Classification metrics for different token dimensionality (VAE latent space dimensionality) on Babel 120.

token dim.	encoder layers	heads	feedforward	top5(%)	top1(%)	top1 norm(%)
256	3	16	256	58.60	29.74	10.53
128	3	16	256	59.53	30.52	11.75
64	3	16	256	59.82	30.76	12.81
32	3	16	256	59.95	30.78	11.71

The results are reported in Table 8.8, and it is evident how the balanced loss is well-suited for the problem, and the focal loss alone does not achieve comparable results to its balanced version.

Fixing all the other hyperparameters and configurations, as the representation and using always the same VAE for the representation The Table 8.9 reports preliminary results on the effects of some hyperparameters of the Transformer architecture. As we can appreciate, almost any configuration works the same, suggesting that the representation is rich enough for them. On the other hand, the table suggests that to improve the overall performance some changes on the classifier head, or on the data and on the VAE encoder are probably needed.

Following the considerations above we continued testing, starting from the representation power of the VAE. In this regard, the first hyperparameter to be tuned on the VAE is the



(a) (b) (c) (d)

Fig. 8.9 Fig. 8.9a Sample distribution per class in train and validation for the Babel-120. Fig. 8.9b Top1 vs Classes for τ=10 and τ=25.Fig. 8.9c Top1 vs Classes for τ=10 and τ=75.Fig. 8.9d Top1 vs Classes for τ=10 and τ=149.

latent space dimensionality, which directly translates into the token feature dimensionality in the Transformer.

In parallel we performed also some other experiments on learning rates for the Transformer architecture, to find the best one; which turned out to be about 0.001 and 0.002.

However, cosine decay can be tricky if you do not arrive at the end of the fixed epochs because it is indeed more similar to a linear decay if it does not reach zero (or very small values). As a pair with Table 8.11 there is Fig. 8.10 where the different losses expose a different learning speed across the steps. As we can see in Fig. 8.10 the architecture is quite robust on a certain range of learning rates. To be highlighted is the overfitting of all of them apart from the smallest one, which at the end it is still in the learning phase.

8.1.3.4 Experiments on the Multi-Resolution

Driven by the curiosity of investigating the exploitation of different resolutions and a possible gain in performance thanks to the information gained from different data granularities at the

Table 8.8 Classification metrics for different loss functions on Babel 120.

dataset	loss	top5(%)	top1(%)	top1 norm(%)
120	balanced-CE	68.46	39.19	22.97
	CE	31.24	31.76	11.72
	balanced-focal	68.84	38.76	17.03
	focal	66.64	38.34	11.27

Table 8.9 Classification metrics for different hyperparameters of the Transformer, Token dimensionality 128, on Babel 120.

encoder layers	heads	dim. feedforward	batch first	top5 (%)	top1 (%)	top1 norm (%)
4	16	256	✓	59.52	30.72	12.55
2	16	256	✓	59.43	30.90	12.30
1	16	256	✓	59.40	30.60	11.57
3	32	256	✓	59.76	30.92	12.08
3	16	256	✓	59.34	30.49	10.43
3	8	256	✓	59.62	30.78	11.92
3	4	256	✓	59.14	30.72	12.18
3	2	256	✓	59.29	31.23	12.73
3	16	1024	✓	59.64	30.62	11.76
3	16	512	✓	59.70	30.76	12.18
3	16	256	✓	60.07	31.07	12.50
3	16	128	✓	59.53	30.52	11.75

same time we tried the following approach. We use the same VAE+Transformer pipeline but using a couple of VAE and Transformers in parallel and perform a late fusion approach on the Transformers outputs. The classifier used here has the same characteristics as the previous one but with a bigger hidden layer of 256 neurons. With the experiment in Table 8.12 we test the use of a temporal multi-scale approach with late-fusion, which does not influence the accuracies (with respect to the single-scale case) but for the top-1 norm, that improves when employing the 25+10 combination of time windows. We might observe that short-time windows perform, on average, better than longer ones, probably because of the richer sequence provided to the Transformer when the clips are shorter in time (hence in a higher amount), but also because subtle differences are easier to be encoded in the latent representation.

8.1.3.5 Experiments On the Kinematics Features

As a final experiment for this section, we consider the one in Table 8.13 where we compare different combinations of kinematic features in input, as explained in Section 7.1.6: the speed $\mathbf{V} = [V_x, V_y, V_z]$, its magnitude V, the position $\mathbf{P} = [X, Y, Z]$, and the combination position+magnitude of velocity, i.e. **P**+V. We observe -in Table 8.13- that the latter combination is the best-performing one. We may hypothesize that while the positions are already very informative, the use of additional dynamic features helps to disambiguate challenging action

Table 8.10 Classification metrics for different token dimensionality (VAE latent space dimensionality) on Babel 120.

token feature dim.	encoder_layers	heads	dim. feedforward	top5 (%)	top1 (%)	top1 norm (%)
256	3	16	256	58.60	29.74	10.53
128	3	16	256	59.53	30.52	11.75
64	3	16	256	59.82	30.76	12.81
32	3	16	256	59.95	30.78	11.71

Table 8.11 Classification metrics for different learning rates, fixing the scheduler as a cosine decay, for Babel 120.

learning rate	latent dims	loss name	scheduler	top1 (%)	top1_norm (%)	top5 (%)
0.003	128	balanced-ce	cosine	60.17	30.68	10.86
0.002	128	balanced-ce	cosine	60.71	31.72	14.15
0.001	128	balanced-ce	cosine	60.79	32.22	13.77
0.0005	128	balanced-ce	cosine	60.41	31.55	13.25
0.0001	128	balanced-ce	cosine	57.39	29.01	6.78

samples. This is the configuration of our method we consider in the comparisons with existing approaches in the next session.

As suggested by results in Table 8.9 another possible way of achieving better results was to work on the upstream data representation part. In Fig. 8.11 we report results fixing VAE and Transformer and changing only the data input shapes and preprocessing as explained in Section 7.1.6

In accordance with the Fig. 8.11 we also show the Table 8.13 with the accuracy metrics reported for these experiments. As we can see from Fig. 8.11 which represents the top-1 accuracy during training, one run of the experiment did not even finish a proper training (stopped by the early-stopping criterion) because the information encoded in the tokens where probably not enough to learn something. The others, in different ways, learn to discriminate between actions. One fundamental observation is that the amount of input data is different for the different runs; e.g. one time instant in the yellow Modulus of Velocity is encoded with a feature depth of 25, instead, the blue one has 100 features per time instant. However, the most interesting observation is that the yellow curve in Fig. 8.11 (the modulus of velocity alone) has better performance with respect to the red curve, which encodes the modulus plus the speeds on the three-axis. Instead, adding to the modulus the positions of the keypoints, helps the classification task, because it is a way of giving spatial information and proximity information between the joints. In these experiments, where we observe the learning curves of the classifier, we used one VAE architecture, of course training it with different data each time, ending up with as many trained VAE models as classifiers trained.

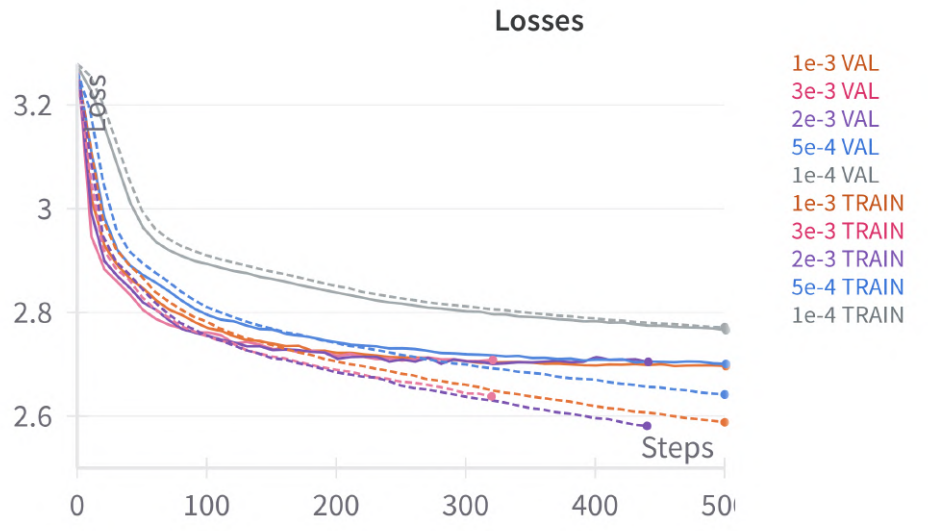


Fig. 8.10 Different Training-Validation losses with different learning rates indicated in the legend, on Babel 120. (1 epoch is approximately 10 steps on the plots.

Table 8.12 Classification metrics with multi-resolution model on Babel 120, also one result evaluated on the test server is reported.

dataset	W1	W2	top5(%)	top1(%)	top1 norm(%)
120	149	10	62.52	35.61	15.52
	75	10	64.24	37.72	18.02
	25	10	67.29	38.58	21.52
test	25	10	66.39*	37.08*	18.48*

8.1.4 Comparison with the State of the Art

We now discuss our results in comparison with the state-of-the-art of the dataset. We report this evaluation in Table 8.14, our reported results are evaluated on the private test set (on the BABEL page server). The performance of our method is only slightly below the best in the literature. In this respect, it is worth noticing that different methods in the table (e.g. 2s-AGCN or MST-GCN) are based on two-stream approaches. In this sense, we highlight the potential of the modular solution we are investigating. Indeed with a simple, "linear" structure and only relying on kinematic features the classification performance is very close to top-performing methods. Those methods take advantage of the usage of connectivity between keypoints (or bones), so we can observe that this piece of information is beneficial for the task. Our architecture, having made the necessary distinctions, uses only one of the two data streams of information of the 2sAGCN which is the leading architecture for the

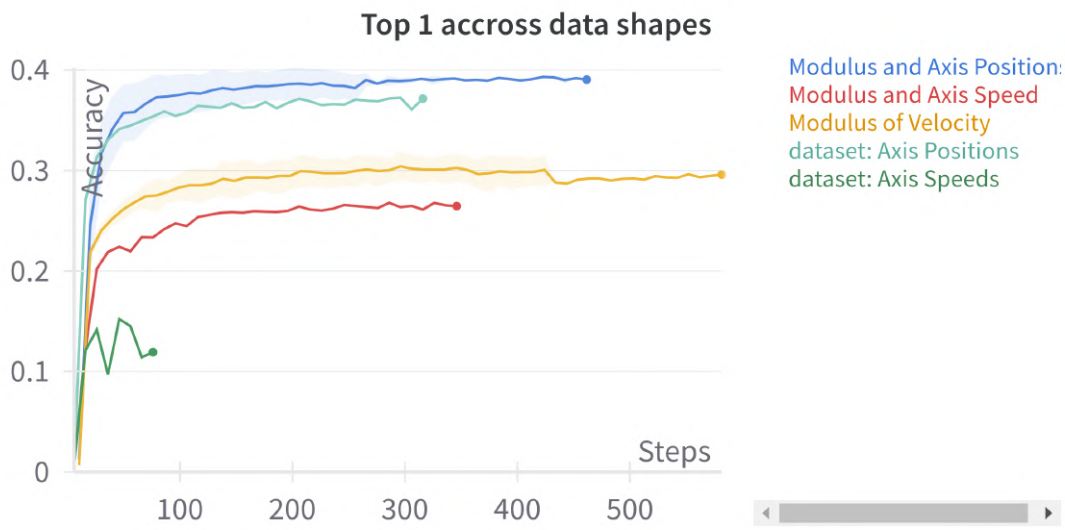


Fig. 8.11 Top 1 Validation Accuracy for all the different data shapes tested. The curves are the mean across many different configuration experiments (where present) and the shadows are the standard deviation among them.

Table 8.13 Classification metrics for different data shapes and W=25. (**V**=3D velocity, V=magnitude of speed, **P**=3D position)

dataset	data shape	top5(%)	top1(%)	top1 norm(%)
120	**V**	37.18	11.92	2.69
	V	59.21	29.74	10.53
	P	65.65	37.15	**19.78**
	P+V	**67.92**	**38.61**	19.11

task. Indeed, to make a comparison only the stream of the node features is used, instead, the connections (or the bones) are not exploited in our method. What we can state is that this type of missing information can be relevant to the task because it makes it easier for the network to understand proximity and causalities between keypoints, due to their connections; however, our goal was multifaceted and was more centred towards the representation than on the final output classification results. However, we consider exploring the role of connectivity between joints in future works.

Looking at the numerical results, and in particular at the difference between our Top-1 and Top-5 we can state that our method is almost as good as the top-performing one for the Top-1 metric, but on the Top-5 it stays behind. This suggests that it is easy enough (in line with others) to get the first class correct, but it is less easy to disambiguate between the others. This is probably due to two factors. The former is our representation where the absence of bone connectivity can lead to ambiguities, but it is still to be proved. The latter is the fact

that there are some actions in the dataset that are ambiguous, or very similar to one another; this lead the model not to disambiguate between them correctly. Another point is the fact that all the architecture underperforms in the Top-1-norm with respect to the Top-1 and this highlights the lack of capability of learning from few samples (remember that the dataset is strongly imbalanced). We should also point the reader to Fig. 8.9b where the per-class accuracy is reported in detail, and it is clear that very few classes reach 50% accuracy, even the more represented one. But it is also very evident that classes above the 80th are not learned at all apart from a few exceptions.

Since we could not test (on the private test set through the BABEL page server) every setup due to the lack of test labels, we examined some of them to highlight their similarities, see Table 8.15. This helps us to show that the results from the validation dataset closely match those from the test set (there is always a difference of 1%-2%), validating the fact of showing the other results only on the validation dataset. Another interesting comment which arises from Table 8.15 is the difference in performances for what it concerns the Top-1-norm metric from validation and test which is unexpectedly high for the 60-classes dataset and lower for the other. We would have expected the opposite due to the huge imbalance in the 120-class set.

Table 8.14 Results for BABEL60 and BABEL120 on the test set (evaluated through the server). With MOSAIC we refer to our action classification method using the embeddings obtained from the VAE trained on BABEL60, while MOSAIC* refers to the action classification model employing the embeddings from the VAE trained on BABEL120.

Actions	Method	Top-5 (%)	Top-1 (%)	Top-1-norm (%)
60	2s-AGCN (from (Punnakkal et al., 2021))	73.18	41.14	24.46
	ST-GCN (from (Cardoso et al., 2022))	44.20	24.20	14.40
	2s-AGCN (from (Cardoso et al., 2022))	67.80	33.80	30.40
	MST-GCN (from (Cardoso et al., 2022))	70.30	36.30	35.40
	Cartdoso et al. (Cardoso et al., 2022)	70.40	36.40	30.30
	2s DG-STGCN)(H. Duan et al., 2022)	-	40.00	-
	MOSAIC (our)	67.50	37.31	20.24
	MOSAIC* (our)	69.77	39.27	20.95
120	2s-AGCN (from (Punnakkal et al., 2021))	70.49	38.41	17.56
	ST-GCN (from (Cardoso et al., 2022))	28.60	20.50	5.50
	2s-AGCN (from (Cardoso et al., 2022))	58.00	27.90	26.60
	MST-GCN (from (Cardoso et al., 2022))	60.10	29.90	29.80
	Cartdoso et al. (Cardoso et al., 2022)	65.40	31.40	28.40
	2s DG-STGCN (H. Duan et al., 2022)	-	32.80	-
	MOSAIC* (our)	66.26	37.29	18.80
	MOSAIC* multi-resolution	66.39	37.08	18.48

Table 8.15 Results for BABEL 60 & 120. Our results comparison between validation and test accuracies.

Actions	Method	Top-5(%)	(val)	Top-1(%)	(val)	Top-1-norm(%)	(val)
60	MOSAIC	67.50	(69.54)	37.31	(38.79)	20.24	(23.04)
	MOSAIC∗	69.77	(71.11)	39.27	(40.11)	20.95	(24.96)
120	MOSAIC∗	66.26	(67.82)	37.29	(38.82)	18.80	(19.07)

8.1.5 Discussion

All the previous experiments assessed the best parameter to achieve the result in Table 8.14. The best data to achieve this result is the concatenation of the 3D position and the magnitude of the velocity in the 3D space. This suggests that normalising the skeleton structure and its reference (forcing the hip to be in the centre and aligned with an axis) makes the positions informative. Indeed, the magnitude of the velocity of each keypoint helps the disambiguation of similar motion patterns which can be similar but performed with a different velocity. For example, walking and running, or walking and backwards-moving are probably done with different velocity magnitudes.

Another interesting finding which seems counterintuitive is that using multi-resolution approaches with this architecture does not pay. But what pays back is the usage of some training strategies as explained before, such as using a handcrafted learning rate for the VAE, in order to force it to work harder on the reconstruction side, and after it on the correct gaussianity of the retrieved representation.

One important point that we are still missing is the fact of representing the embeddings and showing their power. The motivation is that we are embedding part of movements that we want to be similar across different actions, so we want them to be superimposed: because we want to retrieve common primitives across different actions. However, we lack annotation for this fine-grained information so the visualisation could only follow a color-coding associated with their respective classes, which is not informative for our purpose. What we are trying to say is that two different actions can be made of the same primitive A (which can be 'rise arm') and a second which is B ('move right leg') but these two primitives can be present at a different time during an action and can be common of many of them, and not having annotation means that we cannot assign primitives to name (apart from doing it manually). So the plot of the primitive sequence in the embedding space does not make a lot of sense, because we are not able to name what we are seeing.

The key contribution of this part is a new deep-learning architecture that starts its reasoning from kinematics primitives of motion and tries to encode them in an unsupervised way. This is one of the first times in the computer vision literature where deep-learning is used for

action classification via kinematics motion primitives, only (Saunders et al., 2021) introduced a similar concept for a different problem. Another yet understudied contribution is the usage of a Transformer to encode non-cyclical time series (human motion as a set of keypoint velocities), which we could not find much reference on this topic. Indeed, the time series usually studied are cyclical with some periodicity such as temperature over the days or seasons.

8.2 ACROSS: Activity Classification via Human-Context Information

8.2.1 Dataset

The dataset used in this work is a subset of the Home Action Genome dataset (Homage) (Rai et al., 2021). The original dataset has 30 hours of videos, 70 classes of daily activities and 453 atomic actions. There are 86 object classes (excluding "person") and 29 relationship classes in the dataset. The Homage dataset has an ego-view and 3rd-view for each video. Each video represents one activity and each activity is made of multiple atomic actions. Ground truth scene graph annotation files are present for all videos representing objects, persons and relationships between them in each frame. A representation of the structure of the Home Action Genome is represented in Fig. 8.12

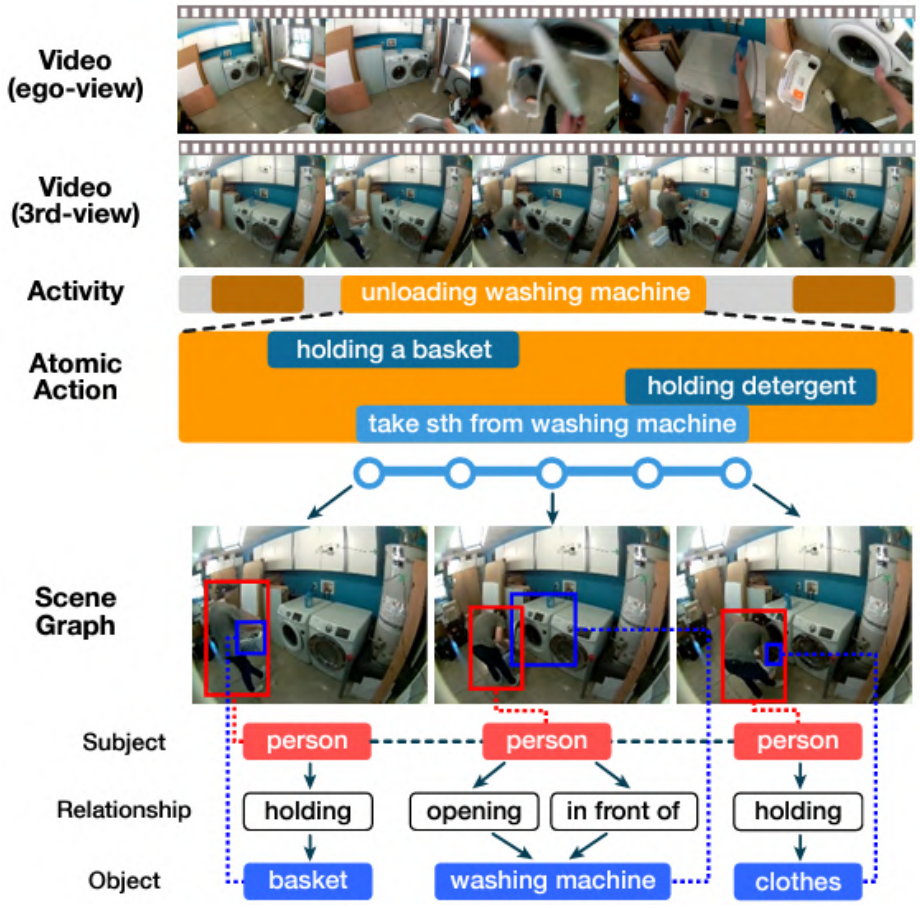


Fig. 8.12 Home Action Genome Dataset image [source (Rai et al., 2021)]

The HOMAGE dataset encompasses three distinct categories of spatial relationships: those pertaining to the subject attention, relative positioning between subjects and objects, and contact between subjects and objects. The delineation of these diverse relationship types is provided in Table 8.16.

Table 8.16 Relationship types in HOMAGE dataset.

Subject Attention	Relative Positioning	Contact	
looking at	in front of	carrying	covered by
not looking at	behind	drinking from	eating
unsure	on the side of	have it on the back	holding
	above	leaning on	lying on
	beneath	not contacting	sitting on
	in	standing on	touching
		twisting	wearing
		wiping	writing on

The subset of the dataset utilised in this study comprises 2617 videos depicting 69 distinct daily activities. These videos encompass 85 object classes, excluding the class "person," thereby totalling 86 when including "person" as a class. Furthermore, the dataset includes 25 relationship classes. Each frame within the videos features a singular subject (person) and one or multiple objects; furthermore, each frame adds the relationships established between the subject and the objects. Multiple relationships may exist between the subject and a single object, as well as between the subject and multiple objects within a frame. Notably, each video portrays a complete activity sequence from inception to conclusion.

The distribution of these videos across the 69 activities is illustrated in Fig. 8.13. The label distribution exhibits a satisfactory balance among the activity classes, with the majority containing a count ranging from 25 to 45 instances.

8.2.2 Experiments

8.2.2.1 Activity Recognition

Let us start to show a preliminary evaluation of the GCN layers by trying the 5 different types of layers as the Convolutional operator in our network. The general network layout is the one shown in Fig. 7.5, where we have four convolutional layers with ReLU activation then global mean pooling and dropout, then two dense layers and finally a softmax layer.

The Conv layers that we are evaluating are the following: TransformerConv, GENConv, GeneralConv, GATConv, GATv2Conv. The forward method of these layers takes the x, edge_index and edge_attribute.

Using the test network of Fig. 7.5 with the different Conv layers and with dropout = 0.2, number of epochs = 200, learning rate = 0.008, batch size = 200 and the Softmax as aggregation are shown in Table 8.17.

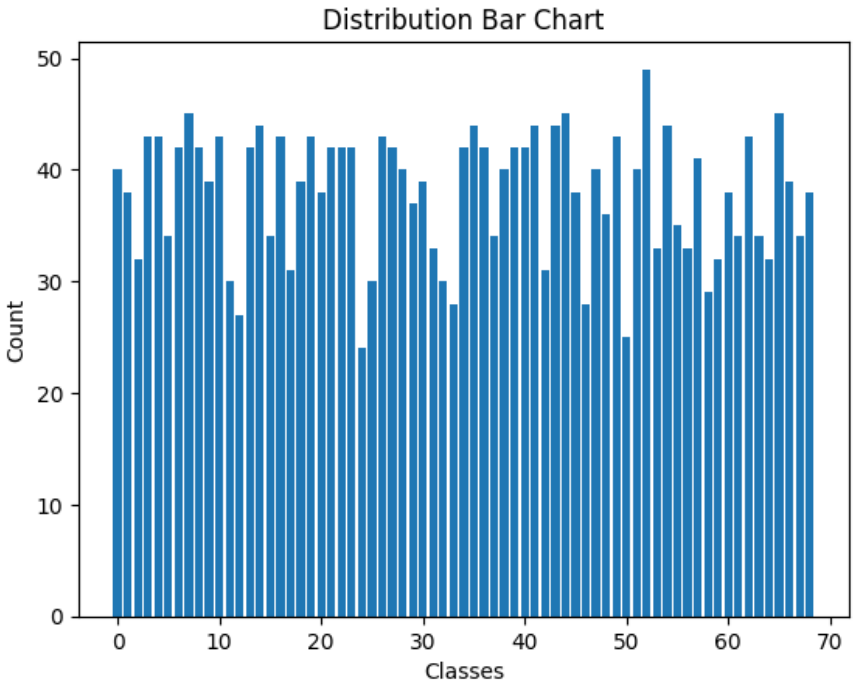


Fig. 8.13 Data Distribution between classes in the Homage dataset.

Table 8.17 Results of using the test network on the different Conv Layers with dropout = 0.2, number of epochs = 200, learning rate = 0.008 and batch size = 200

Conv Layer	Test Accuracy
GENConv	0.67
GATConv	0.44
GeneralConv	0.42
TransformerConv	0.29
GATv2Conv	0.18

Thanks to other experiments aimed at finding the correct network size and fixing all hyperparameters, we can say that the best model is composed of 3 Convolutional (GENConv) Layers and 2 Fully Connected layers as classifier of the flattened feature vector.

For the model optimisation task, and finding optimal hyperparameters we fix the number of epochs to a reasonable number equal to 250, the dropout to 0.2 and the batch size to 200. We also fixed the network to GENConv(5,64), GENConv(64,64), GENConv(64,64), GENConv(64,64), dense(64,64), dense(64,69). The Learning rate follows Table 8.18 The results of the experiments for finding the optimal model are shown in Table 8.18

Table 8.18 Train and Test Accuracy on different models and configurations, to find the optimal hyperparameters.

Learning Rate	Training Acc	Test Acc
0.01	0.87	0.67
0.008	0.92	0.71
0.005	0.94	0.68

The model exhibits overfitting, which we will mitigate in future experiments, for example using early stopping or data augmentation. We present the best of our preliminary attempt to solve this task, there can be many adjustments to be done on the way, but it seems a pretty good start.

Table 8.19 Best accuracy reached with optimal configuration.

model	test accuracy
best	0.75

8.2.2.2 Activity Anticipation

This subsection reports some preliminary experiments regarding the activity anticipation task using the same approach as the recognition task, only anticipating the classification before the completion of an activity.

As an experimental approach, we decided to use and remove a percentage of frames from the end of each video. Because some videos could have become too short, we set a minimal threshold for the minimal allowed number of frames per video. From now on, only videos with more than 10 frames are taken into consideration.

After eliminating these videos, the total number of videos goes from 2617 to 2432 (22% on test). We then create the scene graphs for these videos without considering the last P%

amount of frames. So for example if a video has 10 frames, and the percentage P is 10% we create the scene graph with just the first 18 frames removing only 2 frames; while if a video has 200 frames and the percentage is 10%, the scene graph is created with the first 180 frames and thus removing 20 frames from the video.

We run this experiment for $P = p, p \in 10, 20, 30, 40, 50, 60, 70, 80, 90$ to see the difference in results and how far we can anticipate the decision while keeping a good accuracy. The results of this experiment are shown in Table 8.20, and visualised in Fig. 8.14.

Table 8.20 Accuracy on the prediction task using only a percentage of frames per video. The percentage used is reported in the first column.

Percentage of frames	Validation Accuracy	Test Accuracy
0	0.95	0.79
10	0.96	0.79
20	0.96	0.76
30	0.93	0.73
40	0.96	0.71
50	0.93	0.70
60	0.91	0.67
70	0.94	0.61
80	0.90	0.57
90	0.88	0.54

There is a few things to note from the results shown in Table 8.20 and Fig. 8.14. First, the test accuracy without anticipation has increased from the previously best test accuracy of 0.75 to 0.79. The only thing that changed is the elimination of the 185 videos with less than 10 frames. This indicates that the model struggles to recognise patterns in videos with few numbers of frames and thus struggles to classify them. Whereas, using videos with more frames improves the results.

Next, we can notice that the test accuracy for anticipation with 10% is the very same as the test accuracy without anticipation. This means that by 90% of the video, the model is able to give the same predictions it would have given with the full 100% video.

From 20% on the test accuracy starts to decrease, which is expected considering the model is processing less and less information each time.

At 50% the test accuracy is still higher than 0.7, meaning that the model only needs half of the video to be able to accurately predict 70% of the activities correctly. At 90% – now the model has access to only the first 10% of the video – it is still able to predict more than half

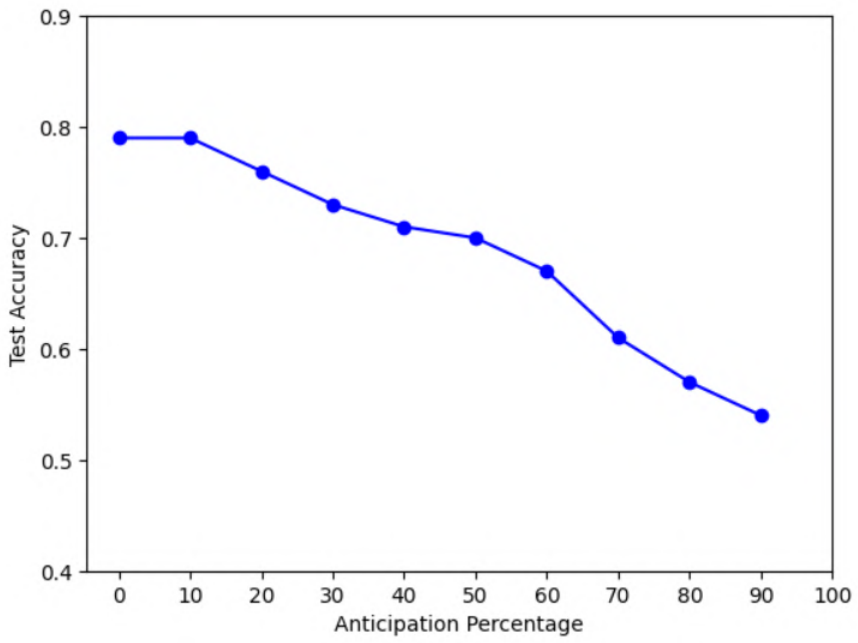


Fig. 8.14 Accuracy of the anticipation task by anticipating the decision at the given percentage. The left part is where the all video is considered, the right where only 10% of the video is observed and the decision taken.

of the classes correctly. Considering that we have 69 classes, this is well above the chance level.

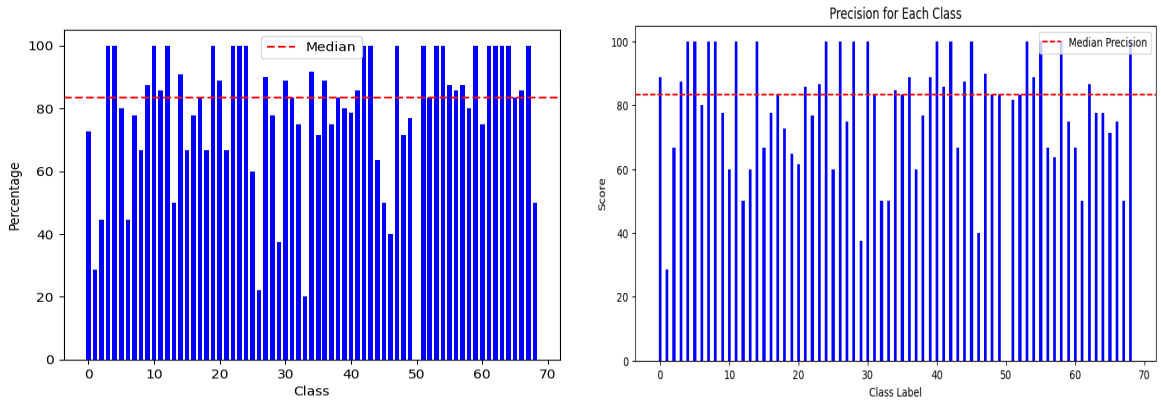
Since the reduced dataset (eliminating the videos with less than 10 frames) leads to better performances with respect to the entire set, we report here some results on the reduced set but using 100% of each video.

The Recall of each class is illustrated in Fig. 8.15a. We can see that 20 classes have a recall of 100% and 1 has 0% while the rest are mostly between 50 and 90%. The median is 83%, while the average is 78.3%, suggesting that on the majority of the classes, our solution performs quite well while in a few of them performs very poorly.

The Precision of each class is illustrated in Fig. 8.15b. We can see that 17 classes have a precision of 100% and 1 has 0%. The median of the precision of all classes is 83%, while the average is 77.8%.

The mean Average Precision (mAP) is then calculated from the precision-recall curve, where each AP is the area under the curve and the mAP is the mean of all APs. We obtain a mAP = 63.1%.

The Confusion Matrix is plotted in Fig. 8.16, showing the percentage of the predicted classes against the actual classes. We can note that the predictions are mostly along the diagonal which means that most of the predictions are correct.



(a) Recall percentage per class.

(b) Precision percentage per class.

Fig. 8.15 Precision and Recall per class, each class is on the x-axis and the y-values are the metric values.

A few examples where classes are confused are listed below:

- class 1: "unpack from a suitcase" was confused 71% of the time with class 29: "pack a suitcase".
- class 29: "Pack a suitcase" was confused 62% of the time with class 1: "unpack a suitcase".
- class 46: "listen to music" was confused 60% of the time with class 57: "use a laptop".
- class 26: "serve dinner" was confused 44% of the time with class 10: "eat dinner" and 33% of the time with class 19: "set table"
- class 25: "load and run dryer" was confused 40% of the time with class 32: "unload drying machine".
- class 13: "work at the table" was confused 25% of the time with class 60: "organize office supplies".

From the examples above, we can clearly see what our algorithm is missing: an oriented temporal component; so not only the temporal connection between frames is important but also its directionality.

8.2.3 Discussion

In this chapter we tackled a well-known problem in computer vision: activity understanding, which is a similar task concerning action classification but refers to longer video sequences

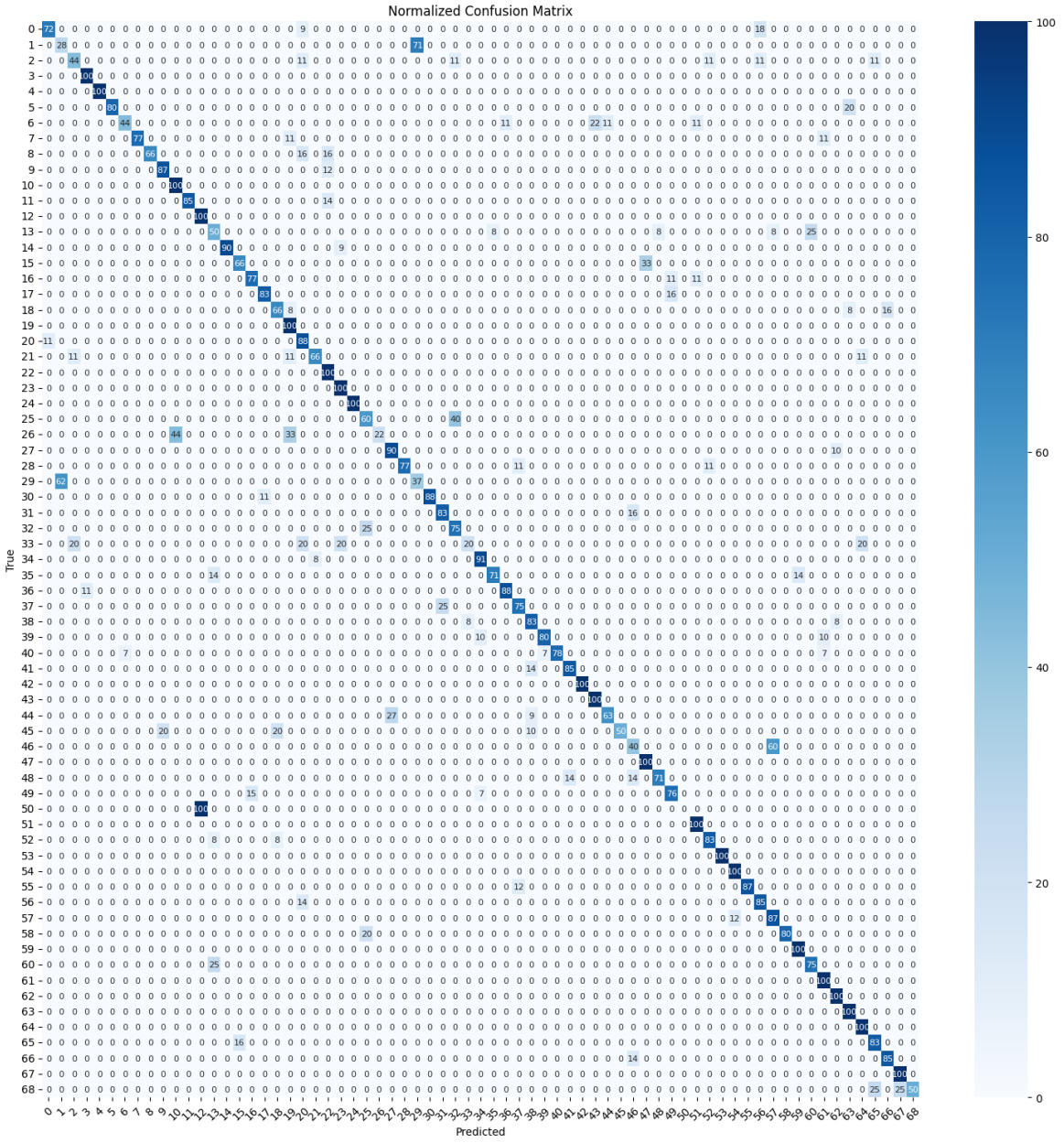


Fig. 8.16 Confusion Matrix highlighting the diagonal except for a few points scattered around which are the misclassified classes. On the rows there are the true values, on the column the predicted, the values are normalised along the raws; so on the diagonal, the accuracy per class is reported.

composed of an ensemble of actions. Our approach in the field is new because people usually tackle this problem directly using deep CNN, or using Deep CNN features like in (Arnab et al., 2021) but new tried to use already pre-extracted bounding boxes and so only semantics of the scene to solve the problem. Our approach is more similar to the one in (Materzynska et al., 2020) where they use a fully connected network to take a global feature for each frame and then aggregate the frames over time with another network. However, their approach fixes the maximum number of objects per frame and the number of frames whereas our graph approach can handle a varying number of objects and frames being much more flexible. We have not compared with them because they faced a different type of problem like ego-vision manipulation and you need to use an ad-hoc detector to recognise only hands. So, our approach has some degree of novelty and achieves promising accuracy, of course, the next step is to build a complete pipeline from RGB frames to activity classification-prediction. Two current drawbacks are the lack of time directionality and the overfitting which is evident from results in Table 8.19 and Table 8.20. The way we think to mitigate the time issue is by inserting a frame number as an edge feature. For the overfitting, we can use early stopping and try a different number of parameters in the network.

Chapter 9

Conclusion

In this thesis we present four different solutions which put the human at the core of analysis, examining cues of social interactions thanks to the HHP-Net algorithm and action recognition through two different perspectives: one relying solely on motion and the other solely on relationships between the human and the environment. Our research places itself in the realm of computer vision and at the intersection of cognitive science and machine learning. Especially in the MOSAIC architecture the inspiration for the chosen approach is clear and tries to replicate some of the human mechanisms.
Recalling the introductory image Fig. 1.1 it is possible to see how this journey touched all four problems and in each of them the keywords of computational aspects or interpretability are somehow present. In the HHP-Net both interpretability -thanks to the embedded uncertainty- and the lightness in computations are of uttermost importance and explicitly stated and proved. Furthermore, the LAEO algorithm is completely designed to be as fast and light as possible with no usage of neural networks that might improve the performances in extreme or degenerated cases. We believe that these two first contributions can be of help in many application scenarios; for example, in a Human-Robot Interaction case they can be used to understand if people in the scene are engaged in some interactions between them or are leaving the stage to the robot to speak/move/do something; furthermore, they can help in predicting the motion of a person inside the robot's area of operation, and maybe prevent harmful behaviours. The second part is focused on human motion from a full-body representation and on activity recognition considering the human in the environment. Whereas the MOSAIC pipeline tries to represent motion as a composition of kinetics primitives to resemble cognitive science principles, the ACCROSS is more focused on the usage of scene graphs as a means to do activity recognition in an interpretable way. The MOSAIC is our first attempt to build a foundation model for human motion understanding based on compositionality of primitives. Natural improvements in this sense are to start from single poses and

then include time, and this is the idea we are going to pursue in the near future. We strongly believe in this approach and we demonstrated that can be accurate as much as state-of-the-art models not relying on compositionality. For the ACCROSS methodology, we believe that the interpretability is embedded in the method itself because it is founded on semantics and in this way it is easily accessible for checking what the network is learning, or looking at when performing predictions. Indeed the method proposes, fuses the features of adjacent nodes and does not use image features (for example extracted from CCNs); according to us, this makes the model easier to inspect. Finally, the test bed scenario in which we performed early activity classification is a promising path to follow for new works. The applications in which these two new pipelines can be applied are numerous: CCTVs, Human Robot Interactions, medical applications for rehabilitation etc.

In the following, we propose a summary of the main technical contributions of each part of the thesis:

- The HHP-Net achieves state-of-the-art results being much smaller than competitors and relying on pre-computed keypoints (the gold standard for human-recognition pipelines, so very often already present). The network is also interchangeable with different pose detectors, as demonstrated by experiments, and it is able to give confidence about its own output.

- Our LAEO algorithm uses very simple geometric concepts to achieve state-of-the-art results on a complex task, thanks to the previous stages which gave as output a very reliable data representation.

- Our MOSAIC architecture tried to privilege the representation of the keypoints of information and took inspiration from cognitive science to be developed. It reached very good performances on the chosen downstream task; however, we glimpse some improvements, especially in the study of the embedding space. Nevertheless, it is an important step towards a full recognition of movements thanks to motion primitives, and to the usage of this new and reach representation.

- The last exploratory part of the thesis had outstanding results on a very promising task which has not been explored in depth in the literature. The major contributions here are the algorithm development and the usage of this dataset to solve this task. We demonstrated it feasible, nevertheless, we foresee new improvements in this regard, especially in the network pipeline, integrating temporal directionally and object features (from a detector) to enhance the performances.

9.1 Limitations and Future works

As we are aware that research is a never-ending journey, we analysed some of the limitations of our works in each of the experimental chapters but we can recall the main points here, to also highlight methodologies or ideas to overcome them in the future. The HHP-Net is very light and efficient and the only future step to strengthen it is the usage of more data, with more variability and many more angle views; or, furthermore, the possibility of relying on the pose estimator to state when a person is seen from behind, enabling the possibility of predicting angles also for these cases.
When regarding the LAEO method, many improvements can be made disregarding the efficiency of the method, for example, the usage of a network doing depth estimation from 2D images. On the contrary, to preserve efficiency, the usage of some heuristics to estimate the depth of people inside the image can be used: e.g. computing their skeleton dimension and assigning to each of them a position in foreground and background, and avoiding detecting interactions between these two sets.
The MOSAIC architecture, as it is, lacks bone connectivity information, which we realised can be beneficial both for the downstream task and for a representation viewpoint. Furthermore, the dataset chosen has a huge variety of classes but some of them have huge semantic overlap which made us interrogate its usage and also the real differences of the data, also in terms of representations. So a natural evolution is the usage of Graphs over humans, as the majority of the literature does, but for the representation part; so encoding the primitives thanks to a graph network and then re-using the same transformer to do the classification. Another direction of improvement we wanted to mention is a training pipeline involving the reconstruction process thanks to a transformer decoder, which we think can be really beneficial from a representation viewpoint. Moreover, some help can also come from the emerging field of disentangled representations, which tries to work with very small representation vectors where each component can be traced back to one of the input parts, for example, a hand, a leg or the head; one implicit assumption is that each part is uncorrelated or very small correlated with others (which does not fully hold for a human body).
Regarding the last exploratory part of the thesis, we have already highlighted one of the main weaknesses which comes out from the results: the lack of temporal directionality. Another improvement is the usage of some deep learning features to be embedded in our graph, for example, some compressed representation extracted from the bounding boxes can be beneficial to give more information in the message passing part.

In conclusion, the thesis proposed an in-depth explanation of four novel methods to study human-centric solutions based on computer vision and deep learning methods. Many exper-

iments support the methodological parts and hopefully, clear explanations and discussion about their strengths and limitations are reported.

References

Abate, Andrea F, Carmen Bisogni, Aniello Castiglione, and Michele Nappi (2022). "Head pose estimation: An extensive survey on recent techniques and applications". In: *Pattern Recognition* 127 (cited on page 34).

Abele, Andrea (1986). "Functions of gaze in social interaction: Communication and monitoring". In: *Jour. of Nonverbal Behavior* 10.2, pages 83–101 (cited on page 43).

Albiero, Vitor, Xingyu Chen, Xi Yin, Guan Pang, and Tal Hassner (2021). "img2pose: Face alignment and detection via 6dof, face pose estimation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7617–7627 (cited on pages 35, 59).

Allingham, Emma, David Hammerschmidt, and Clemens Wöllner (2021). "Time perception in human movement: Effects of speed and agency on duration estimation". In: *Quarterly Journal of Experimental Psychology* 74.3, pages 559–572 (cited on page 88).

Arnab, Anurag, Chen Sun, and Cordelia Schmid (2021). "Unified graph structured models for video understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8117–8126 (cited on pages 83, 124).

Asperti, Andrea and Matteo Trentin (2020). "Balancing reconstruction error and kullback-leibler divergence in variational autoencoders". In: *IEEE Access* 8, pages 199440–199448 (cited on page 103).

Barra, Paola, Silvio Barra, Carmen Bisogni, Maria De Marsico, and Michele Nappi (2020). "Web-shaped model for head pose estimation: An approach for best exemplar selection". In: *TIP* 29, pages 5457–5468 (cited on page 34).

Bazarevsky, Valentin, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann (2020). "Blazepose: On-device real-time body pose tracking". In: *arXiv preprint arXiv:2006.10204* (cited on page 34).

Bertasius, Gedas, Heng Wang, and Lorenzo Torresani (2021). "Is space-time attention all you need for video understanding?" In: *ICML*. Volume 2. 3, page 4 (cited on page 76).

Bisogni, Carmen, Michele Nappi, Chiara Pero, and Stefano Ricciardi (2021). "FASHE: A fractal based strategy for head pose estimation". In: *IEEE Transactions on Image Processing* 30, pages 3192–3203 (cited on page 35).

Blank, Moshe, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri (2005). "Actions as space-time shapes". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Volume 2. IEEE, pages 1395–1402 (cited on page 75).

Bolotta, Samuele and Guillaume Dumas (2022). "Social neuro AI: Social interaction as the "dark matter" of AI". In: *Frontiers in Computer Science* 4, page 846440 (cited on page 31).

Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun (2013). "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (cited on page 81).

Bulat, Adrian and Georgios Tzimiropoulos (2017). "How Far are We from Solving the 2D and 3D Face Alignment Problem? (and a Dataset of 230,000 3D Facial Landmarks)". In: *ICCV* (cited on pages 35, 60, 61).

Campbell, Kenneth L (2012). "The SHRP 2 naturalistic driving study: Addressing driver performance and behavior in traffic safety". In: *Tr News* 282 (cited on page 34).

Cantarini, Giorgio, Federico Figari Tomenotti, Nicoletta Noceti, and Francesca Odone (2022). "HHP-Net: A Light Heteroscedastic Neural Network for Head Pose Estimation With Uncertainty". In: *WACV*, pages 3521–3530 (cited on pages 2, 37).

Cao, Z., G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh (2019). "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *PAMI* (cited on pages 24, 34, 38, 47, 56).

Cao, Zhiwen, Zongcheng Chu, Dongfang Liu, and Yingjie Chen (2021a). "A Vector-based Representation to Enhance Head Pose Estimation". In: *WACV* (cited on page 34).

— (2021b). "A vector-based representation to enhance head pose estimation". In: *CVPR*, pages 1188–1197 (cited on pages 59, 60).

Cardoso, Danilo Barros, Luiza C.B. Campos, and Erickson R. Nascimento (2022). "An Action Recognition Approach with Context and Multiscale Motion Awareness". In: *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Volume 1, pages 73–78. DOI: 10.1109/SIBGRAPI55357.2022.9991807 (cited on pages 104, 113).

Carreira, Joao and Andrew Zisserman (2017). "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308 (cited on page 76).

Chang, Fei, Jiabei Zeng, Qiaoyun Liu, and Shiguang Shan (2023). "Gaze Pattern Recognition in Dyadic Communication". In: *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*, pages 1–7 (cited on pages 33, 62, 64).

Chang, Xiaojun, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann (2021). "A comprehensive survey of scene graphs: Generation and application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1, pages 1–26 (cited on page 83).

Chen, Tianshui, Weihao Yu, Riquan Chen, and Liang Lin (2019). "Knowledge-embedded routing network for scene graph generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6163–6171 (cited on page 82).

Chen, Yuxiao, Long Zhao, Jianbo Yuan, Yu Tian, Zhaoyang Xia, Shijie Geng, Ligong Han, and Dimitris N Metaxas (2022). "Hierarchically self-supervised transformer for human skeleton representation learning". In: *European Conference on Computer Vision*. Springer, pages 185–202 (cited on page 78).

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. Edited by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, pages 1724–1734. DOI: 10.3115/V1/D14-1179. URL: https://doi.org/10.3115/v1/d14-1179 (cited on page 19).

Cho, Sangwoo, Muhammad Maqbool, Fei Liu, and Hassan Foroosh (2020). "Self-attention network for skeleton-based human action recognition". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 635–644 (cited on page 78).

Chong, Eunji, Nataniel Ruiz, Yongxin Wang, Yun Zhang, Agata Rozga, and James M Rehg (2018). "Connecting gaze, scene, and attention: Generalized attention estimation via joint modeling of gaze and scene saliency". In: *Proceedings of the European conference on computer vision (ECCV)*, pages 383–398 (cited on page 33).

Chong, Eunji, Yongxin Wang, Nataniel Ruiz, and James M Rehg (2020). "Detecting Attended Visual Targets in Video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5396–5406 (cited on page 33).

Cieślik, Karol and Marian J Łopatka (2022). "Research on speed and acceleration of hand movements as command signals for anthropomorphic manipulators as a master-slave system". In: *Applied Sciences* 12.8, page 3863 (cited on page 88).

Cipolla, R., Y. Gal, and A. Kendall (2018). "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *CVPR*, pages 7482–7491 (cited on page 36).

Colyer, Steffi L, Murray Evans, Darren P Cosker, and Aki IT Salo (2018). "A review of the evolution of vision-based motion analysis and the integration of advanced computer vision methods towards developing a markerless system". In: *Sports medicine-open* 4.1, pages 1–15 (cited on page 34).

Cong, Yuren, Michael Ying Yang, and Bodo Rosenhahn (2023). "Reltr: Relation transformer for scene graph generation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (cited on page 82).

Corbellini, Nicola, Eleonora Ceccaldi, Giovanna Varni, and Gualtiero Volpe (2022). "An exploratory study on group potency classification from non-verbal social behaviours". In: *International Conference on Pattern Recognition*. Springer, pages 240–255 (cited on page 31).

Cristianini, Nello (2023). *The Shortcut: Why Intelligent Machines Do Not Think Like Us*. CRC Press (cited on page 31).

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pages 303–314 (cited on page 10).

Debnath, Bappaditya, Mary O'brien, Motonori Yamaguchi, and Ardhendu Behera (2022). "A review of computer vision-based approaches for physical rehabilitation and assessment". In: *Multimedia Systems* 28.1, pages 209–239 (cited on page 1).

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems* 29 (cited on page 81).

Dessalene, Eadom, Michael Maynord, Cornelia Fermüller, and Yiannis Aloimonos (2023). "Therbligs in action: Video understanding through motion primitives". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10618–10626 (cited on page 81).

Dhingra, Naina (2022). "LwPosr: Lightweight Efficient Fine Grained Head Pose Estimation". In: *WACV*, pages 1495–1505 (cited on pages 34, 35, 59, 60).

Dias, Philipe A., Damiano Malafronte, Henry Medeiros, and Francesca Odone (2020). "Gaze Estimation for Assisted Living Environments". In: *WACV* (cited on pages 35, 36, 39, 51).

Diba, Ali, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool (2017). "Temporal 3d convnets: New architecture and transfer learning for video classification". In: *arXiv preprint arXiv:1711.08200* (cited on page 76).

Doosti, Bardia, Ching-Hui Chen, Raviteja Vemulapalli, Xuhui Jia, Yukun Zhu, and Bradley Green (2021). "Boosting Image-based Mutual Gaze Detection using Pseudo 3D Gaze". In: *AAAI*, pages 1273–1281 (cited on page 33).

Drouard, Vincent, Silèye Ba, Georgios Evangelidis, Antoine Deleforge, and Radu Horaud (2015). "Head Pose Estimation via Probabilistic High-Dimensional Regression". In: *ICIP*, pages 4624–4628 (cited on pages 59, 60).

Du, Yong, Wei Wang, and Liang Wang (2015). "Hierarchical recurrent neural network for skeleton based action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118 (cited on page 77).

Duan, Haodong, Jiaqi Wang, Kai Chen, and Dahua Lin (2022). "DG-STGCN: dynamic spatial-temporal modeling for skeleton-based action recognition". In: *arXiv preprint arXiv:2210.05895* (cited on page 113).

Duan, Kaiwen, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian (2019). "CenterNet: Keypoint Triplets for Object Detection". In: *ICCV* (cited on pages 34, 38, 56, 62).

Duvenaud, David Kristjanson (1999). *Bayesina Neural Network*. URL: https://www.cs.toronto.edu/~duvenaud/distill_bayes_net/public/ (visited on 01/20/2024) (cited on page 15).

Fan, Lifeng, Wenguan Wang, Siyuan Huang, Xinyu Tang, and Song-Chun Zhu (2019). "Understanding human gaze communication by spatio-temporal graph reasoning". In: *Proceedings of the IEEE International Conference on Computer Vision*, pages 5724–5733 (cited on page 33).

Fan, Xiaolong, Maoguo Gong, Yu Xie, Fenlong Jiang, and Hao Li (2020). "Structured self-attention architecture for graph-level representation learning". In: *Pattern Recognition* 100, page 107084 (cited on page 81).

Fanelli, G., M. Dantone, J. Gall, A. Fossati, and L. van Gool (2013). "Random Forests for Real Time 3D Face Analysis". In: *IJCV* 101.3, pages 437–458 (cited on pages 35, 49, 60, 61).

Fanelli, Gabriele, Thibaut Weise, Juergen Gall, and Luc Van Gool (2011). "Real time head pose estimation from consumer depth cameras". In: *Joint PR symp.* Pages 101–110 (cited on pages 34, 48).

Fang, Hao-Shu, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu (2022). "Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (cited on page 34).

Al-Faris, Mahmoud, John Chiverton, David Ndzi, and Ahmed Isam Ahmed (2020). "A review on computer vision-based methods for human action recognition". In: *Journal of imaging* 6.6, page 46 (cited on page 71).

Feichtenhofer, Christoph, Haoqi Fan, Jitendra Malik, and Kaiming He (2019). "Slowfast networks for video recognition". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211 (cited on page 76).

Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman (2016). "Convolutional two-stream network fusion for video action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941 (cited on page 76).

Feng, Di, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer (2019). "Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3d object detection". In: *Intelligent Vehicles Symp.* Pages 1280–1287 (cited on page 52).

Fernández, Alberto, Rubén Usamentiaga, Juan Luis Carús, and Rubén Casado (2016). "Driver distraction using visual-based sensors and algorithms". In: *Sensors* 16.11, page 1805 (cited on page 38).

Figari Tomenotti, Federico, Nicoletta Noceti, and Francesca Odone (2024). "Head pose estimation with uncertainty and an application to dyadic interaction detection". In: *Computer Vision and Image Understanding* 243, page 103999. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2024.103999. URL: https://www.sciencedirect.com/science/article/pii/S1077314224000808 (cited on pages 2, 37, 39, 40, 43, 48, 50, 51, 53, 55, 63, 65).

Fogassi, Leonardo, Pier Francesco Ferrari, Benno Gesierich, Stefano Rozzi, Fabian Chersi, and Giacomo Rizzolatti (2005). "Parietal lobe: from action organization to intention understanding". In: *Science* 308.5722, pages 662–667 (cited on page 80).

Fu, Huan, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao (no date). "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 66).

Garello, Luca, Matteo Moro, Chiara Tacchino, Francesca Campone, Paola Durand, Isabella Blanchi, Paolo Moretti, Maura Casadio, and Francesca Odone (2021). "A Study of At-term and Preterm Infants' Motion Based on Markerless Video Analysis". In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, pages 1196–1200 (cited on page 1).

Gasparetto, Andrea, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli (2022). "A survey on text classification algorithms: From text to predictions". In: *Information* 13.2, page 83 (cited on page 89).

Gaveau, Jérémie and Charalambos Papaxanthis (2011). "The temporal structure of vertical arm movements". In: *PLoS One* 6.7, e22045 (cited on page 88).

Girshick, Ross (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448 (cited on page 82).

Golda, Thomas, Johanna Thiemich, Mickael Cormier, and Jürgen Beyerer (2022). "For the Sake of Privacy: Skeleton-Based Salient Behavior Recognition". In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, pages 3983–3987 (cited on page 78).

Gong, Wenjuan, Xuena Zhang, Jordi Gonzàlez, Andrews Sobral, Thierry Bouwmans, Changhe Tu, and El-hadi Zahzah (2016). "Human pose estimation from monocular images: A comprehensive survey". In: *Sensors* 16.12, page 1966 (cited on page 34).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cited on page 11).

Gu, Chunhui et al. (2018). "Ava: A video dataset of spatio-temporally localized atomic visual actions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6047–6056 (cited on page 83).

Gu, J., X. Yang, S. De Mello, and J. Kautz (2017). "Dynamic Facial Analysis: From Bayesian Filtering to Recurrent Neural Network". In: *CVPR*, pages 1531–1540 (cited on pages 34, 59, 60).

Guo, Hang, Zhengxi Hu, and Jingtai Liu (2022). "MGTR: End-to-End Mutual Gaze Detection with Transformer". In: *Proceedings of the Asian Conference on Computer Vision*, pages 1590–1605 (cited on pages 33, 34).

Gupta, Suresh Chand, Deepak Kumar, and VijayAnant Athavale (2021). "A review on human action recognition approaches". In: *2021 10th IEEE International Conference on*

Communication Systems and Network Technologies (CSNT). IEEE, pages 338–344 (cited on page 71).

Hard, Bridgette Martin, Sandra C Lozano, and Barbara Tversky (2006). "Hierarchical encoding of behavior: translating perception into action." In: *Journal of Experimental Psychology: General* 135.4, page 588 (cited on page 80).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778 (cited on page 14).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pages 1735–1780. DOI: 10.1162/NECO.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cited on page 19).

Hong, Chaoqun, Jun Yu, Jian Zhang, Xiongnan Jin, and Kyong-Ho Lee (2018). "Multimodal face-pose estimation with multitask manifold deep learning". In: *IEEE transactions on industrial informatics* 15.7, pages 3952–3961 (cited on page 34).

Hsu, Heng-Wei, Tung-Yu Wu, Sheng Wan, Wing Hung Wong, and Chen-Yi Lee (2018). "Quatnet: Quaternion-based head pose estimation with multiregression loss". In: *IEEE Transactions on Multimedia* 21.4, pages 1035–1046 (cited on pages 35, 59).

Ikizler, Nazli and David Alexander Forsyth (2008). "Searching for Complex Human Activities with No Visual Examples". In: *International Journal of Computer Vision* 80, pages 337–357. URL: https://api.semanticscholar.org/CorpusID:10706743 (cited on page 80).

Jacobs, Robert A, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton (1991). "Adaptive mixtures of local experts". In: *Neural computation* 3.1, pages 79–87 (cited on page 81).

Jospin, Laurent Valentin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun (2022). "Hands-on Bayesian neural networks—A tutorial for deep learning users". In: *IEEE Computational Intelligence Magazine* 17.2, pages 29–48 (cited on page 15).

Ju, Jianping, Hong Zheng, Congcong Li, Xi Li, Hai Liu, and Tingting Liu (2022). "AGCNNs: Attention-guided convolutional neural networks for infrared head pose estimation in assisted driving system". In: *Infrared Physics & Technology* 123 (cited on page 35).

Kazemi, Vahid and Josephine Sullivan (2014). "One millisecond face alignment with an ensemble of regression trees". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874 (cited on pages 60, 61).

Ke, Bingxin, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler (2024). "Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 66).

Kendall, Alex and Yarin Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Adv. in Neural Information Processing Systems*. Volume 30 (cited on pages 36, 41, 42).

Kingma, Diederik P and Max Welling (2013a). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (cited on page 17).

— (2013b). "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114. URL: https://api.semanticscholar.org/CorpusID:216078090 (cited on page 18).

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (cited on pages 23, 81).

Kong, Yu and Yun Fu (2022). "Human action recognition and prediction: A survey". In: *International Journal of Computer Vision* 130.5, pages 1366–1401 (cited on page 75).

Krishna, Ranjay et al. (2017). "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International journal of computer vision* 123, pages 32–73 (cited on page 83).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (cited on page 14).

Kuehne, H., H. Jhuang, E. Garrote, T. Poggio, and T. Serre (2011). "HMDB: a large video database for human motion recognition". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (cited on page 78).

Kukleva, Anna, Makarand Tapaswi, and Ivan Laptev (2020). "Learning interactions and relationships between movie characters". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9849–9858 (cited on page 33).

Kulic, Dana and Yoshihiko Nakamura (2008). "Scaffolding on-line segmentation of full body human motion patterns". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pages 2860–2866 (cited on pages 81, 88).

Kulić, Dana, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura (2012). "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *The International Journal of Robotics Research* 31.3, pages 330–345 (cited on pages 81, 88).

Kumar, A., A. Alavi, and R. Chellappa (2017). "KEPLER: Keypoint and Pose Estimation of Unconstrained Faces by Learning Efficient H-CNN Regressors". In: *Int. Conf. on Automatic Face Gesture Recognition*, pages 258–265 (cited on pages 36, 60, 61).

Kuznetsova, Alina et al. (2020). "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale". In: *International journal of computer vision* 128.7, pages 1956–1981 (cited on page 83).

Lan, Tian, Yuke Zhu, Amir Zamir, and Silvio Savarese (2015). "Action Recognition by Hierarchical Mid-Level Action Elements". In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4552–4560. URL: https://api.semanticscholar.org/CorpusID:311006 (cited on page 80).

Lathuiliere, Stephane, Remi Juge, Pablo Mesejo, Rafael Munoz-Salinas, and Radu Horaud (2017). "Deep Mixture of Linear Inverse Regressions Applied to Head-Pose Estimation". In: *CVPR* (cited on pages 59, 60).

Lea, Colin, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager (2017). "Temporal convolutional networks for action segmentation and detection". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165 (cited on page 77).

Lee, Kin Man, Arjun Krishna, Zulfiqar Zaidi, Rohan Paleja, Letian Chen, Erin Hedlund-Botti, Mariah Schrum, and Matthew Gombolay (2023). "The effect of robot skill level and communication in rapid, proximate human-robot collaboration". In: *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pages 261–270 (cited on page 1).

Li, Guohao, Chenxin Xiong, Ali Thabet, and Bernard Ghanem (2020). "Deepergcn: All you need to train deeper gcns". In: *arXiv preprint arXiv:2006.07739* (cited on page 95).

Li, Maosen, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian (2019). "Actional-structural graph convolutional networks for skeleton-based action recognition". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3595–3603 (cited on page 77).

Liao, Wentong, Shuai Lin, Bodo Rosenhahn, and Michael Ying Yang (2017). "Natural Language Guided Visual Relationship Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 444–453. URL: https://api.semanticscholar.org/CorpusID:28440081 (cited on page 83).

Lillo, Ivan, Álvaro Soto, and Juan Carlos Niebles (2014). "Discriminative Hierarchical Modeling of Spatio-temporally Composable Human Activities". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 812–819. URL: https://api.semanticscholar.org/CorpusID:1526478 (cited on page 80).

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár (2017a). "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988 (cited on page 90).

— (2017b). "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988 (cited on page 106).

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, pages 740–755 (cited on page 34).

Liu, Hai, Shuai Fang, Zhaoli Zhang, Duantengchuan Li, Ke Lin, and Jiazhang Wang (2021). "MFDNet: Collaborative poses perception and matrix Fisher distribution for head pose estimation". In: *IEEE Transactions on Multimedia* 24, pages 2449–2460 (cited on page 34).

Liu, Hai, Tingting Liu, Zhaoli Zhang, Arun Kumar Sangaiah, Bing Yang, and Youfu Li (2022). "Arhpe: Asymmetric relation-aware representation learning for head pose estimation in industrial human–computer interaction". In: *IEEE Transactions on Industrial Informatics* 18.10, pages 7107–7117 (cited on page 34).

Liu, Hai, Cheng Zhang, Yongjian Deng, Tingting Liu, Zhaoli Zhang, and You-Fu Li (2023). "Orientation Cues-Aware Facial Relationship Representation for Head Pose Estimation via Transformer". In: *IEEE Transactions on Image Processing* 32, pages 6289–6302 (cited on page 35).

Liu, Hengyue, Ning Yan, Masood Mortazavi, and Bir Bhanu (2021). "Fully convolutional scene graph generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11546–11556 (cited on page 82).

Liu, Jun, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot (2019). "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding". In: *IEEE transactions on pattern analysis and machine intelligence* 42.10, pages 2684–2701 (cited on pages 79, 98, 99).

Liu, Jun, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, and Alex C Kot (2017). "Skeleton-based human action recognition with global context-aware attention LSTM networks". In: *IEEE Transactions on Image Processing* 27.4, pages 1586–1599 (cited on page 77).

Liu, Tingting, Jixin Wang, Bing Yang, and Xuan Wang (2021). "NGDNet: Nonuniform Gaussian-label distribution learning for infrared head pose estimation and on-task behavior understanding in the classroom". In: *Neurocomputing* 436, pages 210–220 (cited on page 34).

Loper, Matthew, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black (Oct. 2015). "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6, 248:1–248:16 (cited on page 99).

Lozano, Sandra C, Bridgette Martin Hard, and Barbara Tversky (2006). "Perspective taking promotes action understanding and learning." In: *Journal of Experimental Psychology: Human Perception and Performance* 32.6, page 1405 (cited on page 80).

Lugaresi, Camillo et al. (2019). "Mediapipe: A framework for building perception pipelines". In: *arXiv preprint arXiv:1906.08172* (cited on pages 34, 38, 56, 62).

Luo, Yue, Jimmy Ren, Zhouxia Wang, Wenxiu Sun, Jinshan Pan, Jianbo Liu, Jiahao Pang, and Liang Lin (2018). "Lstm pose machines". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5207–5215 (cited on page 34).

Luvizon, Diogo C, David Picard, and Hedi Tabia (2020). "Multi-task deep learning for real-time 3D human pose estimation and action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 43.8, pages 2752–2764 (cited on page 34).

MacKay, David J. C. (1992). "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Comput.* 4.3, pages 448–472 (cited on page 41).

Madrigal, Francisco and Frederic Lerasle (2020). "Robust head pose estimation based on key frames for human-machine interaction". In: *EURASIP Journal on Image and Video Processing* 2020, pages 1–19 (cited on page 34).

Mahmood, Naureen, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black (Oct. 2019). "AMASS: Archive of Motion Capture as Surface Shapes". In: *International Conference on Computer Vision*, pages 5442–5451 (cited on page 98).

Manuel Marin-Jimenez, Andrew Zisserman and Vittorio Ferrari (2011). ""Here's looking at you, kid". Detecting people looking at each other in videos". In: *Proceedings of the British Machine Vision Conference*. http://dx.doi.org/10.5244/C.25.22. BMVA Press, pages 22.1–22.12. ISBN: 1-901725-43-X (cited on page 33).

Marin-Jimenez, Manuel J, Vicky Kalogeiton, Pablo Medina-Suarez, and Andrew Zisserman (2019). "Laeo-net: revisiting people looking at each other in videos". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3477–3485 (cited on pages 33, 62, 63, 64).

Marín-Jiménez, Manuel, Vicky Kalogeiton, Pablo Medina-Suárez, and Andrew Zisserman (2020). "LAEO-Net++: revisiting people Looking At Each Other in videos". In: *PAMI*, pages 1–16 (cited on pages 33, 43, 62, 64).

Marín-Jiménez, Manuel Jesús, Andrew Zisserman, Marcin Eichner, and Vittorio Ferrari (2014). "Detecting people looking at each other in videos". In: *Int. J. of Comp. Vis.* 106.3, pages 282–296 (cited on page 33).

Martin Koestinger Paul Wohlhart, Peter M. Roth and Horst Bischof (2011). "Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization". In: *Int. Work. on Benchmarking Facial Image Analysis Technologies* (cited on page 48).

Massé, B., S. Ba, and R. Horaud (2018). "Tracking Gaze and Visual Focus of Attention of People Involved in Social Interaction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.11, pages 2711–2724. DOI: 10.1109/TPAMI.2017.2782819 (cited on page 32).

Materzynska, Joanna, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell (2020). "Something-else: Compositional action recognition with spatial-temporal interaction networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1049–1059 (cited on pages 83, 124).

McMahon, Emalie and Leyla Isik (2023). "Seeing social interactions". In: *Trends in Cognitive Sciences* (cited on page 32).

Moro, Matteo, Giorgia Marchesi, Filip Hesse, Francesca Odone, and Maura Casadio (2022). "Markerless vs. Marker-Based Gait Analysis: A Proof of Concept Study". In: *Sensors* 22.5 (cited on page 34).

Mukherjee, Sankha S. and Neil Martin Robertson (Nov. 2015). "Deep Head Pose: Gaze-Direction Estimation in Multimodal Video". In: *IEEE Transactions on Multimedia* 17.11, pages 2094–2107 (cited on pages 34, 59, 60).

Nair, Vipul et al. (2020). "Action similarity judgment based on kinematic primitives". In: *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, pages 1–8 (cited on pages 80, 85, 88).

Natale, Lorenzo, Chiara Bartolozzi, Francesco Nori, Giulio Sandini, and Giorgio Metta (2019). "iCub". In: *Humanoid Robotics: A Reference*. Edited by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, pages 291–323. ISBN: 978-94-007-6046-2. DOI: 10.1007/978-94-007-6046-2_21 (cited on page 64).

Newtson, Darren and Gretchen Engquist (1976). "The perceptual organization of ongoing behavior". In: *Journal of Experimental Social Psychology* 12.5, pages 436–450 (cited on pages 71, 80).

Nix, D.A. and A.S. Weigend (1994). "Estimating the mean and variance of the target probability distribution". In: *ICNN* (cited on page 41).

Othman, Uqba and Erfu Yang (2023). "Human–robot collaborations in smart manufacturing environments: review and outlook". In: *Sensors* 23.12, page 5663 (cited on page 1).

Park, Hyun, Eakta Jain, and Yaser Sheikh (2012). "3d social saliency from head-mounted cameras". In: *Advances in Neural Information Processing Systems* 25, pages 422–430 (cited on page 32).

Plizzari, Chiara, Marco Cannici, and Matteo Matteucci (2021). "Skeleton-based action recognition via spatial and temporal transformer networks". In: *Computer Vision and Image Understanding* 208, page 103219 (cited on page 77).

Punnakkal, Abhinanda R, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J Black (2021). "BABEL: Bodies, action and behavior with english labels". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 722–731 (cited on pages 98, 101, 106, 113).

Rahmaniar, Wahyu, Qazi Mazhar ul Haq, and Ting-Lan Lin (2022). "Wide Range Head Pose Estimation Using A Single RGB Camera for Intelligent Surveillance". In: *Sensors* (cited on page 35).

Rai, Nishant, Haofeng Chen, Jingwei Ji, Rishi Desai, Kazuki Kozuka, Shun Ishizaka, Ehsan Adeli, and Juan Carlos Niebles (2021). "Home action genome: Cooperative compositional action understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11184–11193 (cited on pages 94, 116).

Ranjan, R., V. M. Patel, and R. Chellappa (2019). "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition". In: *PAMI* 41, pages 121–135 (cited on page 36).

Recasens, A., A. Khosla, C. Vondrick, and A. Torralba (2015). "Where are they looking?" In: *NIPS* (cited on page 35).

Recasens, Adria, Aditya Khosla, Carl Vondrick, and Antonio Torralba (2015). "Where are they looking?" In: *Advances in Neural Information Processing Systems (NIPS)* (cited on page 32).

Robinson, Nicole, Brendan Tidd, Dylan Campbell, Dana Kulić, and Peter Corke (2023). "Robotic vision for human-robot interaction and collaboration: A survey and systematic

review". In: *ACM Transactions on Human-Robot Interaction* 12.1, pages 1–66 (cited on page 1).
Ruan, Zeyu, Changqing Zou, Longhai Wu, Gangshan Wu, and Limin Wang (2021). "Sadrnet: Self-aligned dual face regression networks for robust 3d dense face alignment and reconstruction". In: *IEEE Transactions on Image Processing* 30, pages 5793–5806 (cited on page 35).
Ruiz, Nataniel, Eunji Chong, and James M. Rehg (2018). "Fine-Grained Head Pose Estimation Without Keypoints". In: *CVPR-W* (cited on pages 35, 36, 57, 59, 60).
Sagonas, Christos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic (2013). "300 faces in-the-wild challenge: The first facial landmark localization challenge". In: *ICCV-W*, pages 397–403 (cited on page 49).
Saunders, Ben, Necati Cihan Camgoz, and Richard Bowden (2021). "Mixed signals: Sign language production via a mixture of motion primitives". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1919–1929 (cited on pages 81, 115).
Schaal, Stefan (2006). "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics". In: *Adaptive motion of animals and machines*. Springer, pages 261–280 (cited on page 81).
Şengönül, Erkan, Refik Samet, Qasem Abu Al-Haija, Ali Alqahtani, Badraddin Alturki, and Abdulaziz A Alsulami (2023). "An analysis of artificial intelligence techniques in surveillance video anomaly detection: A comprehensive survey". In: *Applied Sciences* 13.8, page 4956 (cited on page 1).
Shabaninia, Elham, Hossein Nezamabadi-pour, and Fatemeh Shafizadegan (2022). "Transformers in Action Recognition: A Review on Temporal Modeling". In: *arXiv preprint arXiv:2302.01921* (cited on page 76).
Shahroudy, Amir, Jun Liu, Tian-Tsong Ng, and Gang Wang (2016). "Ntu rgb+ d: A large scale dataset for 3d human activity analysis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019 (cited on pages 79, 87).
Shao, Mingzhen, Zhun Sun, Mete Ozay, and Takayuki Okatani (2019). "Improving head pose estimation with a combined loss and bounding box margin adjustment". In: *Int. Conf. on Automatic Face Gesture Recognition* (cited on pages 35, 59, 60).
Shi, Lei, Yifan Zhang, Jian Cheng, and Hanqing Lu (2019). "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *CVPR* (cited on page 34).
Si, Chenyang, Ya Jing, Wei Wang, Liang Wang, and Tieniu Tan (2018). "Skeleton-based action recognition with spatial reasoning and temporal stack learning". In: *Proceedings of the European conference on computer vision (ECCV)*, pages 103–118 (cited on page 77).
Simonyan, Karen and Andrew Zisserman (2014a). "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems* 27 (cited on page 75).
— (2014b). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556. URL: https://api.semanticscholar.org/CorpusID:14124313 (cited on page 14).
Song, Ziyang, Ziyi Yin, Zejian Yuan, Chong Zhang, Wanchao Chi, Yonggen Ling, and Shenghao Zhang (2021). "Attention-Oriented Action Recognition for Real- Time Human-Robot Interaction". In: *ICPR*, pages 7087–7094 (cited on page 34).
Soo Park, Hyun and Jianbo Shi (2015). "Social saliency prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4777–4785 (cited on page 32).

Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah (2012). "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (cited on page 78).

Speer, Nicole K, Khena M Swallow, and Jeffery M Zacks (2003). "Activation of human motion processing areas during event perception". In: *Cognitive, Affective, & Behavioral Neuroscience* 3, pages 335–345 (cited on page 80).

Suma, Dr V (2019). "Computer vision for human-machine interaction-review". In: *Journal of Trends in Computer Science and Smart Technology* 1.2, pages 131–139 (cited on page 31).

Summers-Stay, Douglas, Ching L Teo, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos (2012). "Using a minimal action grammar for activity understanding in the real world". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pages 4104–4111 (cited on page 80).

Sun, Zehua, Qiuhong Ke, Hossein Rahmani, Mohammed Bennamoun, Gang Wang, and Jun Liu (2022). "Human action recognition from various data modalities: A review". In: *IEEE transactions on pattern analysis and machine intelligence* (cited on page 75).

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9 (cited on page 14).

Tian, Yingjie, Yuqi Zhang, and Haibin Zhang (2023). "Recent advances in stochastic gradient descent in deep learning". In: *Mathematics* 11.3, page 682 (cited on page 11).

Trabelsi, Rim, Jagannadan Varadarajan, Yong Pei, Le Zhang, Issam Jabri, Ammar Bouallegue, and Pierre Moulin (2017). "Robust Multi-Modal Cues for Dyadic Human Interaction Recognition". In: *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*. ACM, pages 47–53 (cited on page 33).

Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri (2015). "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497 (cited on page 76).

Ulhaq, Anwaar, Naveed Akhtar, Ganna Pogrebna, and Ajmal Mian (2022). "Vision transformers for action recognition: A survey". In: *arXiv preprint arXiv:2209.05700* (cited on page 76).

Varol, Gül, Ivan Laptev, and Cordelia Schmid (2017). "Long-term temporal convolutions for action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 40.6, pages 1510–1517 (cited on page 76).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30 (cited on pages 19, 21, 22).

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (cited on page 24).

Vinciarelli, Alessandro, Maja Pantic, and Hervé Bourlard (2009). "Social signal processing: Survey of an emerging domain". In: *Image and vision computing* 27.12, pages 1743–1759 (cited on page 31).

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2015). "Show and tell: A neural image caption generator". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society,

pages 3156–3164. DOI: 10.1109/CVPR.2015.7298935. URL: https://doi.org/10.1109/CVPR.2015.7298935 (cited on page 19).

Wang, Jinbao, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao (2021). "Deep 3D human pose estimation: A review". In: *Computer Vision and Image Understanding* 210, page 103225 (cited on page 34).

Wang, Limin, Yu Qiao, and Xiaoou Tang (2015). "Action recognition with trajectory-pooled deep-convolutional descriptors". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314 (cited on page 76).

Weinland, Daniel, Remi Ronfard, and Edmond Boyer (2011). "A survey of vision-based methods for action representation, segmentation and recognition". In: *Computer vision and image understanding* 115.2, pages 224–241 (cited on page 71).

Wörgötter, Florentin, F Ziaeetabar, S Pfeiffer, O Kaya, T Kulvicius, and M Tamosiunaite (2020a). "Humans predict action using grammar-like structures". In: *Scientific reports* 10.1, pages 1–11 (cited on page 85).

Wörgötter, Florentin, Fatemeh Ziaeetabar, S Pfeiffer, O Kaya, T Kulvicius, and M Tamosiunaite (2020b). "Humans predict action using grammar-like structures". In: *Scientific reports* 10.1, page 3999 (cited on pages 80, 81).

Xia, Jiahao, Haimin Zhang, Shiping Wen, Shuo Yang, and Min Xu (2022). "An efficient multitask neural network for face alignment, head pose estimation and face tracking". In: *Expert Systems with Applications* (cited on page 36).

Xin, Miao, Shentong Mo, and Yuanze Lin (2021). "Eva-gcn: Head pose estimation based on graph convolutional networks". In: *CVPR*, pages 1462–1471 (cited on page 61).

Xu, Yuanquan, Cheolkon Jung, and Yakun Chang (2022). "Head pose estimation using deep neural networks and 3D point clouds". In: *Pattern Recognition* 121 (cited on page 34).

Yan, Shen, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid (2022). "Multiview transformers for video recognition". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3333–3343 (cited on page 76).

Yan, Sijie, Yuanjun Xiong, and Dahua Lin (2018). "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *Proceedings of the AAAI conference on artificial intelligence*. Volume 32. 1 (cited on page 77).

Yang, Fan, Yang Wu, Sakriani Sakti, and Satoshi Nakamura (2019). "Make skeleton-based action recognition model smaller, faster and better". In: *Proceedings of the ACM multimedia asia*, pages 1–6 (cited on page 77).

Yang, Tsun-Yi, Yi-Ting Chen, Yen-Yu Lin, and Yung-Yu Chuang (2019). "FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation From a Single Image". In: *CVPR* (cited on pages 35, 49, 59, 60).

Yang, Wei, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang (2016). "End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3073–3082 (cited on page 34).

Yang, Yuyi, Yanzhang Wang, Yilin Cao, Zhiyang Zhao, Xinpeng Liu, Yihong Wang, Haiyang Zhang, and Yushan Pan (2023). "Human Robot Collaboration in Industrial Applications". In: *2023 9th International Conference on Virtual Reality (ICVR)*. IEEE, pages 247–255 (cited on page 1).

Yin, Guojun, Lu Sheng, Bin Liu, Nenghai Yu, Xiaogang Wang, Jing Shao, and Chen Change Loy (2018). "Zoom-net: Mining deep feature interactions for visual relationship recogni-

tion". In: *Proceedings of the European conference on computer vision (ECCV)*, pages 322–338 (cited on page 83).
Yin, Xi, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker (2017). "Towards Large-Pose Face Frontalization in the Wild". In: *ICCV* (cited on page 48).
Yu, Jun, Chaoqun Hong, Yong Rui, and Dacheng Tao (2017). "Multitask autoencoder model for recovering human poses". In: *IEEE Transactions on Industrial Electronics* 65.6, pages 5060–5068 (cited on page 34).
Yu, Ruichi, Ang Li, Vlad I Morariu, and Larry S Davis (2017). "Visual relationship detection with internal and external linguistic knowledge distillation". In: *Proceedings of the IEEE international conference on computer vision*, pages 1974–1982 (cited on page 83).
Zacks, Jeffrey M, Todd S Braver, Margaret A Sheridan, David I Donaldson, Abraham Z Snyder, John M Ollinger, Randy L Buckner, and Marcus E Raichle (2001). "Human brain activity time-locked to perceptual event boundaries". In: *Nature neuroscience* 4.6, pages 651–655 (cited on pages 80, 88).
Zacks, Jeffrey M and Khena M Swallow (2007). "Event segmentation". In: *Current directions in psychological science* 16.2, pages 80–84 (cited on pages 71, 80).
Zellers, Rowan, Mark Yatskar, Sam Thomson, and Yejin Choi (2018). "Neural motifs: Scene graph parsing with global context". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5831–5840 (cited on page 83).
Zhang, Aston, Zachary C Lipton, Mu Li, and Alexander J Smola (2021). "Dive into deep learning". In: *arXiv preprint arXiv:2106.11342* (cited on page 11).
Zhang, Bowen, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang (2016). "Real-time action recognition with enhanced motion vector CNNs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2718–2726 (cited on page 76).
Zhang, Hao, Mengmeng Wang, Yong Liu, and Yi Yuan (2020). "FDN: Feature Decoupling Network for Head Pose Estimation". In: *AAAI*, pages 12789–12796 (cited on pages 34, 59, 60).
Zhang, Ji, Kevin Shih, Andrew Tao, Bryan Catanzaro, and Ahmed Elgammal (2018). "An interpretable model for scene graph generation". In: *arXiv preprint arXiv:1811.09543* (cited on page 82).
Zhang, Jiaxu, Wei Xie, Chao Wang, Ruide Tu, and Zhigang Tu (2023). "Graph-aware transformer for skeleton-based action recognition". In: *The Visual Computer* 39.10, pages 4501–4512 (cited on page 78).
Zhang, Xikun, Chang Xu, Xinmei Tian, and Dacheng Tao (2019). "Graph edge convolutional neural networks for skeleton-based action recognition". In: *IEEE transactions on neural networks and learning systems* 31.8, pages 3047–3060 (cited on page 77).
Zhang, Yong, Yingwei Pan, Ting Yao, Rui Huang, Tao Mei, and Chang-Wen Chen (2023). "Learning to generate language-supervised and open-vocabulary scene graph using pre-trained visual-semantic space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2915–2924 (cited on page 83).
Zhao, Hao, Ming Lu, Anbang Yao, Yurong Chen, and Li Zhang (2019). "Learning to draw sight lines". In: *International Journal of Computer Vision*, pages 1–25 (cited on page 32).
Zheng, Ce, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah (2023). "Deep learning-based human pose estimation: A survey". In: *ACM Computing Surveys* 56.1, pages 1–37 (cited on page 34).
Zhou, Huifen, Chaoqun Hong, Yong Han, Pengcheng Huang, and Yanhui Zhuang (2021). "MH Pose: 3D Human Pose Estimation based on High-quality Heatmap". In: *2021 IEEE*

International Conference on Big Data (Big Data). IEEE, pages 3215–3222 (cited on page 34).
Zhou, Yijun and James Gregson (2020). "WHENet: Real-time Fine-Grained Estimation for Wide Range Head Pose". In: *BMVC* (cited on pages 34, 36, 59, 60).
Zhu, Guangming et al. (2022). "Scene graph generation: A comprehensive survey". In: *arXiv preprint arXiv:2201.00443* (cited on pages 82, 83).
Zhu, Xiangyu, Xiaoming Liu, Zhen Lei, and Stan Z. Li (2019). "Face Alignment in Full Pose Range: A 3D Total Solution". In: *PAMI* 41, pages 78–92 (cited on pages 35, 60, 61).
Zhu, Yaohui, Shuqiang Jiang, and Xiangyang Li (2017). "Visual relationship detection with object spatial distribution". In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, pages 379–384 (cited on page 82).
Zhu, Yi et al. (2020). "A comprehensive study of deep video action recognition". In: *arXiv preprint arXiv:2012.06567* (cited on page 75).

Appendix A

LAEO demo

In this chapter, we illustrate a demo implemented using the Head Pose Estimation method developed in Chapter 4. The implementation has been developed for the RAISE project and it is available on our GitHub page, here. An example –only working on single images– can be found on our Hugging Face repository.

A.1 Real Time LAEO demo

The real-time demo has been developed in Python using a pre-trained Centernet network, precisely this one. And on top of it, we used our HHP-Net and weights can be found here.

The implementation is straightforward and it is a stack of the two network with some utilities to show the input of the webcam and the head direction in the form of arrows projected on the image plane.

Algorithm 2 LAEO demo

$I \leftarrow ImageRead \leftarrow webcam$
$Keypoints \leftarrow I$
$HeadPose \leftarrow keypoints$
$HeadPose_{2d} \leftarrow$ ProjectTo2d($HeadPose$)
$I^* \leftarrow$ PlotOnImage($HeadPose_{2d}$)
$I^* \leftarrow$ PlotOnImage($keypoints$)
return I^*

The performances are of course machine dependent and with no GPU available but only a CPU the speed is at approximately 1 fps, taking into account that no real efficient mechanism has been implemented (e.g. TensorFlow rebuild for some additional Intel operation set, customised on the hardware, OpenCV build, etc.). On a device with a GPU using 3Gb of

memory, it shows the results at 7 fps. Also here no procedures to gain some speed has been employed because it is a demo or proof-of-concept and not a market application. However, to gain some speed and have lighter computations, it is possible to adopt a lighter Centernet version (e.g. MobileNet), prune out HHP-Net with small degradation in performance; on top of it, using multiprocessing, like acquiring frames almost in parallel and not sequentially with the network inference may be of great help.

Zedcam implementation

We also implemented a demo version which uses the ZED Camera (developed by StereoLabs) which is a stereo camera enabled with depth estimation. This make possible to extend the LAEO algorithm in 3D thanks to the depth estimation and the application of a 3D object detector. We have not developed it, because the LAEO algorithm working on the image plane satisfied the requirements for our project, but we started experimenting with some depth hints. In this case, if 3D depth estimation is needed, the computational burden increases quite dramatically.

2D projection of 3D directions

The projection of the Head Pose from 2D to 3D and its representation need to be explained. The projection from 3D to 2D is performed in the following way:

$$\begin{aligned} x &= sin(yaw) \\ y &= -cos(yaw) \cdot sin(pitch) \end{aligned} \tag{A.1}$$

where x, y are the pixel coordinates, the yaw, pitch, roll representation has a fixed reference system which is identified by the unit vector $\hat{z}$ exiting perpendicular from the frame; so the xy plane is identified by $\hat{z}$. The equation set is a known transformation [1]. Another methodologies of projection tries to maintain some 3D information, projecting the 3 vectors identifying the

[1] Theory and Application of Kane's Method, by Roithmayr, Carlos M.; Hodges, Dewey H

Fig. A.1 Example of multi-person environment, where red arrows are people not involved in LAEO interaction, whereas green is over thresholds. The bigger match is highlighted with a red line connecting the LAEO couple. This frame was taken in a 3D scenario and it is possible to see how the depth usage can boost the performance of the algorithm.

3 directions on the image plane, each vector is integral with one head axis:

$$\begin{aligned} x1 &= cos(yaw) \cdot cos(roll) \\ x2 &= cos(pitch) \cdot sin(roll) + cos(roll) \cdot sin(pitch) \cdot sin(yaw) \\ y1 &= -cos(yaw) \cdot sin(roll) \\ y2 &= cos(pitch) \cdot cos(roll) - sin(pitch) \cdot sin(yaw) \cdot sin(roll) \\ z1 &= sin(yaw) \\ z2 &= -cos(yaw) \cdot sin(pitch) \end{aligned} \tag{A.2}$$

x, y are the same axis a portrait would have if taken perfectly in front, and the z axis point outwards of the portrait. Each vector is identified by two coordinates, and the starting point is taken at the centroid of the head, so every above vector is defined up to a scale parameter.

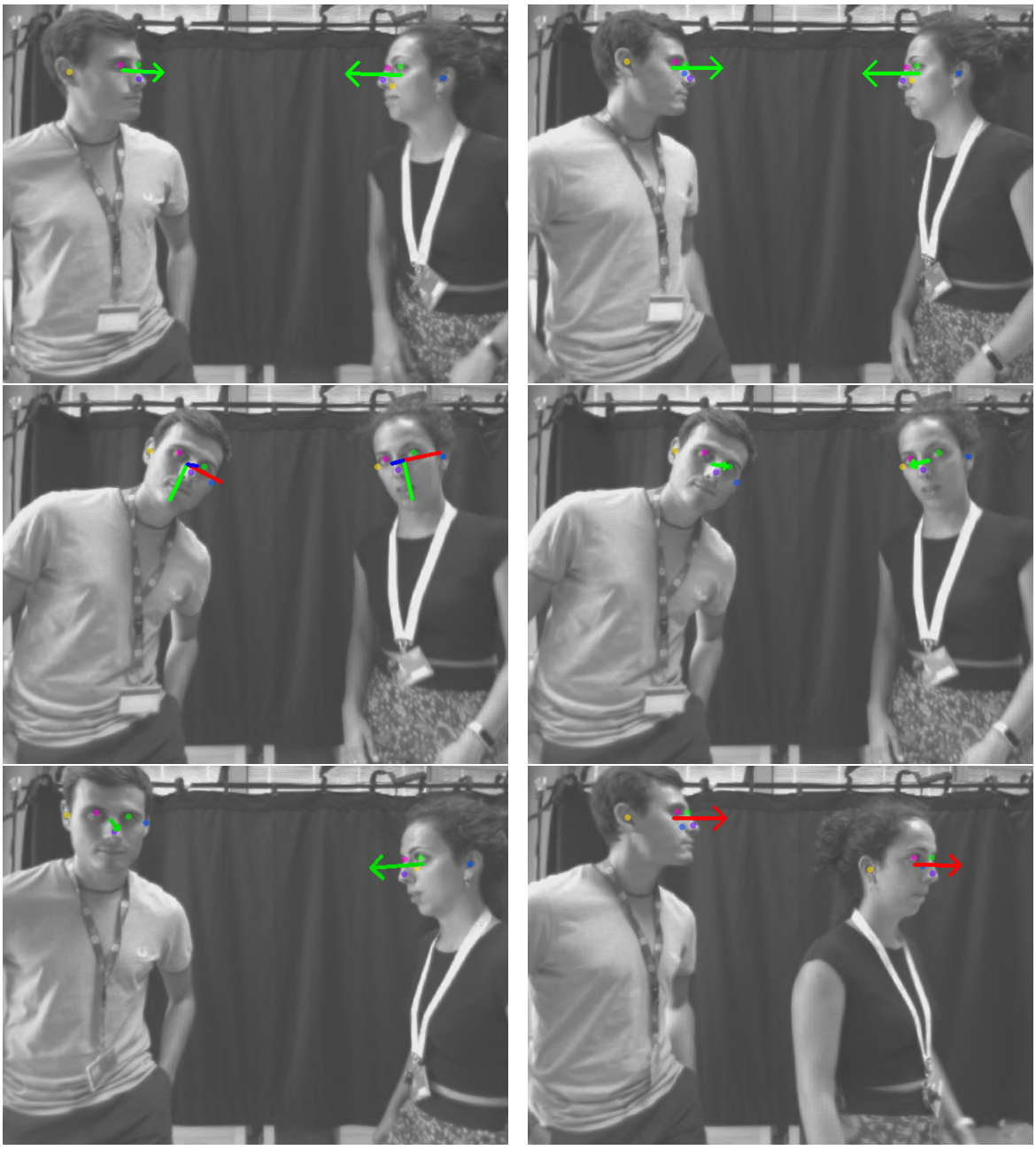


Fig. A.2 Output examples: Green arrow LAEO, red arrow not-LAEO. In the first row two interactions with a mutual gaze are shown: LAEO. In the second row, the same frame with two different representations of the head pose, on the left, the three-axis projected in 2D, and on the right the usual vector. The third row presents a misclassified example, where the guy looks towards the camera but the 2D projections and incorrect thresholds lead to a misclassification; on the right, a correct non-LAEO interaction.

Appendix B

Dataset: Upper-Body Human Motion

During our research, we acquired a small dataset of 320 videos of upper-body actions (person seated at a table moving objects) with a stereo camera and a second low-resolution camera. The dataset encompasses 10 different actions performed twice by 16 participants.

B.1 Dataset design

The dataset design started thinking of some simple actions were primitives of motions could have been easily recognised and with some actions sharing some of the same primitives. Moreover, we decided to have only actions involving the hands and the upper body, to not consider other body motions in our analysis. The actions chosen are:

1. **drinking**: the subject was asked to reach, grasp a glass and drink and return to its rest position.
2. **eat crisps**: the subject was asked to reach a crisp in front of him and eat it, and return to its resting position.
3. **open close bottle**: the subject was asked to reach a bottle, unscrew the cap, screw it back and return to its resting position.
4. **play with Rubik's cube**: the subject was asked to reach a the Rubik's cube, grasp it and do some moves and return to its resting position.
5. **sanitise hands**: the subject was asked to pretend to sanitise their hands with gel, and return to its resting position.
6. **touch bottle**: the subject was asked to reach a bottle in front of him, touch it (with no instructions about how to touch it) and return to its resting position.

7. **touch Rubik's cube**: the subject was asked to reach the Rubik's cube on the table, touch it (with no instructions about how to touch it) and return to its resting position.
8. **transport bottle**: the subject was asked to move the bottle in front of him to a new fixed position and return to its resting position.
9. **transport pen**: the subject was asked to move the pen on the table to a new fixed position and return to its resting position.
10. **transport Rubik's cube**: the subject was asked to move the Rubik's cube on the table to a new fixed position and return to its resting position.

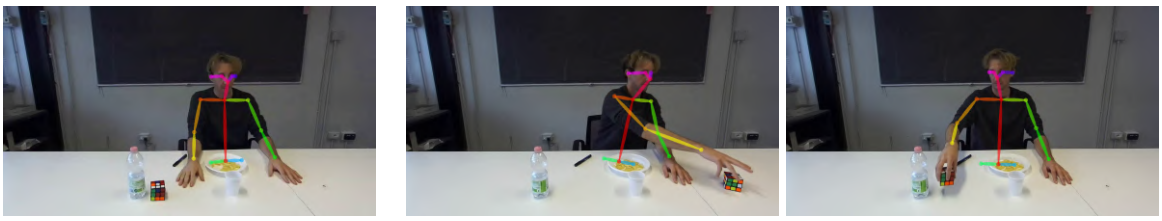
The actions can be divided in: 1,2 are eating, 3,4,5 are actions performed with two hands, 6,7 are touch actions and 8,9,10 are transport actions. The protocol of acquisition consists of defining the positions of objects on the tables but also standardising what to say to the participants. This led also to some different behaviours between people, for example in the performance of touch actions. Some performed it approaching the object with the tip of their fingers, others performed a grasp-and-release action without actually holding the object. It is possible to see that some actions should be simple, with fewer primitives, such as touch, and others instead can have multiple: reach object, grasp, move object, release and return to rest pose.

B.2 Dataset acquisition

The dataset has been acquired with a ZED camera, at 25fps, and with another simple webcam with a lateral view. From the ZED data we extracted the RGB streams, from which we acquired the human pose keypoints in 2D and we use the depth information extracted from the stereo vision to give the 3rd dimensionality to the keypoints. So we obtained a dataset with RGB, depth, 2D and 3D keypoints. For the acquisition, the camera was placed in front of the performer, which was instead seated behind a table. The setting and one example is shown in Section B.2

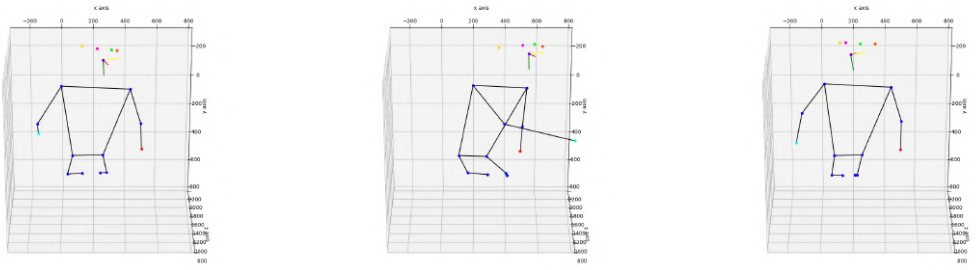
B.3 Dataset annotations

The annotation in the dataset is only at a level of classes, and each video has one label. We used the dataset (action classification task) splitting in train-validation doing a Leave One Out across subjects and averaging over the splits, and this is the suggested procedure with such a limited amount of data and limited subject availability.



(a) (b) (c)

Fig. B.1 Three frames from a transport video. The object disposal can be appreciated from these frames.



(a) (b) (c)

Fig. B.2 Three frames of keypoints from the transport video above.

Appendix C

Network Models

The following models are related to MOSAIC Chapter 7, and are a detailed list of layers and trainable parameters used in some of the experiments. The VAE is composed as:

Modules	Parameters
encoder.conv1.weight	38400
encoder.conv1.bias	128
encoder.conv2.weight	49152
encoder.conv2.bias	128
encoder.conv3.weight	98304
encoder.conv3.bias	256
encoder.conv3_a.weight	393216
encoder.conv3_a.bias	512
encoder.conv3_b.weight	393216
encoder.conv3_b.bias	256
encoder.conv4.weight	196608
encoder.conv4.bias	256
encoder.linear.weight	9830400
encoder.linear.bias	512
encoder.linear_mu.weight	65536
encoder.linear_mu.bias	128
encoder.linear_sigma.weight	65536
encoder.linear_sigma.bias	128
decoder.linear_input.weight	2457600
decoder.linear_input.bias	19200
decoder.anti_conv1.weight	196608
decoder.anti_conv1.bias	256
decoder.anti_conv1_a.weight	393216
decoder.anti_conv1_a.bias	512
decoder.anti_conv1_b.weight	393216
decoder.anti_conv1_b.bias	256
decoder.anti_conv2.weight	98304
decoder.anti_conv2.bias	128
decoder.anti_conv3.weight	49152
decoder.anti_conv3.bias	128
decoder.anti_conv4.weight	38400
decoder.anti_conv4.bias	100

Whereas the Transformer, in the default case is:

Modules	Parameters
class_token	128
transformer_encoder.layers.0.self_attn.in_proj_weight	49152
transformer_encoder.layers.0.self_attn.in_proj_bias	384
transformer_encoder.layers.0.self_attn.out_proj.weight	16384
transformer_encoder.layers.0.self_attn.out_proj.bias	128
transformer_encoder.layers.0.linear1.weight	32768
transformer_encoder.layers.0.linear1.bias	256
transformer_encoder.layers.0.linear2.weight	32768
transformer_encoder.layers.0.linear2.bias	128
transformer_encoder.layers.0.norm1.weight	128
transformer_encoder.layers.0.norm1.bias	128
transformer_encoder.layers.0.norm2.weight	128
transformer_encoder.layers.0.norm2.bias	128
transformer_encoder.layers.1.self_attn.in_proj_weight	49152
transformer_encoder.layers.1.self_attn.in_proj_bias	384
transformer_encoder.layers.1.self_attn.out_proj.weight	16384
transformer_encoder.layers.1.self_attn.out_proj.bias	128
transformer_encoder.layers.1.linear1.weight	32768
transformer_encoder.layers.1.linear1.bias	256
transformer_encoder.layers.1.linear2.weight	32768
transformer_encoder.layers.1.linear2.bias	128
transformer_encoder.layers.1.norm1.weight	128
transformer_encoder.layers.1.norm1.bias	128
transformer_encoder.layers.1.norm2.weight	128
transformer_encoder.layers.1.norm2.bias	128
transformer_encoder.layers.2.self_attn.in_proj_weight	49152
transformer_encoder.layers.2.self_attn.in_proj_bias	384
transformer_encoder.layers.2.self_attn.out_proj.weight	16384
transformer_encoder.layers.2.self_attn.out_proj.bias	128
transformer_encoder.layers.2.linear1.weight	32768
transformer_encoder.layers.2.linear1.bias	256
transformer_encoder.layers.2.linear2.weight	32768
transformer_encoder.layers.2.linear2.bias	128
transformer_encoder.layers.2.norm1.weight	128
transformer_encoder.layers.2.norm1.bias	128
transformer_encoder.layers.2.norm2.weight	128
transformer_encoder.layers.2.norm2.bias	128
input_layer.weight	16384
input_layer.bias	128
classifier.linear.weight	8192
classifier.linear.bias	64
classifier.output.weight	7680
classifier.output.bias	120

The smaller Transformer used is made of 1 encoder layer with 8 attention heads:

Modules	Parameters
class_token	128
transformer_encoder.layers.0.self_attn.in_proj_weight	49152
transformer_encoder.layers.0.self_attn.in_proj_bias	384
transformer_encoder.layers.0.self_attn.out_proj.weight	16384
transformer_encoder.layers.0.self_attn.out_proj.bias	128
transformer_encoder.layers.0.linear1.weight	32768
transformer_encoder.layers.0.linear1.bias	256
transformer_encoder.layers.0.linear2.weight	32768
transformer_encoder.layers.0.linear2.bias	128
transformer_encoder.layers.0.norm1.weight	128
transformer_encoder.layers.0.norm1.bias	128
transformer_encoder.layers.0.norm2.weight	128
transformer_encoder.layers.0.norm2.bias	128
input_layer.weight	16384
input_layer.bias	128
classifier.linear.weight	16384
classifier.linear.bias	128
classifier.output.weight	15360
classifier.output.bias	120

Index